Università degli Studi di Padova

Department of Information Engineering

Ph.D. Course in Information Engineering

Science and Information Technology Curriculum

XXXV Series

Tackling the Distribution Shift in Visual Understanding Applications

Coordinator

Prof. Neviani

Università di Padova

Supervisor

Prof. Zanuttigh

Università di Padova

Ph.D. Candidate

Marco Toldo

Abstract

The rise of deep learning allowed for a significant leap forward in the field of computer vision in terms of the potential expressed by predictive algorithms. While traditional techniques were struggling to offer satisfactory performance, nowadays deep neural networks have been shown to perform strikingly well in a wide range of tasks, paving the way to application-oriented solutions based on such algorithms. Yet, real-world practical applications typically involve learning settings that depart from the standard static single-stage supervised setup, where deep learning algorithms have proven to excel. On the one side, data collection settings seldom match target environments on which the learning algorithm is intended to be applied. This environmental discrepancy calls for domain adaptation, with the final objective to develop predictive models that are robust to change of input distribution between supervised and target setups. On the other side, in an ever-changing world, it is of primary importance to continually acquire new knowledge to keep up with novel tasks, without losing what has been laboriously achieved so far. This presence of dynamically evolving tasks demands for solutions capable of continual learning with just new-task training data at disposal, without erasing previously learned information. Both problems are generally encountered simultaneously, with novel tasks urging for new learning phases to master them, which, in turn, are inevitably introducing dynamically mutable acquisition settings leading to shifts in data distribution. In this thesis we propose to investigate both problems, that is learning under domain and task shift. We first address them individually, and then face them together. In the first part of the thesis, we investigate the domain adaptation problem in its unsupervised form. Unsupervised domain adaptation comes in handy when relying on label-abundant domains for label-costly tasks, such as semantic segmentation. We first provide a broad and detailed overview of different ways in which this problem can be approached, and then propose new techniques to perform adaptation at multiple levels of data representation. In particular, image-level distribution alignment is achieved by image-to-image translation, whereas model-produced representations are adapted at the output-level by domain adversarial schemes and self-training, and at the intermediate level by a clustering based class-conditional regularization. In the second part, we shift our focus to continual task learning. Class incremental learning is crucial to sustain the dynamic expansion of the pool of semantic classes when past training data is no longer accessible. We show that regaining access and replaying former data distributions, still following an exemplar-free setup, represents a successful strategy, especially for long training progressions. We first devise an incremental framework that models the underlying representation drift undergone by old classes, and leverages the up-to-date estimated feature distribution to reproduce samples of no longer available categories in the latent space. Then, we propose to replay image-level information, by retrieving coarsely-

labeled samples, via generative models and web-crawling, and pseudo-labeling them. Still, while the domain adaptation and continual task learning problems have been extensively studied as separate entities, methods that target only one of them perform poorly in a general setting comprising both scenarios. Therefore, in the third part of the thesis we develop a framework apt to deal with the general continual learning problem that encompasses both domain and task shift. We propose a domain stylization scheme to cope with domain incremental shift and spread task-related information across all domains encountered, joined by a robust distillation mechanism to preserve and adapt previously acquired knowledge to new environments.

Sommario

La diffusione di tecniche di deep learning ha fatto compiere un notevole balzo in avanti al campo della computer vision in termini di potenzialità espresse dagli algoritmi predittivi. Dove le tecniche tradizionali faticavano a offrire prestazioni soddisfacenti, oggi le reti neurali hanno dimostrato di poter raggiungere risultati notevoli in un'ampia gamma di applicazioni, rendendo possibile lo sviluppo di strumenti di utilità pratica basati su questi algoritmi. Tuttavia, esigenze concrete nel mondo reale generalmente prevedono contesti di apprendimento che si discostano dal setup standard supervisionato (il quale si sviluppa in una singola fase), in cui gli algoritmi di deep learning hanno indiscutibilmente dimostrato di eccellere. Da un lato, i dati raccolti raramente descrivono appieno gli scenari su cui l'algoritmo finale è destinato ad essere utilizzato. Questa discrepanza richiede la capacità di adattamento ad un diverso dominio, ovvero di sviluppare modelli predittivi robusti ad un cambio di distribuzione in input tra i dati supervisionati e lo scenario di destinazione. Da un altro lato, in un mondo in continua evoluzione, è di primaria importanza acquisire continuamente nuove conoscenze che consentano di tenere il passo con i nuovi obiettivi, senza disperdere ciò che è stato faticosamente appreso in precedenza. Garantire la dinamicità delle capacità predittive richiede strumenti in grado di apprendere in modo progressivo, senza far uso di dati provenienti da fasi precedenti e senza eliminare l'esperienza acquisita in passato. Entrambi i problemi sono spesso sperimentati in contemporanea, dato che la presenza di compiti predittivi incrementali, cioè che richiedono più fasi di apprendimento per essere assimilati, è accompagnata da molteplici fasi di acquisizione di dati, che inevitabilmente comportano variazioni della distribuzione dei dati di input. In questa tesi ci proponiamo di indagare entrambi i problemi, ovvero l'apprendimento con domini e compiti predittivi dinamici, che verranno affrontati prima singolarmente e poi insieme. Nella prima parte della tesi, studieremo il problema dell'adattamento di dominio in forma non supervisionata. L'adattamento non supervisionato di dominio è particolarmente utile quando ci si avvale di domini sorgente con abbondanza di dati per compiti predittivi ad alto costo di supervisione, come la segmentazione semantica. In primo luogo, forniremo una panoramica ampia e dettagliata dei diversi modi in cui questo problema può essere affrontato, per poi proporre nuove tecniche di adattamento a più stadi di rappresentazione dei dati. In particolare, analizzeremo l'allineamento della distribuzione nello spazio di input mediante una tecnica trasferimento di stile; inoltre, proporremo di adattare le rappresentazioni prodotte dal modello sia a livello di output, mediante uno schema adversariale e di auto-apprendimento, che a livello intermedio, mediante una regolarizzazione basata su un algoritmo di clustering. Nella seconda parte, sposteremo la nostra attenzione su un apprendimento dinamico del compito predittivo di classificazione, ovvero quando le classi da apprendere vengono sperimentate in modo incrementale. La ca-

pacità di apprendere classi in modo incrementale è fondamentale per sostenere l'espansione dinamica del gruppo di classi considerate qualora non si avessero più a disposizione dati di addestramento per compiti passati. Mostreremo che il poter recuperare e riprodurre le distribuzioni di dati da fasi di apprendimento precedenti, sempre assumendo di non mantenere nessun dato in memoria tra una fase e l'altra, costituisce una strategia di successo, soprattutto se l'addestramento dura per molteplici fasi incrementali. Per prima cosa, presenteremo un framework incrementale che modella la deriva della rappresentazione subita dalle classi passate e che sfrutta una stima aggiornata della distribuzione dei dati per riprodurre campioni di categorie non più disponibili all'interno di uno spazio latente. In seguito, proporremo di riprodurre le informazioni sotto forma di immagini, recuperando campioni con supervisione limitata, tramite modelli generativi e web-crawling, e pseudo-etichettandoli.

Tuttavia, mentre i problemi di adattamento di dominio e di apprendimento continuo sono stati ampiamente studiati come entità separate, metodi che si limitano ad affrontare sono uno dei due problemi ottengono scarsi risultati in un contesto generale che comprende entrambi gli scenari. Pertanto, nella terza parte della tesi svilupperemo un framework adatto ad affrontare il problema generale di apprendimento continuo che comprenda sia il cambiamento di dominio che quello di compito predittivo. Proporremo una tecnica di stilizzazione di immagini che consenta di affrontare il cambiamento incrementale del dominio e di estendere le informazioni relative al compito predittivo a tutti i domini incontrati, assieme ad un meccanismo di distillazione che preservi e adatti le informazioni precedentemente acquisite a nuovi contesti.

Contents

Abstract v

List of figures xiv

List of tables xxi

1 Introduction 1
1.1 Scene Understanding in the Wild . . . 1
1.1.1 Semantic Segmentation . . . 3
1.1.2 Transfer Learning . . . 4
1.2 Contributions . . . 6
1.2.1 Part I: Unsupervised Domain Adaptation . . . 6
1.2.2 Part II: Class Incremental Learning . . . 7
1.2.3 Part III: Continual Learning under Domain and Task Shift . . . 8

I Learning Under Domain Shift 9

2 Transfer Learning Across Domains 11
2.1 Unsupervised Domain Adaptation . . . 11
2.2 Problem Formulation . . . 13
2.3 Adaptation Focus . . . 15
2.3.1 Input Level Adaptation . . . 15
2.3.2 Feature Level Adaptation . . . 16
2.3.3 Output Level Adaptation . . . 17
2.4 Unsupervised Domain Adaptation Techniques . . . 18
2.4.1 Domain Adversarial Adaptation . . . 19
2.4.2 Generative-Based Adaptation . . . 25
2.4.3 Classifier Discrepancy . . . 29
2.4.4 Self-Supervised Learning . . . 31
2.4.5 Curriculum Learning . . . 34
2.4.6 Multi-Tasking . . . 36
2.5 Experimental Setups . . . 36
2.5.1 Synthetic to Real Adaptation . . . 36
2.5.2 Employed Datasets . . . 37

3 Generative Input Adaptation 41
3.1 Introduction . . . 41
3.2 Cycle Consistent Domain Adaptation . . . 42
3.2.1 Consistency of Input Appearance Across Domains . . . 43
3.2.2 Semantic Consistency Across Domains . . . 44
3.2.3 Consistency of Latent Representation Across Domains . . . 45
3.3 Experimental Setup . . . 46
3.4 Experimental Results . . . 48
3.4.1 Image Domain Translation . . . 48
3.4.2 Adaptation from GTA5 to Cityscapes . . . 50
3.4.3 Adaptation from the SYNTHIA dataset . . . 52
3.4.4 Ablation Study . . . 54

4 Adversarial Output Adaptation 57
4.1 Introduction . . . 57
4.2 Adversarial Adaptation and Self-Training . . . 58
4.2.1 Domain Adversarial Adaptation . . . 58
4.2.2 Class-Adaptive Self-Training . . . 60
4.3 Experimental Setup . . . 62
4.4 Experimental Results . . . 62
4.4.1 Evaluation on the Cityscapes Dataset . . . 62
4.4.2 Evaluation on the Mapillary dataset . . . 64
4.4.3 Ablation Study . . . 66

5 Class Conditional Feature Adaptation 69
5.1 Introduction . . . 69
5.2 Problem Formulation . . . 72
5.3 Orthogonal and Clustered Embeddings . . . 73
5.3.1 Discriminative Clustering . . . 74
5.3.2 Orthogonality of Individual Feature Representations . . . 75
5.3.3 Feature Sparsity . . . 76
5.4 Latent Space Regularization . . . 76
5.4.1 Clustering of Latent Representations . . . 77
5.4.2 Perpendicularity of Latent Representations . . . 79
5.4.3 Latent Norm Alignment Constraint . . . 80
5.5 Experimental Setup . . . 81
5.6 Experimental Results . . . 82
5.6.1 Adaptation from GTA5 to Cityscapes . . . 83
5.6.2 Adaptation from SYNTHIA to Cityscapes . . . 86
5.6.3 Orthogonal and Clustered Embeddings: Ablation Study . . . 89
5.6.4 Latent Space Regularization: Ablation Study . . . 90

II Learning under Task Shift **95**

6 Transfer Learning Across Tasks **97**
6.1 Continual Task Learning . . . 97
6.2 Problem Formulation . . . 99
6.3 Continual Task Learning Techniques . . . 101
6.3.1 Class Incremental Learning . . . 101
6.3.2 Incremental Semantic Segmentation . . . 104
6.4 Experimental Setups . . . 108
6.4.1 Task Incremental Setups . . . 108
6.4.2 Employed Datasets . . . 110

7 Continual Learning by Feature-level Replay **113**
7.1 Introduction . . . 113
7.2 Problem Formulation . . . 115
7.3 Modeling Representation Drift . . . 117
7.3.1 Modeling Feature Drift . . . 118
7.3.2 Modeling Semantic Drift . . . 119
7.4 Training Models of Representation Drift . . . 120
7.4.1 Training Classification Models . . . 120
7.4.2 Training Representation Drift Models . . . 121
7.4.3 Optimization of Model Parameters . . . 122
7.5 Design of Representation Drift Models . . . 123
7.5.1 Modeling Feature Drift . . . 123
7.5.2 Modeling Semantic Drift . . . 123
7.6 Experimental Setup . . . 124
7.6.1 Training Details . . . 124
7.6.2 Implementation Details . . . 126
7.6.3 Evaluation metrics . . . 127
7.7 Experimental Results . . . 127
7.7.1 Comparison with the State-of-the-Art . . . 127
7.7.2 Additional Results . . . 129
7.7.3 Ablation Study . . . 131

8 Continual Learning by Image-level Replay **141**
8.1 Introduction . . . 141
8.2 Problem Formulation . . . 143
8.3 RECALL: Replay in Continual Learning . . . 144
8.3.1 Architecture of Replay Block . . . 145
8.3.2 Background Self-Inpainting . . . 146
8.3.3 Incremental Training with Replay Block . . . 147

8.4 Replay Strategies . . . 149
8.4.1 Replay by GAN . . . 149
8.4.2 Replay by Web Crawler . . . 150
8.5 Experimental Setup . . . 151
8.6 Experimental Results . . . 151
8.6.1 Addition of the last class . . . 153
8.6.2 Addition of last 5 classes . . . 153
8.6.3 Addition of last 10 classes . . . 154
8.6.4 Per-Task and Per-Class Results . . . 159
8.6.5 Ablation Study . . . 159

III Learning Under Task and Domain Shift 167

9 Transfer Learning Across Tasks and Domains **169**
9.1 Introduction . . . 169
9.2 Problem Formulation . . . 170
9.3 Adaptation with Variable Tasks or Domains . . . 171
9.4 Experimental Setups . . . 173

10 Generalized Continual Semantic Segmentation **177**
10.1 Introduction . . . 177
10.2 Problem Formulation . . . 179
10.3 Overview of the Proposed Method . . . 180
10.3.1 Domain Stylization . . . 182
10.4 Learning Across Tasks and Domains . . . 184
10.4.1 Learning New Classes over New Domains . . . 184
10.4.2 Learning New Classes over Past Domains . . . 185
10.4.3 Adapting Old Classes to New Domains . . . 186
10.4.4 Preserving Old Classes on Old Domains . . . 188
10.5 Experimental Setup . . . 189
10.5.1 Incremental Learning Setup . . . 189
10.5.2 Implementation Details . . . 191
10.5.3 Competitors . . . 191
10.5.4 Evaluation metrics . . . 192
10.6 Experimental Results . . . 193
10.6.1 Evaluation on Urban Scenes . . . 193
10.6.2 Evaluation with Worldwide Data . . . 199
10.6.3 Evaluation with Variable Environmental Conditions . . . 200
10.6.4 Ablation Studies . . . 201

11 Conclusion 207

References 210

Acknowledgments 235

Listing of figures

1.1 Overview of visual tasks ranging from whole-image classification (sparse task) to pixel-level semantic segmentation (dense task). 3

2.1 Different settings for domain adaptation, according to how source and target class sets are related. 13

2.2 Typical encoder-decoder architecture for semantic segmentation, highlighting different network stages on which domain adaptation strategies can be applied, from the input image space up to intermediate and output network activations. 17

2.3 Venn diagram of the most popular UDA strategies for semantic segmentation. Each method falls under the set representing the adaptation techniques that are used. 18

2.4 Training of a generative adversarial network. The upper part displays the discriminator's update phase, while in the lower part the generator's update step is shown. 21

2.5 Graphical representation of the standard adversarial adaptation strategy. A domain discrimination captures the statistical discrepancy between source and target representations (*e.g.*, segmentation network's output or features maps computed from one or the other domain). Its supervisory signal is then exploited to perform domain alignment. 22

2.6 Graphical representation of an output adversarial adaptation strategy, where domain alignment is performed indirectly by bridging the distribution gap between source annotation maps and network predictions from either source or target domains. 23

2.7 Overview of the generative-based adaptation approach built upon cycle-consistent image-to-image translation. Source translated input images are exploited as a form of target-like artificial supervision during the learning process. 25

2.8 Comparison between synthetic and real images. 37

2.9 Images from real-world datasets. 38

2.10 Images from synthetic datasets. 39

3.1 Architecture of the proposed framework. Yellow blocks correspond to the CycleGAN module for image-to-image translation. Original and translated scenes from both source and target sets are projected by the encoder to a latent space on which we apply an extra couple of domain discriminators (green blocks). Structural consistency on generated samples is enforced by the cycle-consistency constraint, whereas semantic uniformity throughout image mapping is promoted by the semantic loss. The segmentation network is reported in blue. 43

3.2 Examples of image translations. In the first up-left quadrant, we move from GTA5 to the adapted image space resembling that of Cityscapes, and back to the reconstructed space in the GTA5 domain. The second quadrant (up-right) shows the translation in opposite direction, *i.e.*, from Cityscapes to GTA5, and then back to the original real Cityscapes domain. The third (down-left) and the fourth (down-right) quadrant are analogous to the previous ones, but with SYNTHIA in place of GTA5. 49

3.3 Semantic segmentation of sample scenes from the Cityscapes validation set when adapting from GTA5 (rows 1 to 4) and SYNTHIA (rows 5 to 8). . . 53

4.1 Architecture of the proposed approach. The semantic segmentation network G is trained with the combination of four losses: a supervised cross entropy on source data $\mathcal{L}_{G,0}$, a double adversarial framework $\mathcal{L}^{s,t}_{G,1}$ and $\mathcal{L}^{t}_{G,2}$, and a self-training module $\mathcal{L}_{G,3}$ with class-wise and time-varying adaptive thresholding mask T_f. 59

4.2 Semantic segmentation of sample scenes from the Cityscapes (a) and Mapillary (b) validation sets. Four UDA scenarios are considered, *i.e.*, adaptation from GTA5 or SYNTHIA source datasets to the Cityscapes or Mapillary target counterparts. 65

4.3 Time average over the initial to current step interval of per-class confidence thresholds for different classes and at different training steps (GTA5 to Mapillary adaptation). 67

5.1 The proposed domain adaptation scheme aims to enforce latent space regularization across domains. Feature clustering objectives are proposed, along with orthogonality, sparsity and norm alignment constraints to achieve this goal. By doing so, features in the previous step (in light gray) are driven to new locations (colored) where features of the same class are clustered, while features of distinct classes are pushed away. To further improve the adaptation performance, features of distinct classes are forced to be orthogonal and sparse, and their norm to be consistent across domains. 70

5.2 Overview of the proposed OCE approach. Features after supervised training on the source domain are represented in light gray, while features of the current step are colored. A set of techniques is employed to better shape the latent feature space spanned by the encoder. Features are clustered and the clusters are forced to be disjoint. At the same time, features belonging to different classes are forced to be orthogonal with respect to each other. Additionally, features are forced to be sparse and an entropy minimization loss could also be added to guide target samples far from the decision boundaries. 73
5.3 Visual summary of the proposed LSR strategy and of the effect of its application on the feature space. The three proposed space shaping constraints are from left to right: Class Clustering (5.4.1), Prototypes Perpendicularity (5.4.2), Norm Alignment and Enhancement (5.4.3). Furthermore, we apply entropy minimization [1]. 77
5.4 Semantic segmentation of sample scenes from the Cityscapes validation set when adaptation is performed from the GTA5 and SYNTHIA source datasets and the DeepLab-V2 with ResNet-101 backbone is employed. 87
5.5 Semantic segmentation of sample scenes from the Cityscapes validation dataset when adaptation is performed from the GTA5 and SYNTHIA source datasets and the DeepLab-V2 with ResNet-101 backbone is employed. 88
5.6 T-SNE computed over features of a single image of the Cityscapes validation set when adapting from GTA5. 90
5.7 Similarity scores computed over images of the Cityscapes validation set when adapting from GTA5. 91
5.8 Sparsity scores computed over images of the Cityscapes validation set when adapting from GTA5. 91
5.9 (a) Ablation analysis on impact of different loss components. (b) Example of semantic label downsampled via nearest (top) or by frequency-aware (bottom) approaches. 92
5.10 Class distribution of segmentation maps downsampled either via histogram-aware or via nearest neighbor. 92
5.11 Feature distribution before and after adaptation. 93
5.12 Average inter-prototype angle, comparison between Source Only and LSR. 93
5.13 T-SNE plots comparing class-wise feature distribution before and after adaptation. Points are color coded according to the legend of Figure 5.5. 94

6.1 Graphical representation of the class incremental continual learning framework. The model is updated to recognize new classes over time without forgetting previously learned ones. 100

6.2 Overview of the different setups for class incremental continual learning in semantic segmentation. The black class represents the *background* class and the white one represents the *void/unlabeled*. 110

7.1 In CIL, the process of training models on new classes causes representations of past categories to constantly change. Yet, unavailability of data of former classes prevents from tracking their evolution in feature spaces, leading to *evanescence* of their representations and, in turn, to catastrophic forgetting. We propose to model representation drift on a semantic level (*i.e.*, the relationship among novel and past classes) and on a feature level (*i.e.*, the combined evolution of features learned by a classification model), and exploit it to infer up-to-date representations of past classes. By injecting old-class knowledge into the learning process, we counteract forgetting. 114

7.2 Illustration of the learning phase (LP) of semantic and feature drift models. SD (top): we model the relationship between representations of new and old classes at the beginning of the incremental step. FD (bottom): we capture the evolution of representations of novel classes within the incremental step. 117

7.3 An outline of our CIL framework augmented with the proposed representation drift models. A classifier is trained on the dataset $\mathcal{T}_t$ at step t (L^t_{ce}). Feature knowledge distillation (L^t_{fkd}) is used to reduce feature drift. We exploit the drift models learned in LP (Figure 7.2) to infer revived evanescent representations (RERs), which are leveraged by L^t_{rd} to inject past knowledge into the current training procedure. 118

7.4 Per step average top-1 accuracy (%) on the CIFAR100, CUB200-2011 and TinyImageNet datasets. 130

7.5 Per step and per-task average forgetting (%) curves. 132

7.6 Analysis of normalised distance between estimated prototypes of classes seen at steps $t = 0$ and $t = 1$, captured at the beginning (left) and end (mid) of step $t = 1$. We report the absolute value of the difference of the two measures (right). We replicate the analysis for the 20-step incremental setup, over the CIFAR100 (top), TinyImageNet (middle) and CUB200 (bottom) datasets. 133

7.7 Analysis of average distance between estimated (*revived*) and evanescent prototypes of old classes. 134

7.8 Average Euclidean distance between the estimated (*revived*) and evanescent prototypes of old classes. 135

7.9 Feature representations of the first four learned classes (CIFAR100, 20 steps) extracted from samples of test set (dots), along with their prototypes computed over available training data (squares), over test data (diamonds) and estimated prototypes (diamonds). In the lower plot, decrease in transparency and increase in brightness indicate that representations are extracted at progressively increasing incremental steps (*i.e.*, at steps 0, 10 and 20). 137
7.10 Relationship between top-1 accuracy (%) and *normalized* Euclidean distance between estimated and evanescent old-class prototypes. Each point depicts a single training phase, and the decrease in transparency indicates progressively increasing incremental steps. For each step, accuracy values have been averaged over all classes observed so far, and distances are averaged over all past classes. 138
7.11 Average entropy (H) and cross-entropy (CE) of p_F (CIFAR100, 20 steps). 138
7.12 Left: average entropy of $p_c(F_i)$ (Eq. (7.11)). Mid-left and mid-right: entropy and cross-entropy with respect to ground-truth of $p_F(c)$ (Eq. (7.10)). Each feature F_j is extracted from a test image belonging to an old class, and π_j are the estimated prototypes of old classes. Right: Top-1 accuracy for the CIFAR100 (top), TinyImagenet (middle) and CUB200 (bottom) (all models were evaluated for 20 incremental steps). 139

8.1 Replay images of previously seen classes are retrieved by a web crawler or a generative network and further labeled. Then, the network is incrementally trained with a mixture of new and replay data. 142
8.2 Overview of the proposed RECALL: class labels from past incremental steps are provided to a Source Block, either a web crawler or a pre-trained conditional GAN, which retrieves a set of unlabeled replay images for the past semantic classes. Then, a Label Evaluation Block produces the missing annotations. Finally, the segmentation network is incrementally trained with a replay-augmented dataset, composed of both new classes data and replay data. 145
8.3 Background self-inpainting process. 147
8.4 Evolution of performance in terms of mIoU on the 10 tasks of the 10-1 disjoint setup. 153
8.5 Qualitative results on disjoint incremental setups: from top to bottom 15-1, 15-5 and 10-1. 155
8.6 Qualitative results on disjoint incremental setups. 156
8.7 Per-step prediction maps on the 15-1 disjoint incremental setup for different training strategies. 157
8.8 Memory occupation in the disjoint scenario. 161
8.9 Distinct interleaving policies in the 15-1 disjoint setup. 162

8.10 Original images from the incremental Pascal VOC dataset, together with replay data generated by GAN or retrieved by Flickr's web crawler. 165

9.1 Domain distribution shift due to change of geographic location (*urban* setting). 174
9.2 Domain distribution shift due to change of geographic location (*worldwide* setting). 174
9.3 Domain distribution shift due to *environmental* factors. 175

10.1 High-level view of our approach. Transparency decrease (top→down and left→right) indicates progression through learning steps. Colored task icons denote presence of supervision within training data, grayscale ones signal lack of supervision. At each step, we leverage training data to *learn* new classes on the new domain. Domain stylization allows to reiterate old-domain distribution, crucial to *learn* new tasks and *preserve* old ones on former domains, and to *adapt* old-domain old-task knowledge to new domains. 178
10.2 Overview of the class and domain incremental setup. When training, each step sees training data from a new domain, labeled on a new class set. When testing, performance is measured on all domains and classes experienced so far. 181
10.3 Domain stylization: to access no longer available old domain data, we stylize new domain data according to former-domain styles stored in a memory bank. In addition, we perform self-stylization to improve generalization performance. 183
10.4 Model architecture: we decompose class and domain IL into simpler subproblems, each addressed by a suitable objective; we learn new classes on new domains (top-left panel), we learn new classes on old domains (top-right panel), we adapt old classes to new domains (bottom-left panel) and preserve old classes on old domains (bottom-right panel). 185
10.5 Qualitative results on CS $\rightarrow$ BDD $\rightarrow$ IDD domain setup and $\mathcal{C}_{bgr} \rightarrow \mathcal{C}_{stat} \rightarrow \mathcal{C}_{mov}$ class setup. 197
10.6 Different ways of pseudo-labeling ($t = 2$). White regions correspond to the *ignore* label. 203
10.7 Domain-knowledge transfer (mIoU$\uparrow$ (%)). 205
10.8 Task-knowledge transfer ($\bar{\Delta}\downarrow$). 206

Listing of tables

2.1 Formal definition of notation used throughout Part I of the dissertation. . . 14

3.1 Results in terms of per-class and mean IoU on the Cityscapes validation set when adapting from GTA5 (top) and SYNTHIA (bottom). The highest values have been highlighted in bold. 50

3.2 Ablation study on the impact of different objectives. Results are reported in terms of per-class and mean IoU on the Cityscapes validation set when adapting from GTA5 (top) and SYNTHIA (bottom). The highest values have been highlighted in bold. 54

4.1 Per-class and mean IoU on the four considered UDA scenarios, *i.e.*, adaptation from GTA5 or SYNTHIA source datasets to the Cityscapes or Mapillary target counterparts. Results are reported on validation sets of the target domains. The highest values have been highlighted in bold. 63

4.2 Ablation results on the impact of loss components (GTA5 to Mapillary adaptation). 67

5.1 Numerical results of the GTA5 and SYNTHIA to Cityscapes adaptation scenarios in terms of per-class and mean IoU. Evaluations are performed on the validation set of the Cityscapes dataset. In all the experiments, the DeepLab-V2 segmentation network is employed, with VGG-16 (top) or ResNet-101 (bottom) backbones. The mIoU* results in the last column refer to the 13-classes configuration, *i.e.*, classes marked with * are ignored. MaxSquares IW$^{(r)}$ denotes our re-implementation, as original results are provided only for the ResNet-101 backbone. The highest values have been highlighted in bold. 84

5.2 Comparison of adaptation strategies in terms of IoU, mIou and mASR (%) (Section 5.6). The mIoU^1 and mASR^1 restrict to 13 classes, ignoring the ones with same superscript. The highest values have been highlighted in bold. 86

5.3 Ablation results on the contribution of each adaptation module in the GTA5 to Cityscapes scenario and with ResNet-101 as backbone. 89

6.1 Formal definition of notation used throughout Part II of the dissertation. . 99

7.1 Formal definition of key notation used throughout Chapter 7. 115

7.2 Architecture of the Gaussian Model (GM) identified by MLP. 124

7.3 Architecture of the Variational Model (VM) identified by conditional VAE. 124
7.4 Per-class average top-1 accuracy (%) on CIFAR100 and TinyImageNet, with different incremental setups. The highest values have been highlighted in bold. 128
7.5 Per-class average top-1 accuracy (%) on CUB200, with different incremental setups. The highest values have been highlighted in bold. 129
7.6 *acc1*:*acc2* top-1 accuracy (%) where *acc1* is the class-wise average accuracy (Eq. (7.7)) and *acc2* is the class- and step- wise average accuracy (Eq. (7.8)). Both measures are computed at the end of the last incremental step. 129

8.1 Formal definition of key notation used throughout Chapter 8. 144
8.2 mIoU on Pascal VOC2012 for different incremental setups. Results of competitors in the upper part come from [2, 3], while we run their implementations for the new scenarios in the lower part. The highest values have been highlighted in bold. 152
8.3 Per-class IoU of compared methods in disjoint experimental protocol on multiple scenarios of Pascal VOC 2012. 158
8.4 Per-round accuracy measures in the 10-1 disjoint scenario. In the top part we report the PA (left) and IoU (right) of the last class currently introduced. The bottom part, instead, shows the mean IoU over the old classes up to the ongoing step (left), as well as the overall mean IoU including the new classes (right). The classes added at each incremental step are: 1:*dining table*, 2:*dog*, 3:*horse*, 4:*motorbike*, 5:*person*, 6:*potted plant*, 7:*sheep*, 8:*sofa*, 9:*train* and 10:*tv/monitor*. The highest values have been highlighted in bold. 160
8.5 mIoU results showing the contribution of each module of our framework, D: Disjoint, O: Overlapped. 162
8.6 mIoU on VOC2012 disjoint 15-1 with replay data. G: GAN, F: Flickr. Naïve: only decoder of last step is used for pseudo-labeling, Ours: our complete approach (RECALL) is used. 163
8.7 Class mapping between Pascal VOC and ImageNet datasets. The table shows the 3 best matching ImageNet classes for each Pascal VOC 2012 class. (*): matching classes for *tv/monitor* are not computed since replay data is not needed. 164

10.1 Formal definition of key notation used throughout Chapter 10. 180
10.2 Training objectives: the *n*/*o* superscripts denote the use of new/old domain data, with $\tilde{\cdot}$ implying stylization. 182
10.3 Split of Cityscapes's (CS) and Shift's class sets following the criterion proposed by [4]. 190
10.4 Class and domain incremental sets. 190

10.5 Experimental results on CS $\rightarrow$ BDD $\rightarrow$ IDD domain setup and $\mathcal{C}_{bgr} \rightarrow \mathcal{C}_{stat} \rightarrow \mathcal{C}_{mov}$ class setup. The best values have been highlighted in bold. . . 193
10.6 Experimental results on BDD $\rightarrow$ IDD $\rightarrow$ CS domain setup and $\mathcal{C}_{bgr} \rightarrow \mathcal{C}_{stat} \rightarrow \mathcal{C}_{mov}$ class setup. The best values have been highlighted in bold. . . 194
10.7 Experimental results on IDD $\rightarrow$ CS $\rightarrow$ BDD domain setup and $\mathcal{C}_{bgr} \rightarrow \mathcal{C}_{stat} \rightarrow \mathcal{C}_{mov}$ class setup. The best values have been highlighted in bold. . . 195
10.8 Generalization performance (Γ_t^{gen}) as mIoU on Mapillary's test set ($\mathcal{C}_{bgr} \rightarrow \mathcal{C}_{stat} \rightarrow \mathcal{C}_{mov}$ setup). The best values have been highlighted in bold. 196
10.10 Experimental results on CS $\rightarrow$ BDD $\rightarrow$ IDD domain setup and $\boldsymbol{\mathcal{C}_{bgr} \rightarrow \mathcal{C}_{mov} \rightarrow \mathcal{C}_{stat}}$ class setup. The best values have been highlighted in bold. . 198
10.11 Experimental results with **DeeplabV3-ResNet101**. The best values have been highlighted in bold. 199
10.12 Experimental results on the Mapillary dataset. The best values have been highlighted in bold. 200
10.13 Experimental results on the Shift dataset. The best values have been highlighted in bold. 201
10.14 Ablation study on the contribution of loss components. The $\mathcal{L}_{kd}^{n}$ notation here implies that pseudo-labels are generated leveraging new-domain input samples. The best values block-wise have been underlined. 202
10.15 Ablation study on pseudo-labeling schemes. The best values have been highlighted in bold. 203
10.17 Ablation study on stylization ($\beta = 0.01$ corresponds to the default configuration). The best values have been highlighted in bold. 204

1 Introduction

1.1 Scene Understanding in the Wild

In recent years, deep learning techniques have reached considerable success in many visual applications, enabling to achieve outstanding results where traditional methods struggled to provide satisfactory performance. We refer to [5, 6, 244] for a comprehensive introduction to computer vision and to recent advances brought by deep learning to this field. However, deep learning solutions typically thrive in a standard supervised learning setting, that is, when it is possible to assume the availability at once of a large training set containing data with the same statistical properties as the target data labeled according to the problem at hand.

When addressing practical applications in real-world environments, however, we are likely to face some limitations introduced by the strict constraints of a fully supervised setup. First of all, a large amount of training data for the considered setting and problem is typically not available, although required to fully reach deep models' potential. While large-scale generic datasets are publicly available, the acquisition and annotation of a significant amount of data for a specific setting is generally too expensive and time consuming to be carried out. These considerations are especially relevant for dense prediction tasks, such as the semantic image segmentation, which requires a noticeable amount of labeled data to be effectively tackled by deep models, while, in turn, the pixel-level annotations needed are extremely expensive to be collected. Hence, exploiting the information encapsulated within a model trained on correlated domains, yet distinct from the target one, could be extremely beneficial when data scarcity is a serious concern [7]. These shortcomings have encouraged the proliferation of Domain Adaptation (DA) techniques, able to transfer the learned knowledge from a generic

source dataset to the target data of the problem at hand. Target data can be either partially supervised, *i.e.*, when a small amount of labeled data for the target set is available, or completely unsupervised (thus leading to the so called Unsupervised Domain Adaptation, *i.e.*, UDA), when no labeling information or even no data at all, is available for the target domain.

Another common limitation is that the available training data may only be partially annotated, or it could not exactly match the overall target problem. Following a rapid increase of interest in these types of learning scenarios, recent research effort has been directed toward the exploitation of the labeled data belonging to a different, yet related, problem, or comprising a set of classes different from the target one. In addition, in many cases the target problem may not even be completely defined at the starting point, but it may evolve as the learning progresses, and new classes or new tasks are gradually added to the original problem. Continual Learning (CL) strategies deal with these dynamic learning frameworks, aiming at progressively mastering new tasks, while preserving knowledge of past ones, without re-training the learning model from scratch. In particular, in the so called Class Incremental Learning (CIL) each progressive task corresponds to a new set of semantic classes to be recognized, along with those observed in the past. The challenge stands in the partial incremental supervision, as training data at each learning step is available only for the novel task.

The aforementioned limitations have been generally addressed by research endevours as separate problems, that is, either focusing on transfer learning across domains in static frameworks, or addressing dynamic learning schemes where distribution shifts are present only within the the input space (*e.g.*, continual domain adaptation) or label space (*e.g.*, class incremental learning). Nonetheless, a more general and more realistic continual learning setup could involve both domain and task shifts altogether. This should mimic the human learning behavior, able to adapt and learn new concepts in different environments, while being fed with data in a continuous flow of information. In other words, addressing this more general problem entails pursuing backward and forward knowledge transfers across multiple domains and tasks.

This general discussion on the need for transfer learning applies to many learning models and target problems, but becomes very relevant when a huge amount of data and a large computational effort for training are needed. In particular, this is the case of image and video understanding problems, which are nowadays typically solved with complex deep learning models. For this reason, transfer learning for this kind of problems has been widely studied and many solutions have been proposed especially in the image classification field, which is the simplest and most classical global-level image understanding problem. In this dissertation we mostly focus on the challenging semantic segmentation task where, differently from image-level classification, a dense pixel-level labeling is performed. This problem is more demanding but also particularly interesting since the labeling operation is highly time-

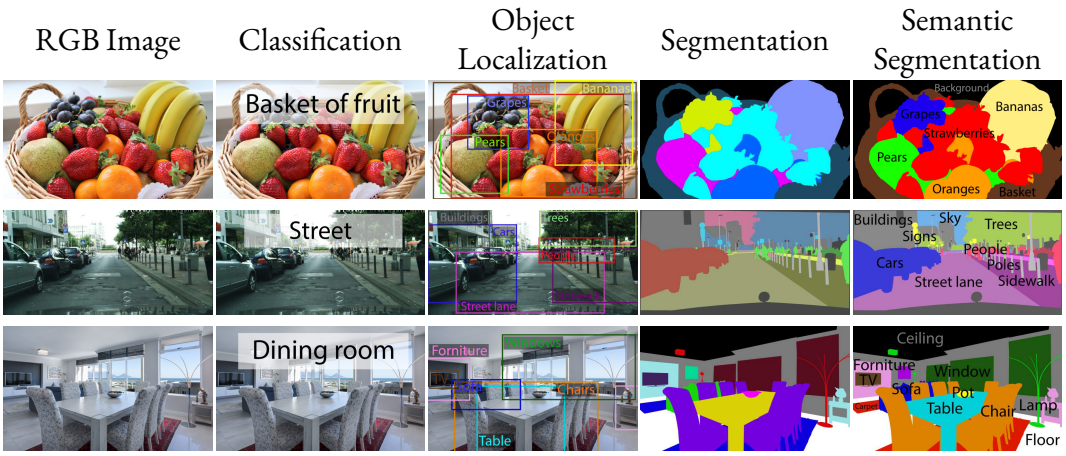


Figure 1.1: Overview of visual tasks ranging from whole-image classification (sparse task) to pixel-level semantic segmentation (dense task).

consuming (much more than in image classification), thus making the construction of complete and large training sets more difficult.

In the remainder of this section we will introduce the semantic segmentation task and provide a formal definition of transfer learning in relation to domain adaptation and class incremental learning.

1.1.1 Semantic Segmentation

Semantic segmentation is one of the most challenging tasks in automatic visual understanding, aiming at a deeper perception of the image content if compared with simpler problems like whole-image classification or object detection [6]. A graphical overview of the most common visual tasks is presented in Figure 1.1. In *image classification*, a single label is assigned to the whole image and denotes the pre-dominant object in the scene. In *object localization*, the objects are identified by means of a bounding box and a label is assigned to each box. In *image segmentation*, the scene is clustered into regions corresponding to the various objects and structures but the regions are not labeled. *Semantic segmentation*, instead, is the task of providing each pixel in an image with a label denoting its semantic content. For this reason, it is often referred to as a dense labeling task, with respect to other simpler problems where coarser semantic information is given as output. Semantic segmentation is a popular research field, with a huge number of approaches that have been proposed to address it. In particular, segmentation methods have recently made a substantial leap forward in terms of state-of-the-art performance thanks to the rise of deep learning.

Semantic segmentation took its first steps as an enriched representation learning and un-

derstanding of a scene with respect to the simpler task of image classification: the advent of novel real-world applications (*e.g.*, autonomous driving) demanding to reach an higher level of scene interpretation, together with novel deep learning architectures and paradigms to enable improved performance, have paved the way to the increase in popularity of the semantic image segmentation task.

While image classification allows to recognize the semantic content of an image at a macroscopic level (*i.e.*, a single label is assigned to the whole image), semantic image segmentation generates a pixel-wise labeling of each element of an image (*i.e.*, a single label is assigned to each image pixel). Being the former a much simpler task, it has been tackled since long time with both traditional techniques (such as SVM or LDA) and, more recently, with deep learning ones. For this reason, some early-stage works in semantic segmentation build up from classification works, adapting and extending them. The most recent state-of-the-art approaches rely on an encoder-decoder structure, composed by an encoder and a decoder to extract semantic clues ranging from local to global, while retaining input spatial dimensionality [6].

Starting from the well-known Fully Convoultional Networks (FCN) architecture [8], many models have been proposed, such as PSPNet [9], DRN [10] and the various versions of the DeepLab architecture [11–13]. These models have shown to achieve outstanding performance, but this is strictly related to the availability of a massive amount of labeled data required for their training. For this reason, even though the pixel-wise annotation procedure is highly expensive and time consuming, many datasets have been created: for example, Cityscapes [14], BDD100k [15], IDD [16], Mapillary [17], GTA5 [18] and SYNTHIA [19] for urban scenes; Pascal VOC [20], MS-COCO [21] and ADE20K [22] for visual objects in common contexts; NYUD-v2[23] and SUN-RGBD [24] for indoor scenes with depth information.

1.1.2 Transfer Learning

In the following, we provide a formal definition of the Transfer Learning problem and we introduce some notation that will be used throughout the dissertation. Let us define a domain $\mathcal{D} = \{\mathcal{X}, \mathbb{P}(\mathcal{X})\}$, where $\mathcal{X}$ is the space of input data and $\mathbb{P}(\mathcal{X})$ is the probability distribution function over that input data. Then, a task $\mathcal{J}$ over the domain $\mathcal{D}$ is a combination of a label space $\mathcal{Y}$ and the predictive function $f(\cdot)$ that models the conditional probability distribution $\mathbb{P}(\mathcal{Y}|\mathcal{X})$. For instance, a possible task might involve recognizing of a set of semantic classes $\mathcal{C}$, at a coarse level (*e.g.*, whole-image classification) or at a fine level (*e.g.*, semantic image segmentation). Thus, any supervised machine learning problem can be generally reconducted to the search for a function $h : \mathcal{X} \rightarrow \mathcal{Y}$ which better approximates the

unknown underlying $f(\cdot)$, by examining a set of labeled training samples drawn from the joint distribution $\mathbb{P}(\mathcal{X}, \mathcal{Y})$ over $\mathcal{X} \times \mathcal{Y}$.

Let us assume that the input data domain $\mathcal{D}$ may be not unique, *e.g.*, there exist separate source $\mathcal{D}_S$ and target $\mathcal{D}_T$ domains (*e.g.*, in classical UDA) or the domain is split into multiple pieces $\mathcal{D}^t$, $t = 0, 1, ..., T$ available for training at separate times (*e.g.*, in generalized CL). Furthermore, over these domains different tasks may need to be solved, *e.g.*, two different tasks $\mathcal{J}_S$ and $\mathcal{J}_T$ could have been respectively chosen for the source and target domains or there can be a sequence of tasks $\mathcal{J}^t$, $t = 0, 1, ..., T$ to be solved at different steps t of the learning process. Then, *transfer learning* is defined as seeking for an improved predictive function $f_T(\cdot)$ over the target domain (or over multiple target domains), relying on useful information extracted from the source task $\mathcal{J}_S$ on $\mathcal{D}_S$, in case $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{J}_S \neq \mathcal{J}_T$, or from target tasks $\{\mathcal{J}^t\}_t$, $\mathcal{J}^i \neq \mathcal{J}^k$, $i \neq k$ [25].

Notice how domain adaptation and class incremental learning can be viewed as two special cases of transfer learning; in the first case, the source and target domains are different while the task is the same, while in the second case the task changes. Continual learning in its more general fashion entails progressively variable domains *and* tasks.

Transfer learning techniques can come in handy in a multitude of practical circumstances. We may, for example, have thoroughly trained a deep model to solve a prediction task by leveraging a set of samples acquired in a context where data collection is rather inexpensive, for instance by making use of artificially generated synthetic data. Then, we might want to apply our prediction framework to different scenarios in the real world, where we may lack the resources to get as many training samples to extensively retrain our model. In this case we are faced with a domain change, while the underlying task we want to solve remains the same. For instance, we could develop an autonomous driving system, which should be capable of understanding its surrounding environment, by exploiting low-cost, abundant and carefully collected synthetic data from a simulator. However, since it is likely that our system will be deployed in the real world, it is of paramount importance to safely transfer skills acquired from the synthetic source domain to the real target one.
Differently, we could have sufficient labeled data to train our model on the current task, but we might want to preserve the model's prediction capabilities on previously acquired tasks for which we no longer have training data. If we can assume that the newly collected data is alike to that used for learning former tasks (*i.e.*, it has the same statistical distribution), then we are just facing a task shift. For instance, we may want to increase the detection proficiency of our autonomous driving agent by refining its classification skills, adding new classes to the pool of semantic categories to be recognized.
In fact, we may have to deal with domain and task shifts together, seeking to improve the predictive ability of our model, while facing data scarcity on some target environments. For

example, our driving agent could be provided with training data from different places around the world in different training phases, each time aiming at gaining new-task knowledge with only the currently experienced data, while the acquired information should be generalizable to different driving conditions and geographic locations.

1.2 Contributions

1.2.1 Part I: Unsupervised Domain Adaptation

We start by dealing with the Domain Adaptation (DA) problem in Part I of the dissertation, with a special focus on its *unsupervised* form (UDA). Under the transfer learning perspective, the considered task (*e.g.*, the set of classes to be recognized) is fixed, whereas the distribution shift involves the statistical properties of input data between source and target domain sets.

We will discuss the domain adaptation of deep neural networks in Chapter 2. In Sections 2.1 and 2.2, we will provide a more detailed description of the problem, which will be introduced in its different configurations according to how the source and target label spaces are related. The adaptation of deep networks to the target domain has been shown to be effectively carried out at different stages, *i.e.*, (i) at the *input level* by translating source images to a new domain more similar to the target one, (ii) at the *output level* by enforcing the alignment of output probability spaces when experiencing data from two separate domains and, finally, (iii) at the *feature level* by constructing a feature space where classes are better distributed, for predictions more robust to change of domain. These goals can be achieved using plentiful different strategies: Section 2.4 will present in detail the most successful technique proposed in the unsupervised domain adaptation field when specifically targeting the semantic segmentation task. The final Section 2.5 of Chapter 2 will detail some of the most common experimental setups to benchmark domain adaptation methods.

In the remaining chapters of the first part of the dissertation we will propose different adaptation frameworks to address the the domain shift at multiple representation levels.

- Chapter 3 will address UDA under a generative perspective, aiming at performing input-level domain alignment by means of image-to-image translation. Target-like artificial supervision will be leveraged to allow the prediction model to directly experience the target distribution during training.

- Chapter 4 will investigate domain adversarial learning, applied within an adaptation framework with multiple discriminators and enhanced by a self-training mechanism. In particular, the output probability maps from both source and target domains will

be aligned to the available source ground-truth, while the discriminator modules will be exploited to measure the prediction accuracy over the unlabeled target images.

- Chapter 5 will target class conditional representation alignment at the feature level by resorting to lightweight clustering and orthogonality constraints, free from the computational burden brought by adversarial learning schemes. We will aim at improving the feature distribution, enforcing intra-class inter-domain alignment and inter-class separation.

1.2.2 Part II: Class Incremental Learning

In the second part of the dissertation, we will analyze Class Incremental Learning strategies, which are designed to handle a change of task over time. Such change is represented by the dynamic expansion of the class set, *i.e.*, the addition of new categories at each new learning step. One of the main objectives is the capability to adapt to the new learning setting, while only leveraging data related to new tasks and without re-training the model from scratch. However, this is highly nontrivial due to the so-called *catastrophic forgetting* phenomenon, *i.e.*, a machine learning model tends to forget knowledge about previous tasks when it learns new ones.

We will start by formally introducing the continual task learning problem in Chapter 6, namely with Sections 6.1 and 6.2. An overview of recent research directions will be further presented in Section 6.3, followed by a detailing of the wide range of experimental scenarios that can be considered for evaluation purposes (Section 6.4). Then, we will show how class incremental learning can be tackled by means of knowledge preservation strategies, based on feature- and image- level replay (Chapters 7 and 8).

- Chapter 7 will address class incremental learning with a focus on the latent space, identifying representation drift as a source of catastrophic forgetting. Feature and semantic drifts are thus learned on the available data (in an exemplar-free setup) and inferred in order to obtain a reliable estimate of feature distribution of past tasks. A simple generative process in the feature space, then, is leveraged to replay past knowledge during the present learning step.

- Chapter 8 will present a replay-base continual learning framework, where image (input) samples of past tasks are generated from scratch by a pre-trained GAN or retrieved by a web crawler. The rehearsal of replay samples will be shown to provide state-of-the-art performance, proving to be extremely beneficial when the incremental learning progresses across a significant number of steps.

1.2.3 Part III: Continual Learning under Domain and Task Shift

In the final part of the dissertation (Part III), we will investigate the still under-explored generalized class and domain incremental learning, for which both domain and task distributions vary over time. The domain shift addressed in Part I under a static perspective now manifests itself in a continual fashion, with multiple input distributions experienced in different learning steps. Domain variability is further joined by mutable task supervision, which characterizes the class incremental learning problem analyzed in Part II. In this more realistic and application-oriented setup, the continual learner must be able to acquire and preserve task-knowledge, while adapting the captured information to all the domains experienced.

First, in Chapter 9 we will provide a more in-depth description of the general continual learning problem (Sections 9.1 and 9.2). In addition, we will overview some relevant works in literature that address domain shift under variable static supervision or dynamic incremental perspective (Section 9.3). We will also discuss the benchmarks used to experimentally simulate a comprehensive continual learning setting, which will be used for validation purposes (Section 9.4).

In Chapter 10, we will then propose a novel learning framework capable to address all the challenges introduced by the new setup, in order to achieve transfer learning across domains and tasks. Where a style transfer mechanism allows to replay domain-specific low-level statistic in order to adapt learned knowledge to former domains, a robust distillation framework enables knowledge of past tasks tasks to be preserved and adapted when encountering new domains.

Part I

Learning Under Domain Shift

2 Transfer Learning Across Domains

2.1 Unsupervised Domain Adaptation

Over the past few years, deep learning has had a groundbreaking impact on the computer vision field. Semantic segmentation is a very challenging task and, before the deep learning revolution, even sophisticated algorithms were achieving only mediocre performance; nowadays, with the advent of deep neural networks we can obtain remarkable results, provided that enough computational resources are available. Nonetheless, the potential stored in deep learning models can be fully unleashed only when sufficiently plentiful and carefully labeled training data is available. The complexity enclosed in the millions of learnable parameters of state-of-the-art deep learning models easily leads to overfitting of the training data, rather than to an enhanced model performance, and this has to be counteracted by using huge datasets for training. A major example of the central role played by the availability of large amounts of training data is the ImageNet large-scale dataset [26], whose contribution in the early development and expansion of deep neural networks for image classification is certainly very relevant.

Unfortunately, the collection and annotation of data samples is often extremely expensive, time consuming and error-prone, since it requires a large amount of human supervision in the process, especially for dense classification tasks such as semantic segmentation. The excessive cost may prevent from gathering enough data to address a new task or to move in a new environment, thus posing a serious threat to the remarkable advance brought by deep learning approaches. Therefore, it may be extremely beneficial to rely on previously built datasets, whenever they share similar properties to the target data. In this way, already avail-

able samples can be efficiently exploited to address the current task, since they belong to a domain correlated to the target one.

Even though the transferring of information from related domains appears quite appealing and fairly straightforward, in practice the process requires careful handling. Deep neural networks typically lack generalization skills; in other words, even a small change in data distribution between training and test statistical properties might cause a severe drop of performance. For this reason, the simple application of a pre-trained model in a new environment is likely to fail, as domain-specific attributes are usually captured alongside domain-invariant ones, thus preventing an effective knowledge transfer. In this scenario, Domain Adaptation comes in handy, as it allows to handle the statistical gap between source and target representations. The ultimate goal of the adaptation effort is to learn a prediction model on a selected task working optimally on both source and target domains, while supervision largely (or solely) comes from the label-abundant source domain. To this end, an efficient transfer of knowledge learned in the source domain to the target one is crucial to eventually reach an overall good performance. Particularly interesting is the Unsupervised Domain Adaptation (UDA) setting, in which target annotations are totally missing. This is an extremely favorable, yet challenging, scenario, as data from the target domain no longer requires expensive labeling.

Recently, the domain adaptation task has been very actively studied in the context of deep learning applied to visual tasks. While deep convolutional frameworks have proven to be capable of learning visual features useful to solve multiple related problems (*e.g.*, image classification, object detection, semantic segmentation), the transferability of those representations typically shows to decrease when moving to deeper network layers [27].

Early works investigating unsupervised domain adaptation for deep networks mainly focused on the image classification task. In many approaches a layer-wise measure of statistical domain discrepancy is jointly estimated and minimized, thus promoting the extraction of domain-invariant feature representations, while discrimination ability is guaranteed by a task-specific loss. Later, adversarial domain adaptation strategies have proven to be extremely successful: the domain discrepancy is expressed in the form of a learnable discriminator and its minimization is performed in an adversarial manner. More details on adversarial learning and its use in domain adaptation will be provided in Section 2.4. This has effectively paved the way for domain adaptation solutions apt to solve the semantic segmentation task, where the inherent higher complexity in terms of network representations needed for pixel-wise classification calls for more advanced solutions. Considerable research effort has been recently directed toward addressing the adaptation problem in semantic segmentation, giving rise to rich literature.

In the remainder of the chapter, we will introduce a formal characterization of the Do-

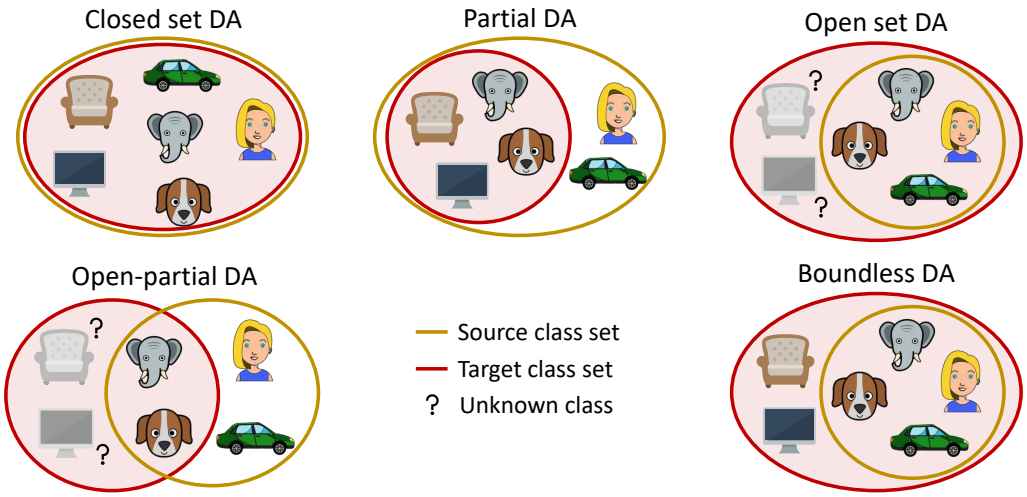


Figure 2.1: Different settings for domain adaptation, according to how source and target class sets are related.

main Adaptation problem (Section 2.2) and provide a comprehensive description of many UDA methods for semantic segmentation present in literature, starting from a coarse categorization based on the representation level of adaptation (Section 2.3), and then moving to a finer classification (Section 2.4). Finally, we will illustrate the most popular experimental setups leveraged to evaluate domain adaptation frameworks (Section 2.5); these setups will come in handy when validation will involve the UDA methods we will detail throughout Part I of the thesis.

2.2 Problem Formulation

Domain Adaptation (DA) is a special case of transfer learning, called Transductive Transfer Learning, in which the source and target tasks coincide ($\mathcal{J}_S = \mathcal{J}_T$), whereas the discrepancy lies in the domain difference ($\mathcal{D}_S \neq \mathcal{D}_T$). In addition, domain adaptation is commonly intended in a homogeneous fashion, where the domain shift happens at a statistical level ($\mathbb{P}(\mathcal{X}_S, \mathcal{Y}_S) \neq \mathbb{P}(\mathcal{X}_T, \mathcal{Y}_T)$) rather than being due to distinct input spaces ($\mathcal{X}_S$ and $\mathcal{X}_T$ belong to the same semantic domain, *e.g.*, urban scene images) [7].

Recently, some research efforts have considered more challenging scenarios than the standard homogeneous DA, allowing for separate sets of semantic classes in the source and target domains ($\mathcal{C}_S$ and $\mathcal{C}_T$). Depending on how the source $\mathcal{C}_S$ and target $\mathcal{C}_T$ sets are related, it is possible to identify multiple DA scenarios (Figure 2.1):

- *Closed Set DA*: it corresponds to the homogeneous case, where semantic classes are completely shared between source and target domains ($\mathcal{C}_S = \mathcal{C}_T$).

Table 2.1: Formal definition of notation used throughout Part I of the dissertation.

Symbol	Definition
$\mathcal{D}_{\{S,T\}}$	Source/Target domain
$\mathcal{X}_{\{S,T\}}$	Source/Target input space
$\mathcal{Y}_{\{S,T\}}$	Source/Target label space
$\mathcal{J}_{\{S,T\}}$	Source/Target task
$\mathcal{C}_{\{S,T\}}$	Source/Target class set
$\mathbf{X}_n^{\{s,t\}}$	Source/Target input sample
$\mathbf{Y}_n^{\{s,t\}}$	Source/Target ground-truth map

- *Partial DA*: in this setup there exist some source classes that do not appear in the target domain ($\mathcal{C}_S \supset \mathcal{C}_T$).

- *Open Set DA*: conversely to partial DA, here the presence of target private classes is admitted, for which no training examples in the source domain are available ($\mathcal{C}_S \subset \mathcal{C}_T$).

- *Open-Partial Set DA*: the source and target domains include separate sets of semantic classes [28], with a subset of those in common ($\mathcal{C}_S \neq \mathcal{C}_T, \mathcal{C}_S \cap \mathcal{C}_T \neq \varnothing$). However, elements belonging to the class subset exclusive to the target domain have only to be acknowledged as not part of the shared classes. This setup has also been defined as *Universal* DA, to signify that it comprises all the DA settings detailed previously as special sub-cases.

- *Boundless DA*: this setup is very similar to the open set one, but objects of target private classes must be explicitly classified rather than only be associated to a general unknown target class. This setting has been recently introduced [29] and it represents the most ambitious one, since it admits complete unawareness beforehand about semantic content of target data.

In the following sections, the focus will be placed on the *standard* most-diffused closed set adaptation, as, up to now, this is by far the most explored setup.

According to the degree of annotations availability in the target domain, the adaptation problem is subject to a further categorization, ranging from the full or partially annotated *supervised* or *semi-supervised* settings to the completely label deprived *unsupervised* scenario. In particular, Unsupervised Domain Adaptation will be investigated, as it has recently witnessed an increase in popularity, especially in relation to the semantic segmentation task, and

it involves many practical applications. More specifically, it is assumed that a set of labeled source data $\{\mathbf{X}_n^s, \mathbf{Y}_n^s\}_n$ drawn accordingly to the source joint distribution over $\mathcal{X}_S \times \mathcal{Y}_S$ is provided, paired with a set of unlabeled samples $\{\mathbf{X}_n^t\}_n$, retrieved from a distinct target marginal distribution over $\mathcal{X}_T$. The objective is to discover a predictive function correctly modeling the task input-label relation in the target domain, while knowledge on the chosen task can be extracted only from source labeled samples. Table 2.1 summarizes the notation introduced so far.

Furthermore, for standard domain adaptation techniques to work, source and target domains should be somehow related, meaning that they should share task-relevant content, while low-level attributes may differ. This scenario is commonly referred to as *one-step* DA, as knowledge transferring happens directly across source and target data without intermediate stages.

2.3 Adaptation Focus

As previously discussed, behind the performance degradation suffered by deep prediction models applied on new target environments lies the covariate shift phenomenon affecting source and target input data samples. For this reason, most of domain adaptation research builds upon bridging the statistical gap between domain distributions, in order for the prediction model to yield satisfactory results whenever those distributions are matched.

Various strategies have been explored to achieve the statistical matching, which will be thoroughly discussed in Section 2.4. A more general categorization of domain adaptation techniques, however, can be inferred, according to where in the employed semantic segmentation model the statistical discrepancy happens to be addressed. In particular, different data representations could be subject to adaptation, from the bare images prior to classification up to intermediate and output network activations (Figure 2.2). In the following, a description of the main ideas behind adaptation approaches will be provided, grouped by where the adaptation effort is focused [233].

2.3.1 Input Level Adaptation

A first strategy is to perform adaptation at the input level, directly on images before they are fed to the segmentation network (as shown in the leftmost part of Figure 2.2). The idea is to force data samples from either domain to reach an uniform visual appearance, meaning that they not only have to carry high-level semantic similarity, but their low-level statistical discrepancy should be matched as well. This because, even if source and target images carry

strong high-level semantic similarity in scene content and layout, inter-domain low-level statistical discrepancy, although mostly lacking semantic significance, is likely to result in an undesirable reduction of the prediction efficacy on target samples. A clear example of this is the synthetic to real adaptation (see Section 2.5.1); although it may be quite realistic, synthetic data can mimic real-word properties up to a certain extent. Thus, it is usually possible to find synthetic peculiar traits, however small, which can undermine the efficacy of a model trained on synthetic data in a real-world environment.

The common approach to address domain adaptation at the input level is to map the data to a new image space, where the projected source (or target) samples carry an enhanced perceptual resemblance to target (or source) ones. This is normally achieved with the help of style-transfer techniques, whose objective is turned into matching source and target marginal distributions in the image space. By feeding in input supervised data from the new domain-invariant space to the segmentation network, the predictor should now able to retain consistent results across domains.

An upside of this approach is its complete independence w.r.t. the segmentation network currently in use, thus inherently missing sufficient discriminative power when it is employed in its vanilla scheme, that is, without any additional regularization constraints. Indeed, alignment of marginal distributions can be fully accomplished, and yet no semantic consistency may be preserved, with class-conditional distributions (not accessible at training time in the unsupervised target domain) still differing across domains. In other words, one might find many domain invariant representations, all lacking semantic discriminativeness to solve the segmentation task in the target domain. This for example could happen when elements of a certain class are mapped to different categories, which may be totally complying with the statistical alignment constraint, while, in fact, disregarding content preservation. To bypass these issues, multiple approaches have been devised to enforce semantic consistency of image translations, for example by resorting to image reconstruction constraints, uniformity of segmentation predictions or ad-hoc engineered techniques to safely manipulate low-level statistics.

2.3.2 Feature Level Adaptation

An alternative approach is to focus the adaptation on feature representations, pursuing a distribution alignment of network latent embeddings, which are normally retrieved from the encoder output in the commonly employed encoder-decoder architecture (even if adaptation at other network stages has also been employed). The primary objective is to build a domain invariant latent space, in which features extracted either from source or target input images observe the same distribution. In the end, learning solely from supervision on source

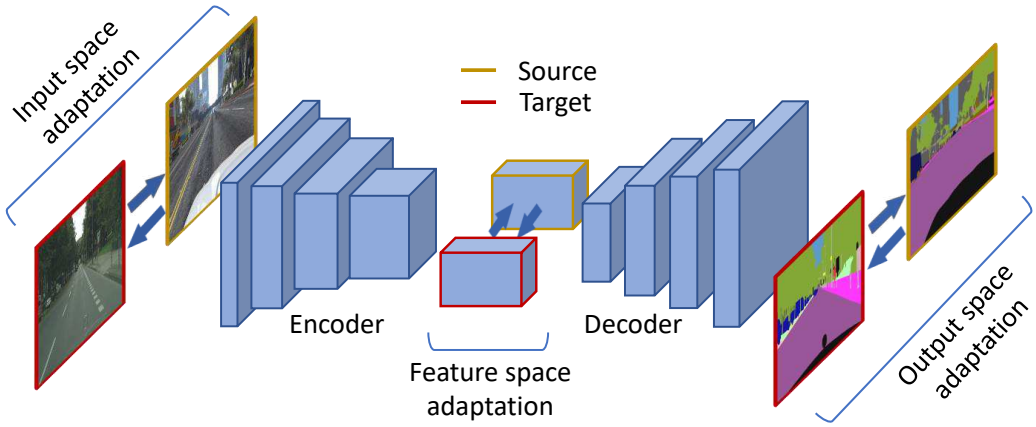


Figure 2.2: Typical encoder-decoder architecture for semantic segmentation, highlighting different network stages on which domain adaptation strategies can be applied, from the input image space up to intermediate and output network activations.

representations should result in a good performance also on the target domain, as shared classification in the adapted latent space should be jointly effective on both source and target representations when distributed alike.

In the context of semantic segmentation, the feature space retains significant complexity due to its high-dimensionality, which is necessary for the prediction model to simultaneously capture global semantic clues, while attaining pixel-level accuracy. Thus, alignment at feature level in its simplest fashion may be insufficient, due to the structural and semantic complexity that feature embeddings possess, which is difficult to fully capture and handle (*e.g.*, by an adversarial discriminator) [30]. In addition, as for the input level adaptation, a semantically unaware alignment of marginal distributions (*e.g.*, standard adversarial adaptation) does not guarantee that the joint input-label distributions are matching, since no information can be derived from unlabeled target samples about the target joint distribution. For these reasons, many feature level adaptation techniques that have been successfully devised for image classification do not easily extend to the dense segmentation task, and in general require careful tuning and further regularization.

2.3.3 Output Level Adaptation

Finally, the last class of domain adaptation techniques exploits a cross-domain distribution alignment over the network output, *i.e.*, typically the output per-class probability space. Not only prediction probability maps have proven to retain sufficient complexity and richness of semantic information, but they also span a low-dimensional space over which statistical align-

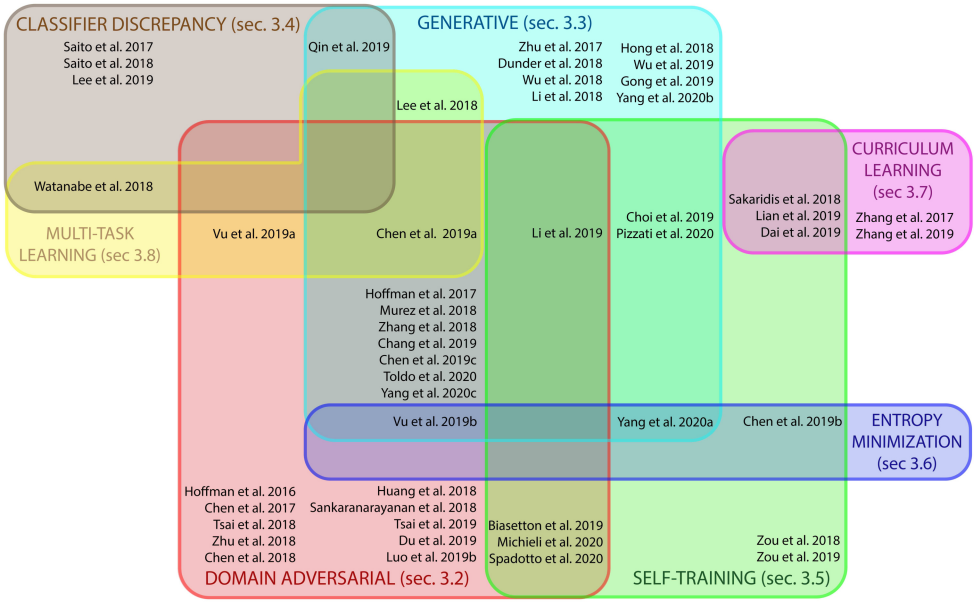


Figure 2.3: Venn diagram of the most popular UDA strategies for semantic segmentation. Each method falls under the set representing the adaptation techniques that are used.

ment happens to be achieved much more effectively, for example by the domain adversarial strategy. In addition, source knowledge can be indirectly translated over the unlabeled target domain by resorting to some form of self-taught supervision extracted from target prediction maps, whose careful introduction in the learning process to support the standard source supervision may result in an effective cross-domain adaptation of the network performance. Source priors derived from label distribution have proven to provide an useful regularization to the learning process as well, since they usually identify high-level semantic properties shared across domains.

2.4 Unsupervised Domain Adaptation Techniques

In the following sections, the most relevant approaches to address the unsupervised domain adaptation problem in *semantic segmentation* will be discussed. For each set of techniques, some works, whose proposed adaptation strategy can be associated to that group, will be presented. Nonetheless, we remark that most of the domain adaptation frameworks recently introduced are likely to make use of a combination of multiple techniques to improve performance.

We group UDA works into 7 main categories as displayed in the graphical overview of Fig-

ure 2.3. Let us briefly introduce each of them, leaving more in-depth details to the following sections.

(i) *Domain adversarial* techniques (Section 2.4.1) learn to produce data with a statistical distribution similar to the one of (source) training samples via adversarial learning schemes.

(ii) *Generative-based* solutions (Section 2.4.2) typically use generative networks to translate data between domains, in order to bridge the statistical gap in the image (input) space, while retaining the original source supervision in the new artificial domain-independent image space.

(iii) *Classifier discrepancy* methods (Section 2.4.3) resort to multiple dense classifiers on top of a single encoder to capture less adapted target representations and, in turn, encourage an improved alignment of cross-domain features far from decision boundaries via an adversarial-like strategy.

(iv) The *Self-training* paradigm (Section 2.4.4) propose to obtain some form of pseudo-labeling (typically using some confidence estimation schemes to select the most reliable predictions) to automatically drive the learning process, *i.e.*, via self-supervision in the target domain.

(v) *Entropy minimization* (Section 2.4.4) aim at minimizing the entropy of target output probability maps to mimic the over-confident behavior of source predictions, thus promoting well-clustered target feature representations. Similar in spirit to self-training, also here the learning process in the target domain is self-supervised.

(vi) *Curriculum learning* strategies (Section 2.4.5) tackle one or more easy tasks first, in order to infer some necessary properties about the target domain (*e.g.*, global label distributions) and then train the segmentation network such that the predictions in the target domain follow those inferred properties.

(vii) *Multi-task* frameworks (Section 2.4.6) solve multiple tasks simultaneously to promote learning of invariant feature representations.

2.4.1 Domain Adversarial Adaptation

Adversarial learning has been originally introduced for image generation [31]. The main objective behind the generative task is to model an unknown probability distribution of data from a training set. The adversarial strategy has proven extremely effective in solving this type of problems, since no explicit expression of the underlying target data distribution has to be found, and, more importantly, no specific learning objective to train the generative model is required. The learning process builds upon a min-max game, in which a generator network is progressively guided by a discriminator network to produce realistic samples. In [31], the discriminator is a binary classifier whose goal is to discern between the original

training data and the data produced by the generator. The generator is instead a generative model that takes in input random noise (or some conditioning data in more recent variations of the approach) and yields data (*i.e.*, images in the setting of interest) resembling the ones in the training set. The generative action is constantly pushed to improve the realism of its output samples to fool the discriminative action of its opponent; this is achieved by using a loss function whose minimization in turn maximizes the errors of the discriminator. The model is trained by alternating a discriminator training step, aiming at maximizing its accuracy, and a generator optimization phase with the opposite target (see Figure 2.5). If correctly carried out, the adversarial competition should result in a statistical distribution of generated data that fully matches the training set one, meaning that original and generated data should be statistically indistinguishable. In addition, the discriminator purpose is to both capture and express a measure of statistical discrepancy in the form of a structured learnable loss. Therefore, the objective function can be thought as being jointly learned and optimized in the adversarial process, allowing it to adapt to the specific context. For this reason, the objective function can be thought as being jointly learned and optimized in the adversarial process, removing, in fact, the necessity to manually design complex losses suitable for the specific context. Therefore, the adversarial learning scheme introduced in [31] can be extended under careful adjustments to address multiple tasks that would normally require different types of application-specific objectives.

Feature Adversarial Adaptation Adversarial learning has been successfully extended to the domain adaptation task [32–34]. The real-fake discriminator is now turned into a domain classifier that is used to drive the adaptation process. Its discriminative action is, in fact, focused on capturing the statistical discrepancy between representations from separate domains, which is responsible for the performance degradation and thus has to be reduced in order to achieve an effective adaptation.

There are two possible targets for the domain classifier. The first is to discriminate between internal or output representations extracted from data in either source or target domains (Figure 2.4). This allows to introduce additional loss terms enforcing the construction of feature or output spaces that are more domain invariant. Alternatively, it is possible to use the discriminator to distinguish between the output of the network (that can correspond both to inputs from the source and from the target domain) and the ground-truth segmentations (that in the unsupervised setting are only present in the source domain). Since in the adversarial model there is no need to have ground-truth data matching the provided samples, this allows to use also the target domain images for which no ground truth is available and to enforce that their predicted segmentation maps have statistical properties similar to the ground-truth ones (Figure 2.6). Using these strategies, the standard supervision from the an-

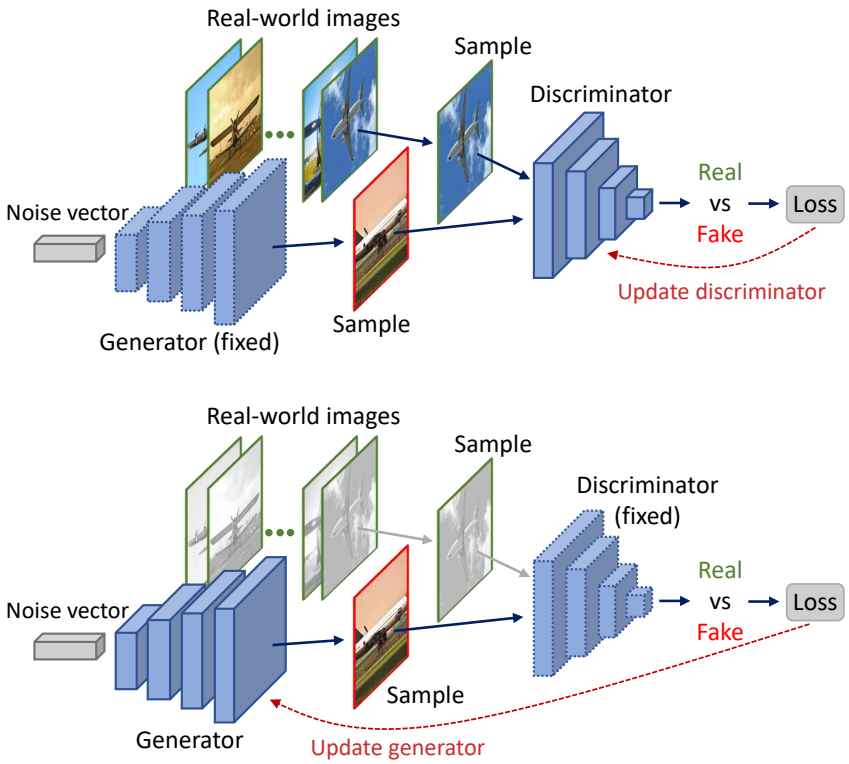


Figure 2.4: Training of a generative adversarial network. The upper part displays the discriminator's update phase, while in the lower part the generator's update step is shown.

notated source data is joined by a supervisory signal from the domain discriminator, which pushes the prediction network towards domain invariance, in turn mitigating the intrinsic bias towards the supervised source domain. In other words, a measure of domain discrepancy is simultaneously learned and minimized within the adversarial competition

Following the success of adversarial domain adaptation for image classification [32–34], the adversarial strategy has been introduced also in the context of semantic segmentation to achieve domain alignment over latent feature embeddings [35]. Hoffman *et al.* [35] have been the first to address domain adaptation in semantic segmentation, and they resort to an adversarial approach. I n particular, they devise a global domain adversarial alignment, based on a domain discriminator taking as input the feature representations from intermediate activations of the fully convolutional segmentation network. In addition, they propose a category specific distribution alignment, which is accomplished by enforcing image-level label distribution constraints on target predictions inferred from source annotations, under the assumption that high-level scene layout is in general shared among source and target images.

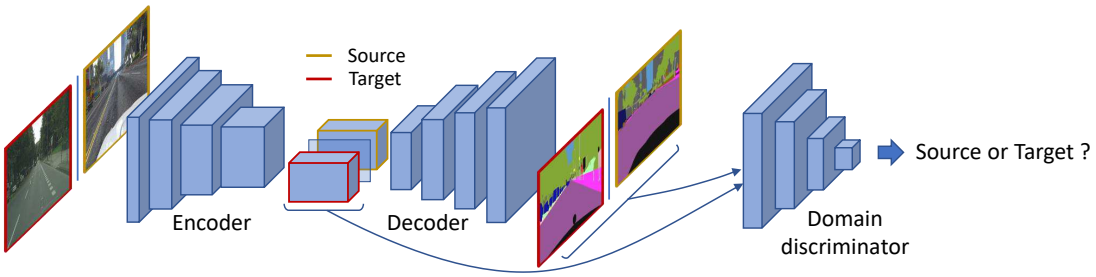


Figure 2.5: Graphical representation of the standard adversarial adaptation strategy. A domain discrimination captures the statistical discrepancy between source and target representations (*e.g.*, segmentation network's output or features maps computed from one or the other domain). Its supervisory signal is then exploited to perform domain alignment.

Following a similar approach to [35], many works have further resorted to adversarial alignment of network latent embeddings [36–44, 234]. As previously discussed, the domain discriminator is able to infer a structured loss to capture global distribution mismatch of cross-domain image representations. Yet, the global domain alignment of marginal distributions provided by the vanilla domain adversarial scheme may end up in incorrect semantic knowledge transfer across domains, with class-conditional distributions neglected in the learning process. For this reason, to reach an effective adversarial adaptation when dealing with the semantic segmentation task, additional modules should be embedded in the adaptation pipeline.

Chen *et al.* [36] use an additional target guided distillation loss by matching network activations from target inputs during the training phase with those from a pre-trained version on the ImageNet dataset [26]. Moreover, the feature adversarial adaptation is enforced independently over different spatial regions of the input image, thus exploiting the underlying spatial structure of input scenes. Zhang *et al.* [37], instead, boost the feature-level adaptation performance by providing the domain discriminator with an Atrous Spatial Pyramid Pooling (ASPP) module [11] to capture multi-scale representations. Furthermore, a possible solution is to integrate adversarial feature alignment in a generative approach (see Section 2.4.2), as done in several works [38, 40, 41, 234]. Here the goal is to match source and target marginal distributions in the input image space by a source-to-target image-to-image translation function, and then cross-domain latent representations are further brought closer by matching source original and target-like source embeddings in a domain adversarial fashion. An alternative is to perform category-wise adaptation adaptation, [43, 44] revisit the original approach of Hoffman *et al.* [35] by assisting the global distribution alignment with class-wise adversarial learning. Chen *et al.* [43] propose to exploit multiple feature discriminators (one for each class), so that negative transfer among different classes in the domain bridging process should be effectively avoided. Du *et al.* [44] propose a similar class-wise adversar-

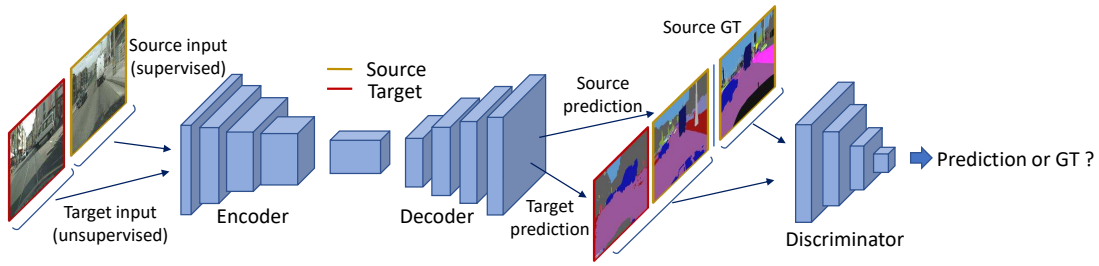


Figure 2.6: Graphical representation of an output adversarial adaptation strategy, where domain alignment is performed indirectly by bridging the distribution gap between source annotation maps and network predictions from either source or target domains.

ial technique, which is improved by imposing independence during the optimization of the multiple discriminators. They argue that soft labels lead to incorrect adaptation on class boundaries, where different class discriminators may provide their guidance simultaneously. Finally, they devise an additional module to adaptively re-weight the contribution of each class component in the adversarial loss, in order to avoid the inherent dominance of classes with higher prediction probability, which turns out to be more easily well-adapted across the domains.

Different from the aforementioned techniques, other works [45–47] seek for domain alignment inside the feature space by applying a reconstruction constraint to ensure that latent embeddings possess enough information to recover the input images from which they have been extracted. To this end, adversarial learning is applied on the reconstruction image-level space. To achieve cross-domain feature distribution alignment, the feature extractor is trained to yield latent representations that can be projected back to both source and target image spaces indistinctly. In these frameworks the backbone encoder of the segmentation network plays a min-max game against the domain discriminator. The encoder, indeed, tries to fool the discriminator on the actual originating feature's domain, by looking at the corresponding reconstructed images projected back into the image space. In other words, the objective is to learn source (target) features that can successfully generate target-like (source-like) images to promote domain invariance of those representations.

Output Adversarial Adaptation As previously observed, the semantic segmentation task entails a quite complex feature space, due to the high dimensionality of its representations. Thus, to bypass the complexity encompassed in feature space adaptation, a research direction has been to focus the adaptation effort to the segmentation output space [48–54, 238]. The low-dimensional output representations, in fact, have been shown to retain enough semantic information for a successful adaptation. In this new output level adversarial scheme,

a domain discriminator learns to discover the domain from which segmentation maps are originated. Simultaneously, the segmentation network plays the generative role by providing cross-domain statistically close predictions to fool the discerning action of the domain classifier.

Tsai *et al.* [48] are the first to propose this type of adaptation: in order to improve the signal flow from the adversarial competition through the segmentation network, they deploy multiple dense classification modules at different depths upon which as many output-level discriminators are applied. Following the technique proposed in [48], other works adopt the output space adversarial adaptation in combination with additional modules. Chen *et al.* [49] combine semantic segmentation and depth estimation to boost the adaptation performance. In particular, they provide the domain discriminator with segmentation and depth prediction maps jointly, in order to fully exploit the strong correlation between the two visual tasks. Luo *et al.* [51] enhances the adversarial scheme by a co-training strategy that highlights regions of the input image with high prediction confidence. In this way, the adversarial loss can be effectively tuned by balancing the contribution of each spatial unit, so that more focus is directed towards less adapted areas.
Other works [53, 54, 238] revisit the adversarial output-level approach. In particular, they utilize a discriminator network which has to distinguish between source ground-truth maps and generated semantic predictions from either source and target data. In doing so, the cross-domain statistical alignment is not directly performed, but forcing the segmentation network output to be distributed as ground-truth labels for both source and target inputs leads to an indirect yet effective alignment between the two domains.

Recently, new approaches [55–57] have been built upon the extraction of meaningful patterns from the segmentation output space. The intention is to provide the domain discriminator with a more functional and significant understanding of source and target representations, allowing it to yield a more effective guidance in the adaptation process. On this regard, Vu *et al.* [55, 56] devise an entropy-minimization strategy (which will be described more in detail in Section 2.4.4) to promote more confident target predictions. They propose an indirect approach relying on the adversarial alignment of the statistics of self-information maps computed on top of source and target predictions. In particular, a domain discriminator has to detect whether a weighted self-information map comes from a source or a target prediction, whereas the segmentation network, trying to deceive the discriminator, is forced to produce low-entropy target maps as to mimic source confident ones. This process effectively pushes decision boundaries away from high-density regions in the representation space.
With a different approach, Tsai *et al.* [57] construct a clustered space over the output prediction space by adding a patch clustering module that discovers patch-wise modes on segmentation maps. First, the module is supervisedly trained on source data by leveraging the available

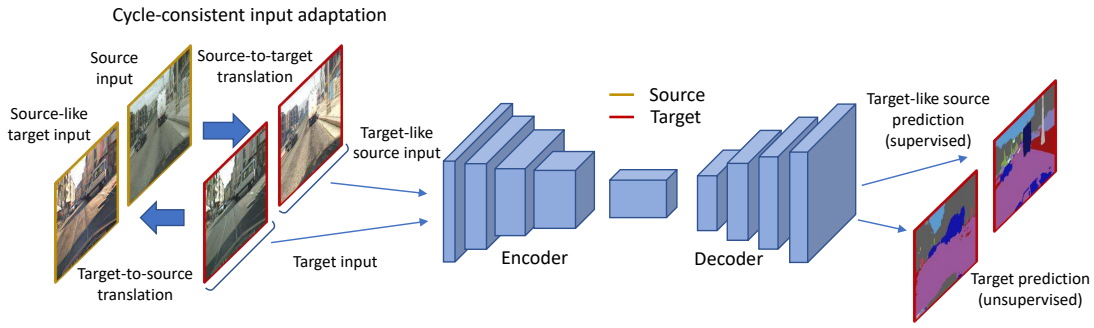


Figure 2.7: Overview of the generative-based adaptation approach built upon cycle-consistent image-to-image translation. Source translated input images are exploited as a form of target-like artificial supervision during the learning process.

annotations, then it is exploited to achieve a patch-wise distribution alignment by enforcing adversarial cross-domain adaptation between its clustered source and target representations. The idea behind this approach is to capture high-level structured patterns, that are essential to solve the semantic segmentation task, to be provided to the domain discriminator for an improved domain statistical alignment. Thus, the achieved domain uniformity on a patch-level should ensure, in principle, that the segmentation task can be effectively solved also in the target domain.

2.4.2 Generative-Based Adaptation

Unsupervised image-to-image translation is a class of generative techniques where the objective is to learn a suitable function to project images from one domain to another, relying solely on the supervision provided by unpaired training data sampled from the considered domains. The idea is to extract visual characteristics peculiar to a specific set of images and transfer those properties to a different data collection. In a more formal definition, the image-to-image translation task aims at discovering a joint distribution of images from different domains. Notice that, since the problem is, in fact, ill-posed, as an infinite set of joint distributions can be inferred from the marginal ones, appropriate constraints must be applied to obtain acceptable solutions.

The image-to-image translation task can be effectively exploited in domain adaptation. What could be achieved is, in fact, the transferring of visual attributes from the target domain to the source one, while preserving source semantic information. By doing so, the covariate shift phenomena responsible for the classifier performance drop is alleviated. In this direction, several works have resorted to an input-level adaptation strategy based on image translation between source and target domains. What all these works have in common,

is the search for domain invariance of visual appearance of images from different domains. This ultimately allows to exploit target-like supervision from translated, yet still annotated, source images.

Cycle Consistent Translation A common approach shared by a multitude of works [38, 40, 41, 46, 58–62, 234] is to exploit the successful CycleGAN [63] model to perform unsupervised image-to-image translation (Figure 2.7). The framework proposed by Zhu *et al.* [63] concurrently learns in adversarial manner the conditional image translations in both the source-to-target and target-to-source directions. The two adversarial generative modules are further tied by a cycle-consistency constraint, driving each of them to learn the inverse projection of the other. The reconstruction requirement is essential to preserve structural geometrical properties of the input scene, but provides no guarantees about the semantic consistency of translations. In fact, while retaining geometrical coherence, the mapping functions could completely disrupt the semantic content of input data.

To tackle this issue, the lack of semantic consistency in the vanilla translation scheme is counteracted by taking advantage of the semantic prediction capability of the segmentation network [38, 40, 41, 58, 234]. In particular, the semantic predictor can be exploited to detect and thus discourage any perturbation of the semantic output which may happen during the translation provided by the CycleGAN's generators. This is generally done by enforcing consistent prediction maps over original and translated versions of the same image. Still, being the prediction maps intrinsically flawed, especially in the target domain where annotations are missing, the inaccurate semantic information provided to the generative module could hurt the learning of the image projections. Thus, some works propose to simultaneously optimize the generative and discriminative framework components in a single stage [234], or even split the segmentation network into separate source and target predictors [41]. Li *et al.* [38] further extend the CycleGAN-based adaptation strategy formulating a bidirectional learning framework. The image-to-image translation and segmentation modules are alternately trained, in an optimization scheme by which each module is provided with positive feedback from the other. The segmentation network benefits from the target-like translated source images with original supervision, while the generative network is aided by the predictor in retaining semantic consistency. This closed-loop structure effectively allows for a progressive adaptation, with both image-to-image translations quality and semantic prediction accuracy gradually enhanced.
An alternative solution could be, instead, to achieve semantic awareness in the image-to-image translation based adaptation by acting directly over the adversarial translation modules [60, 61]. Li *et al.* [60] propose to assist the cycle-consistent image-to-image translation framework by a soft gradient-sensitive loss to preserve semantic content in the cross-domain

projection focusing on semantic boundaries. The idea behind this approach is that, no matter how low-level visual attributes change between domains, the edges defining semantic uniform regions should be easily detectable, regardless the distribution the image is drawn from. Thus, a gradient-based edge detector should discover consistent edge maps between original images and their transformed versions. In addition, following the intuition that semantically different regions of an image should face a different adaptation, they devise a semantic-aware discriminator structure. In doing so, the discriminator can semantically-wise evaluate resemblance between original and translated samples.
Recently, Yang *et al.* [61] introduce a phase consistency constraint to the CycleGAN pixel-level adaptation module, observing that the semantic content of an image is mostly encoded in the phase of its Fourier transform, whereas alterations of the amplitude to the representation in frequency does not change its composition.
With a different adaptation perspective, Gong *et al.* [62] adapt the CycleGAN model to generate a continuous flow of domains ranging from source to target ones, by conditioning the generative networks with a continuous variable representing the domain. The reason behind the retrieval of intermediate domains spanning between the two original ones is to ease the adaptation task, by progressively characterizing the domain shift affecting the input data distributions. Moreover, they suggest that resorting to target-like training data from diverse target-like domain distributions improves the generalization capability of the segmentation network.

One-way Translation To reduce the computational burden of the bi-directional structure of CycleGAN (which entails a total of at least four neural networks to be added to the semantic predictor) other works [49, 64–66] discard the backward source-to-target projection branch, seeking for a more light-weight input-level adaptation module, still based on generative adversarial framework. The translation consistency is granted, for example, by the correlation to a related task (*e.g.*, depth estimation) [49, 64], which is jointly addressed with the semantic segmentation. Choi *et al.* [65], instead, improve the generator of the original GAN framework with feature normalization modules at multiple depths to provide style information to source representations, whereas source content is preserved. Furthermore, a semantic consistency loss from a pre-trained segmentation network promotes coherence of image translations, providing, in fact, a regularizing effect in absence of the cycle-consistency one. Hong *et al.* [66] use a conditional generative function to model the residual representation between source and target feature maps, which is optimized in an adversarial framework. In doing so, they avoid any reliance on a shared domain-invariant latent space assumption, which may be not satisfied due to the highly structured nature of semantic segmentation. The generator takes as input low-level source feature maps, together with a noise sample, and

is encouraged to produce high-level feature maps with target-like distribution by a discriminator, that expresses a measure of statistical distance between original and reproduced target representations. Both source original and domain-transformed representations are provided to a dense classifier to compute the cross-entropy loss.

In order to lessen the bias towards the source domain, Yang *et al.* [52] resort to the target-to-source image-to-image translation, in place of the more common source-to-target one, generally employed to generate a form or target supervision from source translated data. The source-like target images are then employed in the supervised training of the predictor thanks to pseudo-labeling. In addition, training the segmentation network directly in the source domain allows to fully exploit the original source annotations, avoiding the risk of semantic alterations which may happen in the source-to-target pixel-level adaptation scenario. Moreover, to align feature representations between domains, they introduce a label-driven reconstruction network. However, differently from the feature-based reconstruction techniques [45–47] (Section 2.4.1), the generative recreation of input images is performed starting from semantic maps from the segmentation output. In doing so, they seek to guide a category-wise alignment of the segmentation network embeddings, since reconstructions that deviate from their target are penalized, thus providing semantic consistency to network predictions.

Style-Content Disentanglement As an alternative to the CycleGAN-based adaptation, style transfer techniques have also been explored to achieve domain invariance of low-level image attributes. Behind these methods lies the principle that any image can be disentangled into its content and style representations. While the style of an image is related to low-level domain-specific traits, its content indicates domain-invariant high-level semantic properties. Therefore, joining source content with target style should provide target-distributed training data, still preserving source semantic annotations. Once more, target supervised training can be performed thanks to the new generated target supervision. A common approach for style transfer is to resort to content and style decomposition in the latent space [50, 67]. Then, source to target translation becomes combining the extracted source content representations to random target style ones, with the mixed representations to be re-projected into the image space. In a recent work [67], the authors perform multi-modal source-to-target image translation based on the MUNIT architecture [68]. The original datasets are augmented with additional web-crawled data, in order to reduce the gap in terms of task-unrelated data properties between sets, while at the same time highlighting the relevant task-related visual features to be matched. Furthermore, the style transfer method allows for multi-modal translation, therefore multiple target styles can be transferred to a single source image, thus increasing training data diversity and, in turn, enforcing the adaptation robustness To avoid the complexity involved in GAN's image generation (specially high resolution images are challenging

to be obtained), different types of style transfer techniques have also been explored, ranging from neural or photo-realistic style transfer [37, 69] to feature re-normalization [65, 70] and low-level frequency spectrum manipulation [30].

Zhang *et al.* [37] adopt traditional techniques of neural style transfer [71, 72] to separate style (low-level feature) from image content (high-level features). In particular, multi-level response maps of a pre-trained CNN are exploited for image synthesis, where image style is expressed by the correlation between feature maps in the form of Gram matrices. Differently, Dundar *et al.* [69] make use of a photo-realistic style transfer algorithm for an iterative optimization by which both the segmentation network and the translation algorithm performances are constantly improved. Alternative approaches [65, 70] opt for the re-normalization of source feature maps, so that their first and second order statistics match those of the target ones, by means of the AdaIN module [73]. Finally, Yang *et al.* [30] remove domain-dependent visual attributes from source images by replacing the low-level frequency spectrum with that of target images, without affecting high-level semantic interpretability. They argue that this simple approach, despite not requiring any additional learnable module, results in a remarkably robust adaptation performance when embedded in a multi-band framework that averages predictions with different degrees of spectral alteration.

2.4.3 Classifier Discrepancy

As mentioned in Section 2.4.1, feature level adversarial adaptation in its standard design involves an additional domain classifier, whose discriminative action over feature representations from source and target domains provides an effective supervisory signal to the learning process, pushing the segmentation network towards domain invariance within the latent space it spans. A separate task-specific objective is instead responsible for the prediction network to learn the actual task with source supervision, *i.e.*, the standard cross-entropy loss for semantic segmentation.

Despite being fairly effective, the standard adversarial adaptation lacks semantic awareness [74, 75]. A proper adversarial alignment, in fact, entails a match of marginal distributions of source and target data, which is typically not followed by a class-conditional statistical alignment as well. This because category-level joint distributions necessarily remain unknown to the domain classifier, as complete lack of supervision in the target domain implies that no information about semantic content of target data is available. The result is that features may be moved near class boundaries, where classification uncertainty could lead to incorrect predictions. Even worse, a class-agnostic transfer of target features might incorrectly align them to source representations of a different semantic class in the domain invariant latent space, which means negative transfer has been introduced.

Aiming at overcoming those issues, Saito *et al.* [74] completely re-design the original domain adversarial approach, proposing the Adversarial Dropout Regularization (ADR) mechanism to reach cross-domain feature alignment away from decision boundaries. In particular, they provide the task-specific dense classifier (*i.e.*, the encoder network) with the discriminative role that was before assigned to an external domain discriminator. By perturbing the classifier using dropout, it is possible to detect where predictions are more uncertain, which happens to be strongly related to the distance of feature representations from decision boundaries. In this novel adversarial scheme, the dense classifier is trained to improve its sensitivity to semantic variations on target representations. By acting against it, the feature extractor (*i.e.*, encoder) aims at providing categorical certainty to the target features it computes. This should effectively remove task-unrelated information enclosed in target representations, which is responsible for highly variable predictions, ultimately pushing them far from decision boundaries.

The ADR approach originally proposed in [74] has been revisited by several works [51, 59, 75–77]. A downside of the classifier discrepancy strategy in its primary scheme is the inherent noise sensitivity acquired by the decoder [74], which is crucial for it to capture the proximity of target samples to the classification boundaries, yet it negatively affects the accuracy of the whole segmentation network, requiring, in fact, an extra training stage to correctly learn the segmentation task. On this regard, the original scheme could be improved by replacing the dropout strategy to retrieve multiple predictions over the same feature representation with a couple of distinct decoders, which are simultaneously trained to provide correct, yet distinct, dense classifications [75]. This should effectively improve the accuracy of the decoder section of the prediction model, at the cost of an extra module to be learned within the training process.

Following a different path, the original adversarial learning scheme could be modified, opting for a non-stochastic virtual dropout mechanism to discover minimum distance dropout masks causing maximum prediction divergence [77]. By doing so, the original dropout-based solution to get distinct predictions from a single classifier is retained, while the aforementioned noise susceptibility problem is simultaneously solved.

The co-training principle based on multiple predictors to estimate the current adaptation performance has further been investigated [51]. In particular, the detection of inconsistent predictions can be exploited to focus the discriminator effort (now in a standard domain adversarial framework) towards less adapted regions of the input image, *i.e.*, those affected by the highest uncertainty. This has proved to ultimately lead towards a more effective domain alignment of source and target representations.

2.4.4 Self-Supervised Learning

Due to the similarity between the two tasks, multiple techniques for unsupervised domain adaptation have been borrowed from the semi-supervised learning (SSL) field. Indeed, UDA can be thought as an extension of the SSL problem, since in both cases part of the training data is unlabeled. However, the original lack of annotations of SSL within the unlabeled training set is joined by a statistical shift in UDA between source and target data, which demands for an extra effort to be addressed.

Self-Training In this direction, a first class of adaptation techniques [1, 30, 38, 53, 54, 58, 65, 78, 79, 238] have resorted to self-training. This approach, commonly employed in semi-supervised learning [80], revolves around the creation of pseudo-labels from highly confident network predictions inferred on unlabeled target data. A form of self-taught supervision is therefore available on the target domain, to be exploited in conjunction with the standard supervision from source labeled data.

As opposed to other adaptation approaches described in previous sections, such as the most successful adversarial ones, feature-level cross-domain alignment is implicitly pursued through target self-supervised learning, as source supervision is indirectly transferred to the target domain by pseudo-labels. Self-training pushes network probability outputs to reach a peaked distribution, which translates into predictions displaying a more confident behavior. A key issue, however, lies in the self-referential nature of this technique, which could lead to catastrophic error propagation if not properly handled. Over-confident incorrect predictions on uncertain pixels, in fact, may result undetected, since any form of supervision on unlabeled target data is missing. In turn, those prediction mistakes could be reinforced by the self-teaching strategy, causing a progressive deviation from the correct solution. To cope with this issue, most of self-training based adaptation approaches apply some filtering strategies to the pseudo-labeling process, so that prediction errors inherently affecting target segmentation maps are largely discarded.

A common approach towards a self-training based adaptation involves offline techniques for pseudo-label computation [38, 78, 79], with the confidence threshold updated multiple times during the adaptation process by looking at the whole available training set. In particular, an iterative self-training optimization procedure is followed, which alternates steps of task supervised learning on both source original and target artificial annotations and pseudo-labeling to generate a self-taught target supervision. Thus, during an entire training stage, artificial target annotations are kept fixed. This offline strategy allows for a stable learning process, at the price of the extra computational burden due to multiple steps of pseudo-annotation on the whole target dataset.

On this regard, Zou *et al.* [78] propose one of the first UDA techniques based on self-training. They devise an iterative self-training optimization scheme, between segmentation network training and target pseudo-label estimation. In particular, the target pseudo-labels are treated as discrete latent variables to be computed through the minimization of a unified training objective. In addition, motivated by the fact that class-unaware pseudo-labels confidence filtering is intrinsically biased towards the easy (*i.e.*, more confident) classes, they devise a class-balancing strategy by setting category-wise confidence thresholds. This should promote inter-class balance, as the same amount of top confident pixels are considered for each class, thus resulting in class-wise uniform contributions to the learning process. Finally, since source and target domains are supposed to share high-level scene layout, they also utilize spatial priors from source label statistics, which are inferred for each semantic category and incorporated in the training objective. Furthermore, Zou *et al.* [79] revisit their previous work in [78] by extending the pseudo-label space from one-hot maps to a continuous space defined by a probability simplex. In this way, by avoiding clear-cut overconfident self-supervision in the whole input image, the effect of the inherent misleading incorrect pixel predictions should be effectively reduced. A continuous pseudo-label space further allows them to introduce a confidence regularizing term in the training objective targeting both pseudo-label (treated as latent variables) and network weights, with the purpose of achieving output smoothness in place of sparse segmentation maps.

In order to avoid slow offline dataset-wise processing, Pizzati *et al.* [67] introduce self-training with weighted pseudo-labels. A learnable confidence threshold is employed for both pseudo-label refinement and weighting, thus making pseudo-labels belonging to a continuous space, while concurrently balancing the impact of uncertain pixels. Target weighted self-generated labels are computed over a single batch, but still retaining a global view, since the confidence threshold is learned throughout the entire training phase.

In a different direction, the self-training strategy could be tied to an output level adversarial adaptation [53, 54, 238]. In particular, the output map from a fully convolutional output-level domain discriminator can be regarded as an accurate measure of prediction reliability on target data, thus providing useful information to refine target pseudo-labels. Thus, the quality of artificial annotations is progressively improved throughout the training process being computed over single batches of target images rather than on the whole dataset, leading to an overall rather effective adaptation.
Michieli *et al.* [54] further improve the pseudo-label selection mechanisms by a region growing strategy. Moreover, Spadotto *et al.* [238] propose to adopt a class-wise adaptive thresholding approach. They select the same fraction of highly confident target pixels for each semantic class, by looking at the batch-wise distribution of the discriminator probability output. In doing so, they provide the adaptation framework with both inter-class confidence

flexibility and time adaptability over the training phase.

As an alternative, pseudo-label reliability can be enhanced by resorting to a form of prediction ensembling [1, 30, 58, 65]. For example, it is possible to exploit an additional network to produce self-guidance over the unlabeled samples [58, 65]. This is done by introducing a teacher network in addition to the original one, which plays the role of a student. Then, the teacher model, whose weights are averaged over the student ones from past training steps, is exploited to guide the learning process of the student network, by yielding target predictions the student network is compelled to emulate. The supervisory action provided by the teacher network leads to more accurate target predictions, on top of which less noisy pseudo-labeling can be performed. The result is a more effective self-training adaptation.
Following different research directions, Chen *et al.* [1] enhance the adaptation of low-level features by introducing an additional ASPP dense classification module. Hence, self-produced guidance in the form of pseudo-labels from the combined knowledge of low and high level target predictions is exploited as additional training objective. Yang *et al.* [30], instead, train multiple instances of the segmentation network with multi-band spectrum adaptation to obtain distinct semantic predictors. Then, target pseudo-labels are generated from the mean prediction of the different segmenter instances, resulting in a more robust adaptation when dealing with multiple rounds of self-training.

Entropy Minimization Adopted from the semi-supervised learning field as well, entropy minimization has been recently introduced to UDA [55]. The intuition behind this approach is that source predictions are more inclined to show a confident behavior, which is revealed by a low entropy level in probability outputs. Conversely, the segmentation output maps from target inputs are likely to display more uncertainty (high entropy), with the noise pattern widely spread and not just limited to regions close to semantic boundaries. Thus, by mirroring the over-confident source behavior in the uncertain target domain, the segmentation network should, in principle, bridge the performance gap that exists between domains. More precisely, the effect of entropy minimization is to avoid classification boundaries in the latent space crossing high density regions, while, at the same time, target representations are well clustered far from those decision boundaries.

The original entropy minimization strategy [55] works at the pixel-level, with each single spatial unit independently contributing to the overall objective. However, to overcome some inherent limitations of this approach, further arrangements have been introduced [1, 30, 55]. A possible solution is to pursue a global distribution alignment of entropy behavior by means of a domain adversarial approach, where the domain discriminator is provided with entropy maps rather than directly with output probability maps as in the standard scheme 2.4.1 [55]. By doing so, structural information enclosed in entropy maps is leveraged, thus leading to

a more effectively domain statistical adaptation. Class-wise priors on label distributions inferred from source annotations are further enforced on target predictions to avoid class imbalance towards easy classes. In addition, it should be remarked that the entropy minimization objective in its original form [55] leads to rapidly exploding gradients when moving from high to low uncertainty regions, which could seriously hinder the learning process. On this regard, a solution could be to modify the standard objective, for example, with a quadratic loss with analogous purposes to the original one, but improved gradient signal properties [1]. This strategy, together with category-wise weighting factors to balance the contribution of different semantic classes, has proved to greatly enhance adaptation process. More recently, Yang *et al.* [30] propose an entropy minimization technique as an additional module to their adaptation scheme. The intent is to seek a regularization effect over the training on unlabeled target data, accomplished by pushing the decision boundaries away from high-density regions in the target latent space, with basically no overhead to the actual framework. The strength of the approach is enhanced by the combined application of other adaptation modules to achieve domain alignment. This, in fact, shifts the UDA task towards SSL, thus making entropy minimization more effective. Moreover, to avoid excessive emphasis on low entropy predictions, they adopt a penalty function that increase the focus on less-adapted high entropy regions of target images.

Finally, entropy minimization has been used together with feature space shaping techniques in a couple of recent works [239, 240]. [239], besides using entropy minimization, forces internal feature representations to be clustered, sparse and orthogonal (if belonging to different classes) in both source and target domains to improve feature-level adaptation. Another recent work [240] further introduces a norm alignment constraint to aid a class-wise feature orthogonality objective in promoting disjoint sets of active feature channels between distinct semantic categories, while driving target embeddings towards the highly confident (*i.e.*, associated with high values of feature norm) source distribution.

2.4.5 Curriculum Learning

Another research area involves curriculum learning, where some easy tasks are solved first (*e.g.*, image classification), inferring some important and useful properties related to the target domain. Then, the acquired information is used to support the training on more challenging task (*e.g.*, semantic segmentation). This family of approaches shares many similarities in spirit with self-training. The main difference between the two approaches lies in the content of the pseudo-labels. While in the self-training approaches the pseudo-label is an estimate of the desired annotation on the target set and it is used as such during training, in curriculum approaches the pseudo-label is represented by some inferred statistical proper-

ties of the target domain (different from the labels for the task) and the network is trained to reproduce such inferred properties in the target predictions.

The first work of this family is [81] and its extension [82], where a couple of easy tasks that are less sensitive to domain discrepancy are solved; namely, the label distribution over global images and the label distribution over local landmark superpixels. The former property is evaluated in the source domain, as the number of pixels in the labels associated to each category, normalized by the total number of pixels. On the other hand, target labels are not available in unsupervised domain adaptation and consequently a machine learning model should be trained on the source domain to estimate them. In the papers, it is argued that this task can be solved more easily than image segmentation and that the results can be used to guide the adaptation of the segmentation task. To estimate the first property on the target domain a logistic regression model is employed. While the first property is useful to guarantee that the ratio among different categories matches the ones of the target domain, samples with semantic maps not following the estimated label distribution on the target domain are still penalized. To solve this problem, a second clue is introduced. Images are divided into superpixels and an SVM classifier is used to select the most representative anchor superpixels and the label distribution is estimated over them. The final objective is a mixture of the pixel-wise cross-entropy of the source samples and the cross-entropy on the two properties on the target domain discussed before. In [83] and [84] the focus is posed on the domain adaptation from clear weather to dense fog images. A novel method, called Curriculum Model Adaptation (CMAda), is proposed to gradually adapt the model to segment images with incrementally growing amount of fog. The algorithm presented starts from a source domain of clear weather images and progress through intermediate target domains of incrementally denser fog, and, finally, reaches the target domain of dense fog images. The segmentation model, initially, is pretrained with supervision on the source domain and then, with as many adaptation training steps as the number of denser fog steps, it is gradually shifted towards the target domain. In [85] the connection between curriculum learning and self-training is highlighted and a method (called self-motivated pyramid curriculum domain adaptation, PyCDA) that uses and merges both techniques is presented. The authors remind that in self-training there are two main training steps that alternates: (1) the evaluation of pseudo-labels for the target domain and (2) the supervised training of the segmentation network with the labeled source domain images and with the target domain images with pseudo-labels. In curriculum learning there are also two steps that alternates: (1) the inferring of properties of the target domain (e.g., frequency label distributions over global images or image regions, like superpixels) and (2) the update of the network parameters using the labeled source domain and the target domain inferred properties. In PyCDA the two approaches are merged: the pseudo-labels used in self-training are considered as a property of the curriculum approach.

2.4.6 Multi-Tasking

The last class of adaptation techniques to be discussed regards multi-tasking [56, 64, 76, 86]. A regularizing action is provided by solving multiple related tasks (*e.g.*, many approaches focus on depth regression) in addition to the semantic segmentation one. The goal is to implicitly extract domain invariant and semantically meaningful representations from images, as they should be more suitable to simultaneously address the related tasks. Depth regression is usually tackled in combination with semantic segmentation, to regularize an input level adaptation process based on source to target image translation (Section 2.4.2). Multi-task adaptation has also shown to be effectively integrated with other adaptation approaches. For example, it can be exploited to enhance the maximum classifier strategy (Section 2.4.3), where instead of a single dense classifier two separate decoder modules are employed to obtain both depth and segmentation maps [76]. Or it is possible to combine multi-tasking with an adversarial entropy minimization technique (Section 2.4.4) [56], by fusing self-information maps with depth prediction ones before feeding them to a domain discriminator. By doing so, the detection capability of the domain discriminator in identifying domain discrepancy over source and target representations is boosted, thus providing, in the end, a more robust statistical alignment.

2.5 Experimental Setups

The following sections will present one of the most common experimental settings employed to validate UDA methods for semantic segmentation, *i.e.*, the synthetic to real adaptation of driving scenes. We will focus on this adaptation setting due to its relevance to real-world applications, especially in relation to the rapidly emerging autonomous driving systems.

2.5.1 Synthetic to Real Adaptation

Domain adaptation is particularly convenient whenever it possible to access annotated training data in large quantity, even though samples are statically different from those in the label-scarce target environment, while maintaining some degree of similarity between datasets in terms of semantic content. In line with these premises, therefore, the synthetic-real approach emerges. Synthetic data is automatically collected and annotated by a computer, without resorting to extremely expensive, time consuming and error-prone human labeling. Thus, by levering artificial data generation, *e.g.*, such as via a realistic simulator [87] or game engine [18], it is possible to build large-scale datasets. In addition, due to the complete control that can be exercised over the simulated environment during the collection campaign, we could in

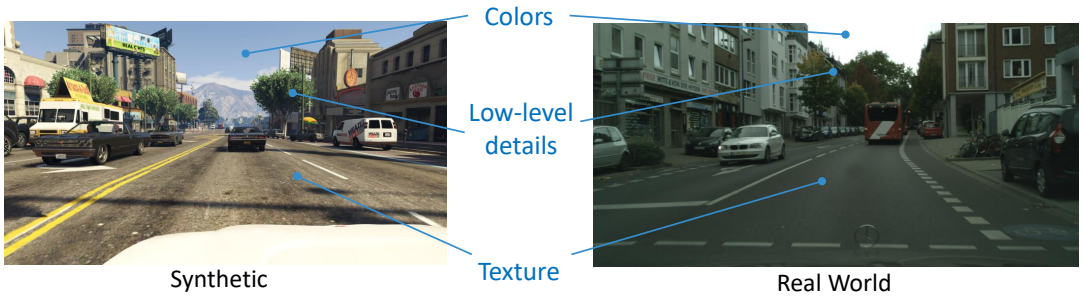


Figure 2.8: Comparison between synthetic and real images.

principle get extreme data variability to improve the learning performance and generalization aptitude for the optimized model. For instance, in case road scene imagery, diverse light and weather conditions, time of day or viewpoint could be easily acquired with minimal effort, with respect to a real-world collection campaign subject to environmental constraints.

On the other hand, however realistic the simulated images are, it is very likely that a distribution shift is present between real and synthetic data. This discrepancy usually entails the low-level statistical distribution, *e.g.*, texture, colors or smaller details, while the higher lever semantic concepts are preserved across domains (Figure 2.8). Simply training over source data leads to overfitting to these domain-dependent clues. By applying domain adaptation we, instead, aim to safely transfer information from source to target domains.

2.5.2 Employed Datasets

This section will introduce and describe some of the most popular real-world and synthetically generated datasets for driving applications, which will be used throughout the thesis for experimental analysis of the developed learning frameworks.

Real Datasets

In the following we will present the real-world road-scene datasets used in the evaluation campaign of semantic segmentation models (Figure 2.9).

Cityscapes. The Cityscapes [14] dataset is a widely known benchmark for autonomous driving applications. It contains images collected across 50 Europeans cities, all located in Central Europe, with almost the totality being within German borders. The provided training dataset is of 2975 samples, while for testing purposes we used the 500 labeled images in the original validation test, which is the common practice.

BDD. The Berkeley DeepDrive dataset [88] provides an assorted collection of road scenes, offering geographic, environmental and weather diversity to improve the robustness and gen-

Figure 2.9: Images from real-world datasets.

eralization aptitude of learned models. All images composing the whole set originate from the United States, even though from varying locations. In total 7000 training individual images are accompanied by dense ground-truth maps, while 1000 samples are used for testing.

IDD. The Indian Driving Dataset [16] comprises urban and suburban driving scenes from the Indian subcontinent. The visual appearance of the semantic categories provided by this benchmark is specific to the considered geographic areas, and differs significantly from the Western counterparts found in previous datasets. Similar in size to the BDD dataset, 6993 training and 781 testing images are provided.

Mapillary Vistas. The Mapillary Vistas dataset [17] contains high resolution images collected world-wide, with highly disparate acquisition settings and locations. The diversity in terms of semantic categories and acquisition settings ensure improved data expressiveness, essential to address the semantic segmentation task. Furthermore, unlike previously introduced benchmarks, samples are not limited to few cities located within quite uniform geographical regions. The training set is made of 20000 densely-labeled samples, and 2000 images are used for testing purposes. We will leverage the Mapillary dataset to artificially generate domain shift by performing continent-wise splitting, as well as to test the domain generalization potential of approaches trained with other less semantically diverse datasets.

Synthetic Datasets

We will additionally introduce some popular synthetic road-scene datasets, commonly employed to validate semantic segmentation models (Figure 2.10).

GTA5. The GTA5 dataset [18] contains 24966 synthetic images acquired through the

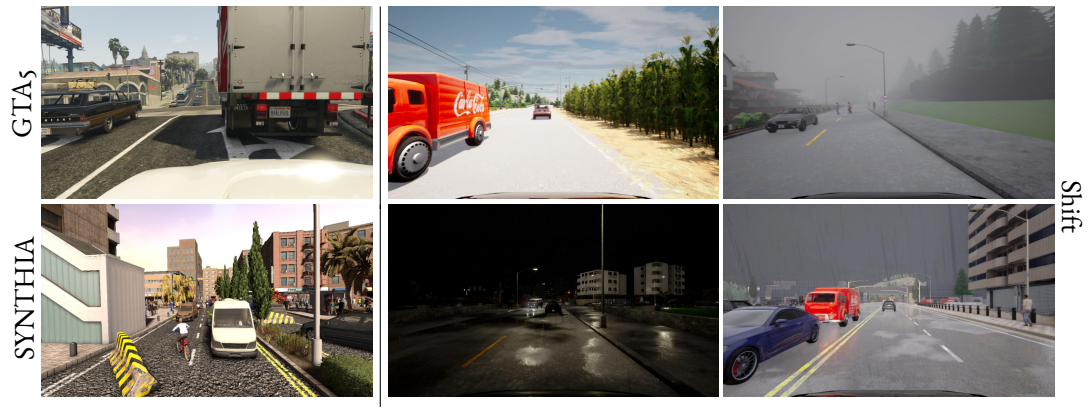


Figure 2.10: Images from synthetic datasets.

video game *Grand Theft Auto 5* and labeled into 19 classes compatible with real datasets. The images are taken from the car perspective in American-style virtual cities and are characterized by a high level of quality and realism. In all the works which will be presented in Part I of the thesis, when will make use of 23966 images for supervised training and 1000 for validation purposes. Furthermore, 19 semantic classes are provided, which are directly comparable with the ones from the Cityscapes dataset [14].

SYNTHIA. Images in the SYNTHIA dataset [19] have been generated through an ad-hoc engine and represent many types of street scenes, acquired from different angles (not only from car drivers viewpoint but also from video-surveillance cameras, from pedestrians, etc...) in various illumination and weather conditions in European-style cities. The visual quality is lower than the one of the GTA5 dataset. We will consider the *SYNTHIA-RAND-CITYSCAPES* subset of the SYNTHIA dataset, which is comprised of 9400 synthetic images and has 16 out of 19 semantic classes comparable with the ones of the Cityscapes dataset [14]. For the supervised training, 9300 images will be used, while 100 images will be held-out for validation purposes.

Shift. The Shift benchmark [89] is a synthetic dataset for autonomous driving, designed to simulate several types of input distribution shifts in both discrete and continuous forms, which could be highly beneficial to develop real-world applications. It presents 153k labeled samples used for training purposes, as well as 51k for testing. The Shift benchmark is particularly suitable to mimic domain shift due to environmental diversity.

3 Generative Input Adaptation

3.1 Introduction

In this section, we introduce an highly performing Unsupervised Domain Adaptation approach [234], with major focus on the input-level domain adaptation (see Section 2.3). The developed architecture is based on the Cycle-GAN framework [63], which converts the input synthetic images to the target (real) domain while preserving the semantic content. Then, the data is sent to a lightweight MobileNet-v2 prediction network [90] that performs the semantic segmentation. An additional loss component forces also the consistency between the semantic maps, thus avoiding the risk that the domain translation affects the semantic content. Finally, as a side objective, we enhance adaptation by means of an additional couple of discriminators working at the intermediate feature level of the network.

Differently from other competing approaches that train independently the various subcomponents, we train the complete architecture end-to-end on both synthetic labeled data and unlabeled real-world data in a single optimization framework based on adversarial learning. Finally, using the simple and fast MobileNet-v2 architecture, the inference stage of the approach is suitable for real-time applications as the autonomous driving scenario chosen for the experimental evaluation.

The experimental evaluation, performed using the synthetic datasets SYNTHIA [19] and GTA5 [18] and the real-world dataset Cityscapes [14], shows how the proposed framework is able to achieve large performance gains on real-world datasets without using any labeled real-world data during training.

The contributions brought in this chapter are the following:

(i) We propose an input-level adaptation strategy that relies on an image-to-image translation mechanism, which encompasses geometrical coherence and semantic consistency within its image generation pipeline to provide reliable pseudo supervision in the originally unlabeled target domain.

(ii) We devise an end-to-end training procedure, based on a lightweight segmentation network for a more application-oriented solution.

(ii) The developed framework surpasses other techniques resorting to the domain adaptation of the input space on multiple synthetic-to-real benchmarks.

The reminder of the chapter is organized as follows: the proposed UDA method based on cycle- and semantic- consistent image-to-image translation will be described in detail in Section 3.2, while training procedures will be presented in Section 3.3; finally, the experimental evaluation will be discussed in Section 3.4.

3.2 Cycle Consistent Domain Adaptation

Our target is to train a semantic segmentation network in a supervised way on synthetic data and then to adapt it in an unsupervised way to real-world data. We assume to have access to synthetic (*i.e.*, source) labeled images $\{(\mathbf{X}_n^s, \mathbf{Y}_n^s)\}_n$, $(\mathbf{X}^s, \mathbf{Y}^s) \in \mathcal{X}_S \times \mathcal{Y}_S$, as well as to real (*i.e.*, target) unlabeled images $\{\mathbf{X}_n^t\}_n$, $\mathbf{X}_n^t \in \mathcal{X}_T$. Due to dissimilar marginal and joint distributions over input and label spaces on both domains, deep models trained on source data struggle to generalize learned knowledge to the target space. To address the effect of domain discrepancy, we resort to a generative approach. We employ an adversarial framework to learn an image-level mapping between source and target spaces. The objective is to produce adapted source images that resemble target ones, while preserving the ground-truth information at our disposal. In this way, we can introduce a form of target supervision by exploiting target-like annotated source images to train the segmentation network.

Figure 3.1 shows the architecture of the proposed framework. As initial step, we adopt a generative approach to learn an image-level mapping for cross-domain image projection. This is achieved using an adversarial learning scheme exploiting a pair of generator-discriminator couples. Meanwhile, a semantic segmentation network is included to enforce semantic consistency to the generative process. The objective is to perform realistic sample translations, while preserving the semantic structure as identified by the semantic classifier. An additional feature-level adaptation is further included, in the form of a pair of feature discriminators. The way they operate is analogous to their image-level counterparts, as they enforce a statistical alignment of source and target data representations. The key difference lies in

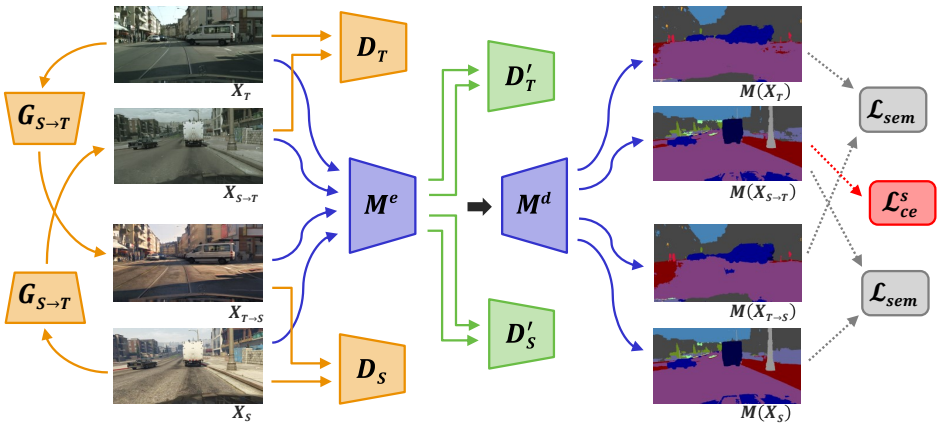


Figure 3.1: Architecture of the proposed framework. Yellow blocks correspond to the CycleGAN module for image-to-image translation. Original and translated scenes from both source and target sets are projected by the encoder to a latent space on which we apply an extra couple of domain discriminators (green blocks). Structural consistency on generated samples is enforced by the cycle-consistency constraint, whereas semantic uniformity throughout image mapping is promoted by the semantic loss. The segmentation network is reported in blue.

the operating space, since they act over an intermediate feature representation produced by the segmentation network, rather than directly within the original image domain.

Our approach is independent of the segmentation architecture and in general any semantic segmentation network can be used, however in our experiments we used the MobileNet-v2 network [90] embedded inside the DeepLab-v3+ framework [12]. The primary component of this widely utilized model is the depthwise separable convolution, a lightweight reinterpretation of the standard convolutional layer responsible for both the efficiency and the reduced weight of the architecture. In addition, inverted residual blocks in place of the standard residual connections further enhance model compactness.

3.2.1 Consistency of Input Appearance Across Domains

The generative module is based on the CycleGAN framework [63]. The source to target (direct) mapping $G_{S\to T}$: $\mathcal{X}_S \to \mathcal{X}_T$ and the target to source one (inverse mapping) $G_{T\to S}$: $\mathcal{X}_T \to \mathcal{X}_S$ are discovered by means of an adversarial competition exploiting a couple of discriminators D_S and D_T. The role of the domain discriminators, following the original concept of GANs [31], is to discriminate between *real* images in their original form and *fake* images, *i.e.*, synthetic data subjected to the domain translation. We resort to the standard adversarial objectives at the image level to train separately each generator-discriminator couple:

$$\mathcal{L}_{i,T} = \sum_{\mathbf{X}^t \in \mathcal{X}_T} \log\left(D_T(\mathbf{X}^t)\right) + \sum_{\mathbf{X}^s \in \mathcal{X}_S} \log(1 - D_T(G_{S\to T}(\mathbf{X}^s))), \tag{3.1}$$

$$\mathcal{L}_{i,S} = \sum_{\mathbf{X}^s \in \mathcal{X}_S} \log(D_S(\mathbf{X}^s)) + \sum_{\mathbf{X}^t \in \mathcal{X}_T} \log(1 - D_S(G_{T \to S}(\mathbf{X}^t))). \tag{3.2}$$

With the aforementioned generative process, we are able to learn a joint source-target distribution starting from the marginal ones, which means we can reproduce the same images in both source and target styles. Unfortunately, since there are infinite joint distributions that match the available marginal ones, we are not guaranteed that the mapping functions we discover are preserving content structure and semantics. In other words, without any additional constraint the $G_{S \to T}$ projection could completely disrupt input source images, still producing new samples with target properties, but far from their original versions. For this reason, we employ an additional loss term enforcing cycle-consistency:

$$\mathcal{L}_{cycle} = \sum_{\mathbf{X}^s \in \mathcal{X}_S} \left[\| G_{T \to S}\left(G_{S \to T}\left(\mathbf{X}^s\right)\right) - \mathbf{X}^s \|_1 \right] + \sum_{\mathbf{X}^t \in \mathcal{X}_T} \left[\| G_{S \to T}\left(G_{T \to S}\left(\mathbf{X}^t\right)\right) - \mathbf{X}^t \|_1 \right]. \tag{3.3}$$

The reconstruction requirement provided by the cycle-consistency loss $\mathcal{L}_{cycle}$ should encourage the preservation of structural properties throughout translations, resulting in a realistic image generation that does not affect the semantic content.

The adversarial strategy we adopt for conditional image generation has proven to be suitable for color and texture changes, but not for more radical geometrical transformations [31]. This is positive for our goal, since we are looking for a cross-domain projection that allows us to safely transfer ground-truth information from an original image to its translated version.

3.2.2 Semantic Consistency Across Domains

Following the the idea introduced in [40], we embed the generative module in a task-specific domain adaptation framework. The adversarial architecture is followed by a network performing semantic segmentation on data from both source and target domains. In our work, we will assume the usage of a fully convolutional network, that in our implementation is the MobileNet-v2 network. We will denote it with $M = M^d \circ M^e$, where M^e is the encoder part of the network, while M^d is the decoder. M is pre-trained on the source domain, before its application in the proposed framework.

Due to the lack of labeling data on the target domain, we can not perform supervised training on this domain. However, we introduce a further loss component to enforce the semantic consistency on the generative action: the segmentation network M is supplied with both original and adapted versions of the same image and we measure the semantic discrepancies (*i.e.*, the differences in the segmentation network output) introduced by the projection between domains. The error information is then propagated back through the classifier up

to the generator, which is optimized in order to minimize semantic alteration (among other objectives). We impose this semantic uniformity by means of a semantic consistency loss:

$$\begin{aligned}\mathcal{L}_{sem}&\left(G_{S\to T}, G_{T\to S}, M, \mathcal{X}_S, \mathcal{X}_T\right)\\ &= \mathcal{L}_{ce}\left(M, G_{S\to T}(\mathcal{X}_S), \rho(M(\mathcal{X}_S))\right)\\ &+ \mathcal{L}_{ce}\left(M, G_{T\to S}(\mathcal{X}_T), \rho(M(\mathcal{X}_T))\right),\end{aligned} \tag{3.4}$$

where $\rho(M(\mathcal{X}))$ is the $\arg\max$ of the output of the semantic segmentation network and:

$$\mathcal{L}_{ce}\left(M, \mathcal{X}, \mathcal{Y}\right) = -\sum_{p\in\mathbf{X}_n}\sum_{c\in C}\mathbf{Y}_n^{(p)}[c]\cdot\log(M(\mathbf{X})_n^{(p)}[c]), \tag{3.5}$$

with $(\mathbf{X}_n, \mathbf{Y}_n) \in \mathcal{X}\times\mathcal{Y}$ and $\mathcal{C}$ the set of semantic classes to learn. Notice that the cross-entropy loss $\mathcal{L}_{ce}$ is computed over segmentation maps obtained by applying the $argmax$ function ρ on semantic predictions $M(\mathcal{X})$, rather than over ground-truth labels. Therefore, its effect is not to promote correct semantic predictions, but to force the generator to yield transformed images that are semantically identical to the original ones when viewed under the scope of M.

This scheme allows us to perform a measure of semantic distance in both domains, without resorting to ground-truth information. On the other side, M is not a perfect predictor (and on the target domain has typically lower performances), therefore an excessive emphasis on this loss may cause undesired artifacts in the generative process.

3.2.3 Consistency of Latent Representation Across Domains

Aiming at further improving our domain adaptation framework, we introduce an additional adversarial module to perform feature-level domain adaptation. The core idea is to replicate the adversarial strategy adopted for the image-to-image translation task, where the goal was a pixel-level distribution alignment. The new objective is instead the adaptation of intermediate feature representations, *i.e.*, to ensure the proper adaptation at the level of the output of the encoder network M^e. The goal now is to make generated images from one domain appear statistically identical to original images from the other domain when looking at their projections in a latent space spanned by the segmentation network. More specifically, we add a couple of feature discriminators and feed them with activations from the output of the MobileNet feature extractor M^e. The adversarial game, then, takes place between a couple of feature discriminators and the joint action of the two original generators together with the encoder of the segmentation network. The feature-level adversarial objectives are the fol-

lowing:

$$\mathcal{L}_{f,T} = \sum_{\mathbf{X}^s \in \mathcal{X}_S} \log(D'_T(M^e(\mathbf{X}^t))) + \sum_{\mathbf{X}^t \in \mathcal{X}_T} \log(1 - D'_T(M^e(G_{S\to T}(\mathbf{X}^s)))), \tag{3.6}$$

$$\mathcal{L}_{f,S} = \sum_{\mathbf{X}^s \in \mathcal{X}_S} \log(D'_S(M^e(\mathbf{X}^s))) + \sum_{\mathbf{X}^t \in \mathcal{X}_T} \log(1 - D'_S(M^e(G_{T\to S}(\mathbf{X}^t)))). \tag{3.7}$$

Where D'_S and D'_T are the feature discriminators working on data from the source and target domain respectively.

Combining together all the different losses, the full objective becomes:

$$\begin{aligned} \mathcal{L}_{tot} = (\mathcal{L}_{i,S} + \mathcal{L}_{i,T}) + \lambda_{feat} \cdot (\mathcal{L}_{f,S} + \mathcal{L}_{f,T}) \\ + \lambda_{cycle} \cdot \mathcal{L}_{cycle} + \lambda_{sem} \cdot \mathcal{L}_{sem} + \lambda_{ce} \cdot \mathcal{L}^s_{ce}. \end{aligned} \tag{3.8}$$

We also denote $\mathcal{L}^s_{ce} = \mathcal{L}_{ce}(M, G_{S\to T}(\mathcal{X}_S), \mathcal{Y}_S)$ the standard cross-entropy loss used to train supervisedly M on source adapted data. The framework optimization then can be expressed as a min-max problem:

$$\min_{G_{S\to T}, G_{T\to S}, M} \max_{D_S, D_T, D'_S, D'_T} \mathcal{L}_{tot}. \tag{3.9}$$

As a result, we have access to a pair of image-to-image mappings capable of translating images across domains, while, at the same time, making generated samples statistically indistinguishable from true ones when projected into the feature space defined by the segmentation network. Additionally, the semantic segmentation network M is adapted to work on the target data in an unsupervised way (without using target ground truth) thanks to the loss $\mathcal{L}^s_{ce}$. Since all the components are simultaneously trained, when improving the image-to-image mappings also the adaptation of M improves. For a more stable training and to avoid saturation effects [63], the logarithm within adversarial losses is replaced by a L2-norm operator and the objectives are split into separate terms for generators and discriminators individual optimization.

3.3 Experimental Setup

Datasets In order to perform the experimental evaluation, we selected a source domain with easily accessible labeled samples that can be obtained in large quantities, *i.e.*, synthetic imagery. As discussed in Section 2.5, synthetic data not only comes with essentially free annotations and a much lower collection cost than real data thanks to automatic generation, but also in principle provides a total control over the virtual environment in terms of point of

view, illumination, objects inside the scene, etc. On the opposite side, we used real images as the target domain. Before going into details, notice that even if the experimental evaluation is performed on the synthetic to real adaptation, the proposed approach can be exploited in any domain adaptation task, not only in this specific setting.

We assume no ground-truth information is accessible in any form for real data (we used real labels only for evaluation of the results), as we are in a fully unsupervised scenario in this domain. As concerns the specific datasets used to train and validate our framework, we choose the publicly accessible GTA5 [18], SYNTHIA [19] and Cityscapes [14] datasets. They are built specifically to address semantic segmentation of urban scenes, which is of strategic importance in autonomous driving. The task is quite challenging, since multiple objects of different sizes with very different occurrence frequencies and various semantic categories have to be recognized with high confidence.

Network implementation The image-level generators and discriminators (*i.e.*, $G_{S\to T}$, $G_{T\to S}$, D_S and D_T) are based on the network architectures introduced in [63]. The generators are composed of stride-2 convolutions, residual blocks and fractionally strided convolutions to recover input dimensionality. Discriminators are fully convolutional networks as well, made by the cascade of 5 stride-2 convolutional layers. To avoid excessive size compression, we employ the same structure for feature-level discriminators (D_S^f and D_T^f) as well, but we modify the stride value to 1 for all layers. As concerns the segmentation network M, we employ the DeepLab-v3+ [12] with the MobileNet-v2 [90] as backbone. We select an output stride of 16 for the feature extractor, whereas for the ASPP block we choose atrous rates of 6, 12 and 18 as suggested by [12].

Training details Differently from [40], we train our framework in a single shot, so that all the networks are simultaneously optimized according to $\mathcal{L}_{tot}$. We then fine-tune it on the source domain for 90K steps using the standard cross entropy loss before the actual optimization of the adaptation framework. All the other networks are instead trained from scratch. We use the Adam optimizer [91] to train all the components of the proposed approach.

After the initialization of the segmentation network, we train our model for a total of 80K iterations with a single NVDIA GeForce GTX 1080 Ti. The segmentation network is kept fixed for the first 20K steps, until the generators start performing acceptable translations, then we train all the various components together. The large amount of model parameters and the high resolution images from source and target sets prevents us from using full size data for training purposes. To overcome this issue, we extract random patches of 600×600 pixels from training samples, which were previously resized to have a predefined width and the original aspect ratio, and we use them as model inputs.

Concerning the parameters, we experimentally set the terms balancing the various components in $\mathcal{L}_{tot}$ to $\lambda_{cycle} = 20$, $\lambda_{sem} = 0.1$, $\lambda_{feat} = 1\text{e-}4$ and $\lambda_{ce} = 1$. For the training of the segmentation network we set β_1 (the exponential decay rate for the first moment estimates) to 0.9, and the weight decay to $4\text{e-}5$. The learning rate is subject to a polynomial decay of power 0.9, and is decreased to 0 from its initial value (respectively $1\text{e-}5$ and $5\text{e-}6$ for the adaptation from the GTA and SYNTHIA datasets). Moreover, we set a batch size of 5 when training the semantic predictor alone in the initial stage, while for full framework optimization, we reduce the batch size to 1 due to memory constraints. For the optimization of the image and feature adversarial models we use a small β_1 term of 0.5 as in [63]. This makes the training process more unstable, but we notice no improvements by changing it. The learning rate for these modules is set to $2\text{e-}4$.

3.4 Experimental Results

In this section, we will fist show some qualitative results from the image translation module. Then, we will show quantitative and qualitative results of the main task, *i.e.*, semantic segmentation on real data, starting from two different synthetic datasets. Finally, some ablation analyses are presented.

3.4.1 Image Domain Translation

The first module in our framework is the image-to-image adaptation in the image space between the synthetic datasets and the real one (Cityscapes) and viceversa. We visualize such cyclic translation in Figure 3.3, where we report the original, adapted and reconstructed images in output from our model for each of the four different considered scenarios. We show both the results when starting from synthetic data and when starting from real scenes.

The most obvious and noticeable difference lies in the shift of colors between real and synthetic images. Synthetic imagery, indeed, are characterized by more vivid colors than the real counterparts hence the adaptation framework needs to compensate for this issue as can be verified in all the proposed qualitative results. Additionally, the textures of some regions (such as the road or the sky) of the images are completely different between the considered domains. In GTA5 and especially in SYNTHIA (where the road texture is not realistic at all) the road is less uniform than the one present in the Cityscapes dataset, so we could observe that our adaptation framework compensate this aspect. In particular, the model adds some textures on the road when converting an image to the synthetic dataset and it removes such textures when dealing with the opposite task. The sky, instead, tends to appear more blue in

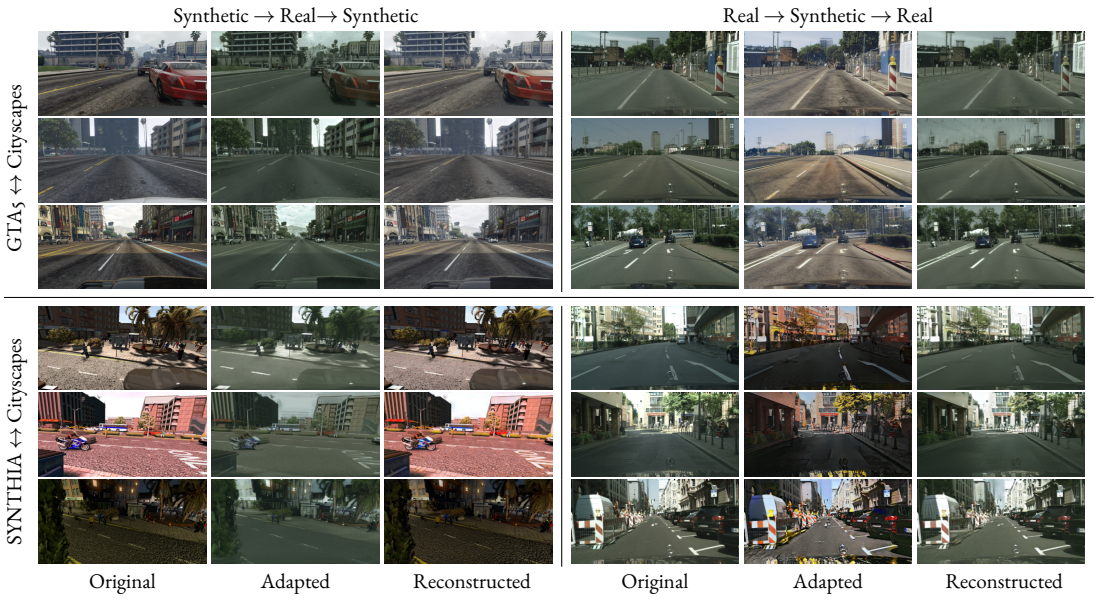


Figure 3.2: Examples of image translations. In the first up-left quadrant, we move from GTA5 to the adapted image space resembling that of Cityscapes, and back to the reconstructed space in the GTA5 domain. The second quadrant (up-right) shows the translation in opposite direction, *i.e.*, from Cityscapes to GTA5, and then back to the original real Cityscapes domain. The third (down-left) and the fourth (down-right) quadrant are analogous to the previous ones, but with SYNTHIA in place of GTA5.

the synthetic imagery rather than in the real ones, where it appears more grayish.

Beside those changes in the appearance of the images, we could notice that in general the semantic content and the geometrical structure is preserved unaltered. The objects do not disappear and they do not change position, as we expect. This is ensured by a combination of factors such as the semantic loss, the cycle-consistency loss and the adaptation loss at the feature level which aims at preserving the extraction of similar features from the same objects even if they belong to two different image-spaces.

Furthermore, when moving from synthetic to real domain, in certain pictures we can observe that our model tends to add an ornament to the bottom of the images (*e.g.*, in the second row from GTA5 to Cityscapes). Although this is irrelevant for the semantic segmentation task, since we exclude the car on which the camera is mounted, we argue that the image-space adaptation is leading to reasonable outcomes because it tries to replicate the trademark of the car used for the Cityscapes data acquisition.

A few artifacts are present especially in the sky region where the model tends to add some shadows coming from clouds or buildings present in other images: this can be noticed in the first and third adapted images from the GTA5 dataset. However, it is noticeable that the cycle-consistency helps the model to recover the exact appearance of the sky of such images.

Table 3.1: Results in terms of per-class and mean IoU on the Cityscapes validation set when adapting from GTA5 (top) and SYNTHIA (bottom). The highest values have been highlighted in bold.

	Method	Road	Sidewalk	Building	Wall	Fence	Pole	T. Light	T. Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorbike	Bicycle	mIoU
GTA5 → Cityscapes	Source only	23.1	13.1	42.6	2.3	13.9	5.0	10.3	8.0	68.6	6.7	24.5	40.8	0.3	48.1	9.4	16.3	0.0	0.0	0.0	17.5
	Ours (full) [234]	**87.6**	**36.7**	**83.5**	**29.1**	**17.8**	**33.6**	24.3	**35.2**	**83.1**	**28.9**	**76.3**	**59.1**	**14.0**	**85.9**	**25.4**	**29.4**	**2.6**	**19.5**	**9.3**	**41.1**
	CycleGAN [63]	84.9	36.4	74.3	12.9	7.1	23.6	7.9	19.9	60.2	13.4	45.8	46.8	5.4	72.4	18.0	22.3	0.8	3.2	0.5	29.3
	CyCADA [40]	83.8	35.3	80.4	20.7	15.7	28.4	**27.0**	24.8	80.2	23.1	69.0	56.6	11.5	80.8	23.4	27.0	2.4	12.4	5.2	37.3
	Oracle	97.7	81.9	91.0	47.6	50.1	58.4	62.3	73.4	91.4	59.8	94.3	77.2	50.5	93.2	59.2	74.8	55.8	49.5	73.0	70.6
SYNTHIA → Cityscapes	Source only	1.4	10.6	29.1	1.0	0.0	17.2	2.0	3.6	68.5	-	65.0	42.3	0.1	41.7	-	8.2	-	0.0	0.5	18.2
	Ours (full) [234]	**53.8**	21.3	**69.4**	3.7	**0.1**	**31.6**	3.5	**12.6**	**77.5**	-	**75.2**	**51.9**	**13.2**	**64.1**	-	**15.9**	-	**10.8**	**16.7**	**32.6**
	CycleGAN [63]	42.9	**26.7**	44.6	0.5	**0.1**	29.2	3.5	5.8	67.1	-	70.8	46.0	3.4	32.8	-	7.0	-	2.4	3.7	24.2
	CyCADA [40]	30.3	15.8	64.0	**5.9**	0.0	30.6	**3.8**	10.4	76.4	-	73.0	42.9	4.9	54.3	-	15.0	-	3.0	8.9	27.5
	Oracle	97.8	83.7	91.2	47.7	49.7	58.8	63.0	73.9	92.4	-	94.4	77.9	53.7	94.1	-	80.5	-	44.7	73.5	73.6

3.4.2 Adaptation from GTA5 to Cityscapes

The first set of experiments regards the unsupervised adaptation of the semantic segmentation network M to the Cityscapes dataset after an initial stage of supervised training on the GTA5 dataset. To evaluate the adaptation performance of our framework we computed the mean Intersection over Union (mIoU) between model predictions and the relative ground-truth label maps for the scenes in the Cityscapes validation set.

The results of these synthetic to real adaptation experiments are summarized in Table 3.1. The baseline approach, *i.e.*, the supervised training of the semantic segmentation on the GTA5 (source) dataset followed by testing on Cityscapes without any adaptation, leads to a very low mIoU of 17.5% (first row). When compared to the training of the same network on the target (Cityscapes) dataset (last row, denoted as *Oracle*), the huge difference reveals the struggle of the predictor to overcome the statistical discrepancy between the source and target domains.

Our unsupervised domain adaptation method brings a huge improvement over the naïve source only approach, reaching a mIoU of 41.1% when employed in its full extent (5th row), with a performance boost of 23.6% over the baseline. Moreover, the accuracy enhancement is well distributed over all the semantic classes, from the most common ones (*e.g.*, *road*, *building*, *sky*), to the less frequent categories (*e.g.*, *train*, *motobike*, *bike*). In particular notice that some of the less frequent are never recognized when no adaptation is performed while the

proposed approach allows to detect them even if they remain very challenging. This proves the effectiveness of our model in mitigating the statistical discrepancy through a combined feature and pixel level alignment.

We compared the results we obtained on the Cityscapes validation set with two approaches of the same family: namely, CycleGAN [63] and CyCADA [40]. Those are very well known approaches and also represent the starting point for some architectural design choices we made. For comparison purposes, we implemented the framework of [40] from scratch and inserted in it the same segmentation network we used in our tests (*i.e.*, DeepLabV3+ [12] with MobileNet-v2 [90] as backbone). As for the training details of the compared methods, we employed the standard training procedures proposed in the respective papers: for CycleGAN [63] we train the model with rescaled images, while for CyCADA [40] we train the model extracting random patches after a rescaling operation.

From Table 3.1 we can appreciate that our method is able to outperform CyCADA by about 4% and 5% respectively, which is uniformly distributed among classes: notice how our approach is the best on 18 out of 19 classes (it is outperformed only by [40] on the traffic light class). This has to be mostly attributed to the feature alignment process which directly influences the generative action and also to the simultaneous optimization of all framework components. Additionally, we could observe that CycleGAN has a much lower accuracy and lies in between the training made only on source data and our full method. Indeed, it consists of a simpler framework where only the image translation based on the cycle-consistency constraint is present.

Figure 3.3 displays some qualitative results in terms of semantic prediction maps for different adaptation strategies. The first and second columns include the original Cityscapes RGB images and their corresponding label maps, while the last 3 columns show the segmentation outputs with no adaptation (*i.e.*, *source only*), using the approach of [40] and with the proposed unsupervised domain adaptation strategy.

The accuracy enhancement introduced by our adaptation strategy can be appreciated from the visual results in Figure 3.3. The first 4 rows show the remarkable improvement achieved over the *source only* baseline when resorting to the GTA5 dataset as source domain. The effect of the adaptation is to boost the detection capability of the predictor M, as it manages to obtain a more accurate understanding of the input target scene, both in terms of correct categorization and precise spatial identification of the different semantic entities. All semantic classes happen to highly benefit from the adaptation, leading to generally better semantic predictions. Anyway some categories exhibit more noticeable improvements, such as the *road* and *sky* classes as it is possible to see on the predicted maps in rows 1, 2 and 4. Column 4 of Figure 3.3 shows the output of [40]: our method outperforms [40] on the majority of the semantic classes, with a particularly noticeable improvement on some of them.

For instance, our approach in general leads to a more precise segmentation of the upper portion of target scenes, mainly involving the *sky* and *building* classes. It is possible to notice how the approach of [40] sometimes confuses part of *sky* as *terrain* or *building* (rows 1, 2 and 4). As previously stated, this is due to the dataset bias occurring between source and target domains in terms of color shift and change of texture, making the detection of semantic components quite hard to achieve, especially when the classification involves different categories (such as *sky* and *building*) sharing a common appearance on the source and target domains. Furthermore, we observe that less frequent classes corresponding to small image details (*e.g.*, *traffic sign*) are more precisely localized, as well as more common categories lacking an always definite semantic categorization (*e.g.*, *wall* and *sidewalk*). Once again, this remarks the capability of our adaptation framework to address domain discrepancy with the combined pixel and feature level alignment.

3.4.3 Adaptation from the SYNTHIA dataset

In the second set of experiments we changed the source domain, replacing the GTA5 dataset with the SYNTHIA one. Table 3.1 shows the numerical results we got from the evaluation. As before, we started by training the semantic segmentation network on synthetic data and measuring its performance on the Cityscapes validation set, which we employ as a baseline (first row). The accuracy (18.2%) happens to be slightly higher than in the first scenario, even if the adaptation task is more challenging due to a wider dataset bias. Anyway, the huge performance gap w.r.t. the target-based supervised optimization still persists, leaving large room for improvement.

Our adaptation framework managed to reduce the domain discrepancy quite successfully boosting the mIoU up to 32.6% from the original 18.2% of the baseline, with an improvement of almost 15%. As for the GTA5 to Cityscapes adaptation, the accuracies for all semantic categories benefit from the unsupervised adaptation. For example, road segmentation is strongly improved, starting from a mIoU for the road class of 1.4% (denoting basically no detection capability) and reaching a final accuracy of 53.8%. This class is particularly interesting since the reason for the low accuracy lies in the rather unrealistic synthetic road texture of SYNTHIA images, whose semantic attributes are not easily generalizable to the Cityscapes dataset. To that end, our framework introduces a substantial distribution alignment both at pixel level (as discussed in Section 3.4.1) and inside the intermediate feature space. We once again compared the adaptation performance of our model with the one of CyCADA [40] in the third row. Our framework is more effective than the competitor, with an average mIoU increase of 5%, shared by the majority of the classes. For instance, the road accuracy is significantly enhanced by our model, proving the better capability of reducing domain dis-

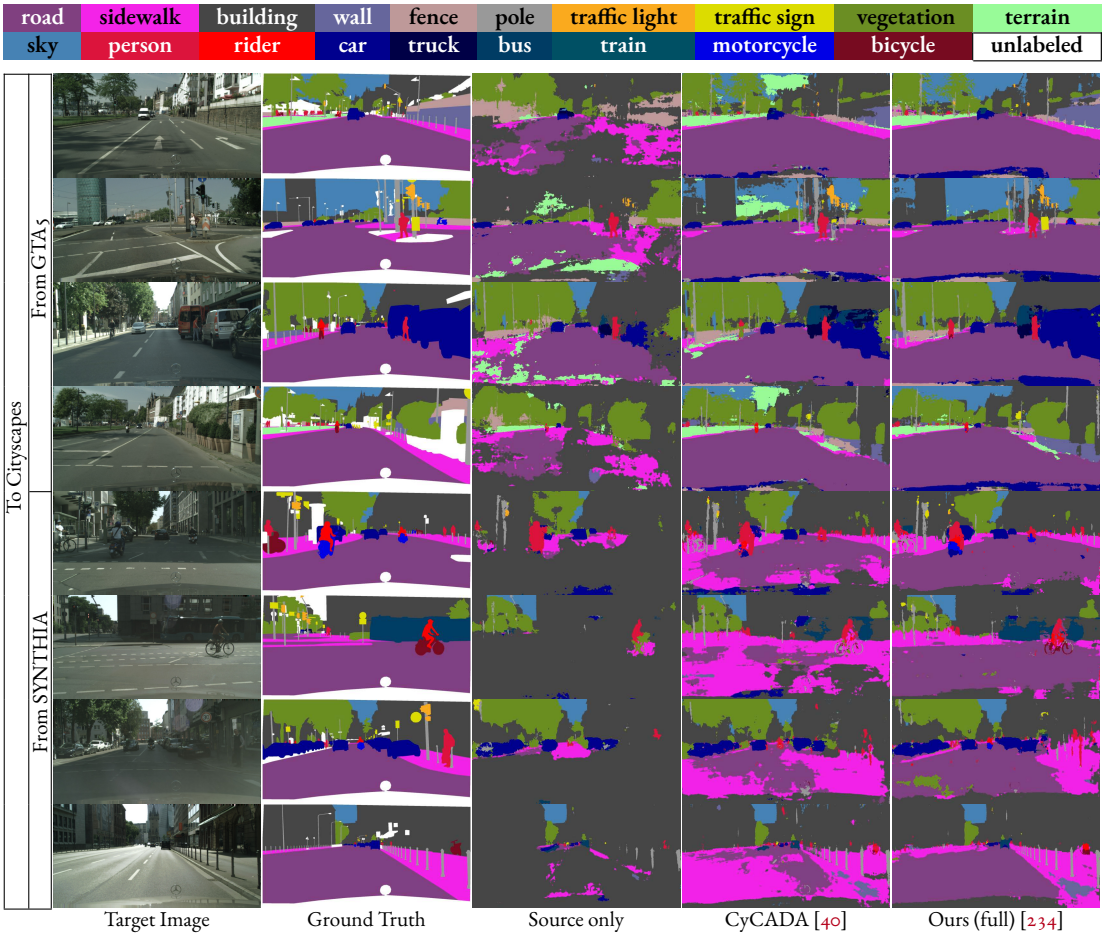


Figure 3.3: Semantic segmentation of sample scenes from the Cityscapes validation set when adapting from GTA5 (rows 1 to 4) and SYNTHIA (rows 5 to 8).

crepancy. Again, the CycleGAN [63] approach has intermediate results between the source only approach and CyCADA [40].

Some qualitative results for the adaptation from the SYNTHIA dataset are shown last 4 rows of Figure 3.3. They confirm the numerical evaluation: the strong diversity between the Cityscapes and SYNTHIA datasets negatively affects the semantic understanding of the predictor, as clearly visible in the third column showing the outputs of the baseline approach with no adaptation. Even common classes such as *road* and *building* suffer from quite flawed semantic detection leading to almost no understanding of target scene structure, with a worse performance when compared to the GTA5 scenario. This has to be ascribed to the poor realism of synthetic images and to the variable view point of SYNTHIA scenes, which are captured from multiple camera angles and not exclusively from a car perspective as for the Cityscapes images. The adaptation successfully mitigates the domain discrepancy providing

Table 3.2: Ablation study on the impact of different objectives. Results are reported in terms of per-class and mean IoU on the Cityscapes validation set when adapting from GTA5 (top) and SYNTHIA (bottom). The highest values have been highlighted in bold.

Method	Road	Sidewalk	Building	Wall	Fence	Pole	T. Light	T. Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorbike	Bicycle	mIoU
Source only	23.1	13.1	42.6	2.3	13.9	5.0	10.3	8.0	68.6	6.7	24.5	40.8	0.3	48.1	9.4	16.3	0.0	0.0	0.0	17.5
$\lambda_{sem}, \lambda_{feat} = 0$	87.8	**39.1**	81.0	24.8	16.0	31.9	27.2	25.7	78.6	22.9	69.7	55.5	**16.7**	85.3	25.7	31.0	**4.4**	14.7	5.6	39.1
$\lambda_{sem} = 0$	**88.3**	37.1	81.1	25.4	15.3	**33.6**	**29.5**	28.8	80.0	24.4	69.2	56.0	15.6	85.0	25.5	30.8	3.9	16.3	6.2	39.6
$\lambda_{feat} = 0$	87.3	36.6	82.8	**29.1**	**19.9**	32.9	24.9	32.3	82.8	28.1	74.6	58.3	11.6	**85.9**	**26.3**	**31.9**	1.3	**19.5**	6.8	40.7
Ours (full) [234]	87.6	36.7	**83.5**	**29.1**	17.8	**33.6**	24.3	**35.2**	**83.1**	**28.9**	**76.3**	**59.1**	14.0	**85.9**	25.4	29.4	2.6	**19.5**	**9.3**	**41.1**
Oracle	97.7	81.9	91.0	47.6	50.1	58.4	62.3	73.4	91.4	59.8	94.3	77.2	50.5	93.2	59.2	74.8	55.8	49.5	73.0	70.6

the predictor with an enhanced perception of the semantic morphology of target inputs. For example, the *road*, which without adaption is incorrectly classified as *building*, probably due to its quite unrealistic texture on source images, after the adaptation is detected with a much greater accuracy.

Furthermore, our approach shows some improvement also w.r.t. the method proposed in [40]. For example, the adaption following [40] struggles in the correct segmentation of *road* and *sidewalk* classes, which are easily mistaken one for the other, an issue greatly reduced in the maps of our approach in the last column. At the same time, semantic predictions exhibit a better detection accuracy on objects belonging to low frequency classes, such as *motorbike* and *bike*, highlighting an increased robustness of our method due to the combined pixel and feature level alignment and to the simultaneous optimization of all the network components.

3.4.4 Ablation Study

Finally, we analyze in detail the performance gain brought by the different components of our framework. For this evaluation we considered the domain adaptation from GTA5 to Cityscapes. A relevant contribution is due to the CycleGAN-based image-to-image translator, which effectively bridges the domain gap at the pixel level by generating realistic target-like labeled data and pushes the accuracy on the Cityscapes validation set up to 39.1% (2nd row in Table 3.2). However, notice that this value is significantly different from the original CycleGAN [63] result reported in Table 3.1 because in our framework the optimization of the segmentation network is made jointly with the cyclic translation and the training considers patches of images. Then, we evaluated the impact on the final adaptation performance of the semantic and feature-based losses by alternately setting to 0 the λ_{sem} and λ_{feat} param-

eters. The regularizing action of the feature level adaptation allows to increase the accuracy to 39.6% (3rd row in Table 3.2) with a small but noticeable impact. The semantic loss (4th row in Table 3.2) has a larger impact, leading to an improvement of 1.6% on the final score. Finally, by using both components together we obtain a combined improvement of 2%, leading to the final accuracy of 41.1%.

4

Adversarial Output Adaptation

4.1 Introduction

In the previous chapter we focused on input data to reduce the domain gap inside the image space and address synthetic-to-real adaptation. In this chapter, instead, we develop an output-level UDA strategy for road driving scenes to adapt an initial learning performed on synthetic data to the real-world case [238]. The complete framework is made of four components. First, a standard cross-entropy loss is employed to perform a supervised training on synthetic data with ground-truth annotations. Then, two adversarial learning schemes with a couple of fully convolutional discriminators are jointly employed. We start from the structure used in [92, 93] for semi-supervised semantic segmentation and adapted in [53, 54] to Unsupervised Domain Adaptation. However, the devised method leverages two discriminative networks, one to differentiate between ground-truth and predicted segmentations and the other to discriminate between segmentation maps coming from synthetic and real-world data. Finally, a self-training component is introduced based on the idea that the output of the discriminator provides a measure of the reliability of the network estimations to be exploited in a self-training framework [53, 92]. In previous works, this module lacked two fundamental aspects: it was not class-wise adaptive and was not mutable during training. In other words, different classes shared the same confidence threshold to select regions for self-training, and its value was kept fixed during all the learning process. In this work, instead, we introduce an adaptive thresholding scheme for the selection of the regions used for self-training that enforces a balanced selection for all classes and allows for variability over different training steps. Hence, the framework can accomplish both inter-class confidence

flexibility and time adaptability throughout the optimization phase. We test our model on the task of domain adaptation for semantic segmentation of urban scenes from synthetic to real-world domains. In particular, we employ the SYNTHIA and GTA5 synthetic datasets to train the supervised component, whereas we resort to real data from Cityscapes and Mapillary datasets for the unsupervised adaptation modules.

In summary, the main contributions of this chapter are:

(i) We introduce a novel adversarial scheme exploiting multiple domain discriminators to align the source and target domains.

(ii) We design a self-training module with adaptive confidence both over different classes and over different training steps.

(iii) We prove the effectiveness of our framework on several experimental scenarios outperforming competing approaches.

The rest of the chapter reflects the following organization: the proposed UDA method based on domain adversarial learning and self-training will be described in Section 4.2; experimental details and evaluation results will be instead reported respectively in Sections 4.3 and 4.4.

4.2 Adversarial Adaptation and Self-Training

In this section the complete architecture of the proposed approach is discussed. An overview of the method is given in Figure 4.1. We denote with G the semantic segmentation network that we want to adapt from supervised synthetic data to unlabeled real data. G takes the role of the generator in our adversarial setup. The optimization of the segmentation network is driven by the minimization of a multi-target objective involving four loss functions. In particular, to guide the adaptation of G to unlabeled real data we employ two discriminative networks and a self-training module.

Let us denote with $\mathbf{X}_n^s$ the generic n-th image in the source (synthetic) domain and with $\mathbf{Y}_n^s$ the corresponding ground-truth segmentation, while $\mathbf{X}_n^t$ is the n-th real-world sample (for which ground truth is not available during training). The first component of our approach, $\mathcal{L}_{G,0}$, is a standard cross-entropy loss working on labeled synthetic data.

4.2.1 Domain Adversarial Adaptation

The second and the third loss functions, minimized during the training of G, are relative to the adversarial training with the couple of discriminator networks. The first discriminator

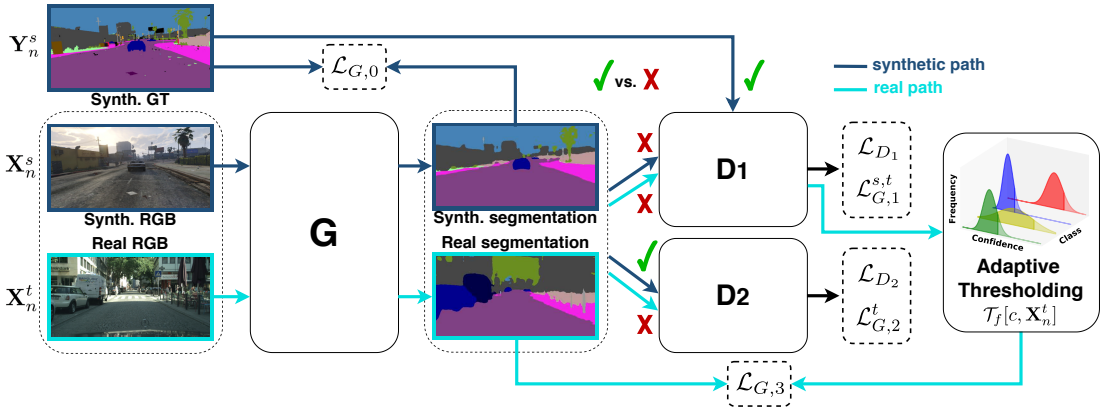


Figure 4.1: Architecture of the proposed approach. The semantic segmentation network G is trained with the combination of four losses: a supervised cross entropy on source data $\mathcal{L}_{G,0}$, a double adversarial framework $\mathcal{L}_{G,1}^{s,t}$ and $\mathcal{L}_{G,2}^{t}$, and a self-training module $\mathcal{L}_{G,3}$ with class-wise and time-varying adaptive thresholding mask T_f.

module D_1 is trained to distinguish between ground-truth and generated maps (the latter either coming from synthetic or real images). The peculiarity of this network is that it is a fully convolutional model and it produces a per-pixel confidence estimation, differently from traditional adversarial frameworks where the discriminator outputs a single binary value for the whole input image. The key idea is that the discriminative action provides a measure of discrepancy between the statistic of source annotations and both source and target prediction maps, and its optimization leads to an indirect yet effective cross-domain distribution alignment [53, 54, 92].

The discriminator network D_1 is made of a stack of 5 convolutional layers each using 4×4 kernels with a stride of 2 and Leaky ReLU activation function. The number of filters (from first to last layer) is 64, 64, 128, 128, 1 and the cascade is followed by a bilinear upsampling to match the original input image resolution. The network D_1 is trained to minimize the loss $\mathcal{L}_{D_1}$ that is a standard cross-entropy loss between D_1's output and the one-hot encoding indicating whether the input segmentation map is the ground truth (class 1) or it is produced by G (class 0), *i.e.*:

$$\mathcal{L}_{D_1} = - \sum_{p \in \mathbf{X}_n^{s,t}} \log(1 - D_1(G(\mathbf{X}_n^{s,t}))^{(p)}) + \log(D_1(\mathbf{Y}_n^s)^{(p)}), \tag{4.1}$$

where p is a generic pixel in the image. Meanwhile, G is forced to produce segmentation maps that resemble ground-truth annotations when inspected by D_1:

$$\mathcal{L}_{G,1}^{s,t} = - \sum_{p \in \mathbf{X}_n^{s,t}} \log(D_1(G(\mathbf{X}_n^{s,t}))^{(p)}). \tag{4.2}$$

The second discriminator, D_2, is trained to differentiate between segmentation maps coming from synthetic or real-world data. Differently from D_1, D_2 is always fed with the generator output, *i.e.*, with source or target images segmented by G, whereas no ground-truth information is employed. D_2 has the same structure as D_1 except that its number of channels has been reduced to 48, 48, 96, 96, 1, as its adaptation objective is complementary to the one of D_1 and requires less computational complexity to be accomplished. The adversarial loss for D_2 can be expressed as:

$$\mathcal{L}_{D_2} = - \sum_{p \in \mathbf{X}_n^{s,t}} \log(1 - D_2(G(\mathbf{X}_n^t))^{(p)}) + \log(D_2(G(\mathbf{X}_n^s))^{(p)}), \tag{4.3}$$

which is a standard cross-entropy loss similar to $\mathcal{L}_{D_1}$. Here the objective is formulated between D_2's output and the one-hot encoding marking with 0 and 1 the segmentation maps produced by G from respectively target and source images. The role of G in the second adversarial competition is to fool D_2, by computing sufficiently realistic target prediction maps resembling the source domain ones. Hence, similarly to Eq. (4.2), the adversarial loss for G related to D_2 is of the form:

$$\mathcal{L}_{G,2}^t = - \sum_{p \in \mathbf{X}_n^t} \log(D_2(G(\mathbf{X}_n^t))^{(p)}). \tag{4.4}$$

The second adversarial framework is specifically focused on the adaptation of target network representations, in that G is trained to produce target segmentation maps which are close to source ones from a statistical point of view. Thus it directly tackles the domain gap causing the performance drop on the target set. Source-target alignment also happens within the first adversarial scenario as a side effect. Indeed, G's output is forced to be distributed as ground-truth labels for both source and target inputs, but no actual cross-domain adaptation of network embeddings is directly performed.

4.2.2 Class-Adaptive Self-Training

The last module in our adaptation framework implements a self-training strategy. As shown by recent works [53, 54, 92], the output of the discriminator D_1 can be interpreted as a measure of confidence for predicted target pixels, since D_1 should identify target predictions that deviate from source ground-truth statistic. Hence, the output of D_1 can be exploited to select the reliable predictions in the target segmentation maps. The selected output samples are then converted into one-hot encoded data, which can be used as a self-taught ground-truth to train G directly on unlabeled target samples. The self-training objective is expressed

by the following loss:

$$\mathcal{L}_{G,3} = -\sum_{p \in \mathbf{X}_n^t} \sum_{c \in \mathcal{C}} \mathcal{M}_f^{(p)} \cdot W_c^s \cdot \hat{\mathbf{Y}}_n^{(p)}[c] \cdot \log\left(G(\mathbf{X}_n^t)^{(p)}[c]\right), \tag{4.5}$$

where $\hat{\mathbf{Y}}_n$ is the one-hot encoded ground truth derived from the per-class argmax of the generated probability map $G(\mathbf{X}_n^t)$, c is a specific class belonging to the set of possible classes $\mathcal{C}$ and W_c^s is a weighting function proportional to the class frequency on the source domain as introduced in [53, 54]. $\mathcal{M}_f^{(p)}[c, \mathbf{X}_n^t]$ (for ease of notation we drop its dependencies on class and training sample in all the equations) is the adaptive thresholding mask for the selection of the regions used for self-training, defined as:

$$\mathcal{M}_f^{(p)} = \begin{cases} 1 & \text{if } \left(D_1(G(\mathbf{X}_n^t))^{(p)} > \mathrm{T}_f[c, \mathbf{X}_n^t]\right) \wedge \left(\hat{\mathbf{Y}}_n^{(p)}[c] = 1\right) \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

Differently from previous works [53, 54, 79, 85], in which the confidence threshold was computed before-hand and kept fixed throughout the training, here we propose to have both a class-level and a training-stage adaptation of the threshold. Indeed, different classes typically have different confidence values which may also vary during training. The class-wise confidence threshold selection for a generic training step and a generic class c is formulated as:

$$\mathrm{T}_f[c, \mathbf{X}_n^t] = Q_f(D_1(G(\mathbf{X}_n^t)[c])), \tag{4.7}$$

where Q_f represents the f-th percentile. The best results are obtained with f in the range of $75 - 80\%$ to enable self-training only in regions with high confidence on target data. We compute T_f for every class at each training step by looking at network predictions on target data in the current batch. This makes the model adaptive both to the statistic of the various classes and to the different training phases. Finally, we can compute the overall loss function for G with a weighted average of the four individual losses, *i.e.*:

$$\mathcal{L}_{full} = \mathcal{L}_{G,0} + \lambda_1^{s,t}\mathcal{L}_{G,1}^{s,t} + \lambda_2^t\mathcal{L}_{G,2}^t + \lambda_3\mathcal{L}_{G,3}, \tag{4.8}$$

with weighting parameters $\lambda_1^{s,t}$, λ_2^t and λ_3 empirically set.

4.3 Experimental Setup

Datasets We evaluate our framework on some publicly available and widely used datasets. Supervised synthetic training is performed on the GTA5 [18] and SYNTHIA [19] datasets. The real-world datasets for unsupervised adaptation and for results evaluation are Cityscapes [14] and Mapillary [17]. The evaluation scenario is the same of competing approaches as [35, 47, 53, 54, 81] to allow for a fair comparison. We provide the segmentation network with images of 750×375px during training while testing is done at the original resolution.

Training Details The approach is agnostic to the deep learning architecture used for G: in principle any semantic segmentation network can be used, in our case we employ the well known Deeplab v2 model [11] with the ResNet-101 backbone.

The generator network G is trained as suggested in [11] using the Stochastic Gradient Descent (SGD) optimizer with momentum set to 0.9 and weight decay to $1\text{e-}4$. The discriminators D_1 and D_2 are trained using the Adam optimizer. Following [53], the learning rate employed for both G and D_1 starts from $1\text{e-}4$ and is decreased up to $1\text{e-}6$ by means of a polynomial decay with power 0.9. As for D_2, the base learning rate is set to $1\text{e-}4$ and $5\text{e-}4$ for respectively the SYNTHIA and GTA5 adaptation scenarios. Following the empirical validation and previous works [53, 54, 92], the weighting parameters are set as $\lambda_1^s = 1\text{e-}2$, $\lambda_1^t = 1\text{e-}3$ and $\lambda_2^t = 1\text{e-}2$, independently of the source and target datasets. The self-training parameter λ_3 is fixed to $5\text{e-}2$ and $1\text{e-}1$ for respectively SYNTHIA and GTA5 cases. This is indeed the most delicate parameter to tune, as the inferior realism of the SYNTHIA dataset suggests a more cautious usage of the self-training module to avoid flawed supervision that noisy target pseudo-labels may provide. The approach is instead more stable with respect to the other weighting parameters. We train the model for $20K$ iterations on a NVIDIA RTX 2080 Ti GPU. The longest training inside this work, *i.e.*, the one with all the loss components enabled, takes about 6 hours to complete.

4.4 Experimental Results

4.4.1 Evaluation on the Cityscapes Dataset

The first scenario we consider for evaluation comprises the adaptation to the Cityscapes dataset from both SYNTHIA and GTA5. As done by competing approaches [35, 36, 48], the numerical performance is expressed in terms of mean Intersection over Union (mIoU) between predicted maps and ground-truth labels over the Cityscapes validation set. The per-

Table 4.1: Per-class and mean IoU on the four considered UDA scenarios, *i.e.*, adaptation from GTA5 or SYNTHIA source datasets to the Cityscapes or Mapillary target counterparts. Results are reported on validation sets of the target domains. The highest values have been highlighted in bold.

			Method	Road	Sidewalk	Building	Wall	Fence	Pole	T. Light	T. Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorbike	Bicycle	mIoU
a)	To Cityscapes	From GTA5	Supervised ($\mathcal{L}_{G,0}$)	49.3	24.4	56.4	6.5	**19.6**	25.6	**23.6**	10.1	82.7	28.5	69.9	**55.5**	4.9	80.9	18.0	33.0	1.2	15.1	0.1	31.9
			Hoffman *et al.* [35]	70.4	32.4	62.1	14.9	5.4	10.9	14.2	2.7	79.2	21.3	64.6	44.1	4.2	70.4	8.0	7.3	0.0	3.5	0.0	27.1
			Hung *et al.* [92]	**81.7**	0.3	68.4	4.5	2.7	8.5	0.6	0.0	82.7	21.5	67.9	40.0	3.3	80.7	**34.2**	**45.9**	0.2	8.7	0.0	29.0
			Zhang *et al.* [81]	74.9	22.0	**71.7**	6.0	11.9	8.4	16.3	**11.1**	75.7	13.3	66.5	38.0	**9.3**	55.2	18.8	18.9	0.0	**16.8**	**14.6**	28.9
			Biasetton *et al.* [53]	54.9	23.8	50.9	16.2	11.2	20.0	3.2	0.0	79.7	31.6	64.9	52.5	7.9	79.5	27.2	41.8	0.5	10.7	1.3	30.4
			Michieli *et al.* [54]	81.0	19.6	65.8	**20.7**	2.9	20.9	6.6	0.2	82.4	33.0	68.2	54.9	6.2	80.3	28.1	41.6	**2.4**	8.5	0.0	33.3
			Ours [238]	77.7	**35.9**	67.2	18.9	12.1	**26.2**	15.9	5.9	**83.7**	**33.3**	**72.7**	53.9	4.2	**82.6**	21.5	41.1	0.1	13.9	0.0	**35.1**
b)		From SYNTHIA	Supervised ($\mathcal{L}_{G,0}$)	17.9	24.2	38.6	**5.0**	0.0	28.7	0.0	4.5	79.3	-	80.8	**54.0**	**8.9**	75.7	-	35.4	-	4.2	3.9	28.8
			Hoffman *et al.* [35]	11.5	19.6	30.8	4.4	0.0	20.3	0.1	**11.7**	42.3	-	68.7	51.2	3.8	54.0	-	3.2	-	0.2	0.6	20.1
			Hung *et al.* [92]	72.5	0.0	63.8	0.0	0.0	16.3	0.0	0.5	**84.7**	-	76.9	45.3	1.5	77.6	-	31.3	-	0.0	0.1	29.4
			Zhang *et al.* [81]	65.2	26.1	74.9	0.1	**0.5**	10.7	**3.7**	3.0	76.1	-	70.6	47.1	8.2	43.2	-	20.7	-	0.7	**13.1**	29.0
			Biasetton *et al.* [53]	78.4	0.1	73.2	0.0	0.0	16.9	0.0	0.2	84.3	-	78.8	46.0	0.3	74.9	-	30.8	-	0.0	0.1	30.2
			Michieli *et al.* [54]	**80.7**	0.3	**75.0**	0.0	0.0	19.5	0.0	0.4	84.0	-	79.4	46.6	0.8	80.8	-	32.8	-	0.5	0.5	31.3
			Ours [238]	72.0	**26.6**	66.1	1.8	0.0	**30.2**	0.0	4.3	81.4	-	**82.2**	51.7	4.0	**82.2**	-	**37.9**	-	**7.7**	5.9	**34.6**
c)	To Mapillary	From GTA5	Supervised ($\mathcal{L}_{G,0}$)	69.8	31.8	58.8	14.6	22.3	28.3	**31.8**	**28.8**	70.0	24.4	72.4	**60.4**	16.8	80.6	36.6	34.3	**10.2**	**26.2**	**0.2**	37.8
			Hung *et al.* [92]	78.2	29.7	68.7	10.0	6.7	17.5	0.0	0.0	76.4	35.2	**95.6**	53.8	13.8	77.5	34.3	30.2	5.0	21.8	0.0	34.4
			Biasetton *et al.* [53]	71.4	25.0	62.0	20.4	17.6	26.8	5.9	0.8	64.6	24.6	86.5	58.3	14.7	80.0	39.3	42.2	5.5	22.3	0.1	35.2
			Michieli *et al.* [54]	79.9	28.0	73.4	**23.0**	29.5	20.9	1.1	0.0	**79.5**	**39.6**	95.0	57.6	9.0	80.6	41.5	40.1	7.4	24.8	0.1	38.5
			Ours [238]	**80.0**	**43.3**	**75.4**	19.4	**29.7**	**29.6**	23.3	16.2	78.5	33.5	93.7	59.0	**20.3**	**82.2**	**44.5**	**43.4**	2.5	22.1	0.0	**41.9**
d)		From SYNTHIA	Supervised ($\mathcal{L}_{G,0}$)	25.4	22.0	56.4	**6.9**	**0.1**	29.4	0.0	**2.8**	72.8	-	92.1	53.7	**16.1**	75.1	-	**30.8**	-	8.6	5.8	31.1
			Hung *et al.* [92]	36.8	20.1	53.9	0.0	0.0	23.7	0.0	0.0	73.9	-	**95.6**	43.4	0.1	64.6	-	19.0	-	0.4	0.5	27.0
			Biasetton *et al.* [53]	16.4	19.1	42.2	2.7	0.0	**33.1**	0.0	1.3	76.5	-	88.0	50.4	10.9	69.9	-	25.5	-	6.1	9.2	28.2
			Michieli *et al.* [54]	57.6	18.3	62.1	0.4	0.0	23.7	0.0	0.0	**79.4**	-	94.8	52.4	9.2	74.2	-	28.3	-	4.0	6.9	32.0
			Ours [238]	**59.0**	**28.4**	**68.6**	0.7	0.0	29.8	0.0	1.8	77.5	-	94.9	**54.6**	12.6	**76.8**	-	28.0	-	**11.2**	**14.7**	**34.9**

class mIoU results of the evaluations are displayed in Table 4.1a) and 4.1b). We denote as *supervised* the naïve approach relying only on source supervision and no form of adaptation, while the numerical performance of our method as a whole is reported in the last row of each section.

Using GTA5 as source domain, the simple supervised approach achieves a final mIoU of 31.8% on the target dataset. The introduction of the multi-domain adversarial learning scheme and adaptive self-training module leads to a performance increment of 3.3%, boosting the mIoU up to 35.1%. Moreover, the improvement is shared by the majority of the semantic classes. Some categories characterized by semantic similarity between them (such as *road*, *sidewalk* and *terrain*) or with appearance discrepancy across source and target domains (such as *car*, *truck* and *bus*) seem to highly benefit from the adaptation, proving that our method succeeded in bridging the domain gap. In Table 4.1 we also report results achieved by some competing approaches. It can be noticed that our strategy outperforms all of them, including those relying on simpler self-training and adversarial learning schemes

(*i.e.*, [53, 54, 92]), demonstrating the efficacy of the multi-domain discrimination and of the adaptive thresholding techniques we introduced. In Figure 4.2a) we show some sample segmented images from the Cityscapes validation set. We can appreciate that our approach leads to a more precise detection of several semantic entities found in the input image. For example, the *road* and *sidewalk* classes are subject to a more accurate recognition, which supports the numerical results.

Section (b) of Table 4.1 reports the results of the adaptation from the SYNTHIA dataset. As for the GTA5 case, our adaptation strategy represents a considerable improvement over the supervised training on the source dataset. The mIoU achieved without adaptation is pushed from 28.8% up to 34.6% by our framework. The increment of almost 6% is even higher than the one achieved with the more realistic GTA5 dataset as source domain, and this proves the effectiveness of the adaptation modules we developed also in a challenging environment with a larger statistical gap across domains. For example, while texture discrepancy between Cityscapes and SYNTHIA causes the road on real-world scenes to be hardly recognized by the segmentation network in lack of adaptation, our strategy successfully provides the predictor with road detection capabilities. Figure 4.2a) includes some qualitative results. Again, we can observe the improved semantic understanding and detection accuracy on classes such as *road* and *sidewalk* with respect to the baseline, as well as with respect to a competing approach based on adversarial and self-training techniques [53].

4.4.2 Evaluation on the Mapillary dataset

We evaluate our approach also on the Mapillary dataset. We start by using the GTA5 dataset for the supervised training as before: the results are shown in Table 4.1c). With no adaptation to the Mapillary dataset the network achieves a mIoU of 37.8%. Thanks to the multiple domain discriminators and to the adaptive self-training techniques our framework is able to reach a mIoU of 41.9%, significantly outperforming all the compared methodologies. We can notice that the improvement with respect to the baseline approach is consistently distributed among the semantic classes and it is particularly evident on the *road* or *building* ones. Qualitative results are shown in the first two rows of Figure 4.2b), where we can verify that most of the noise present in the supervised training and in [53] is filtered out by the proposed framework. In particular, the *vegetation* and *sidewalk* categories highly benefit from the domain adaptation with class-wise and time-variable confidence threshold selection.

Furthermore, we can appreciate that, also on the Mapillary dataset, the accuracy is lower when the SYNTHIA dataset is used for supervised training, leading to a mIoU of 31.3% only. As already noticed from the evaluations on Cityscapes, some classes (*i.e.*, *road*, *sidewalk* and *building*) have a low accuracy due to the poor texture representation and vastly

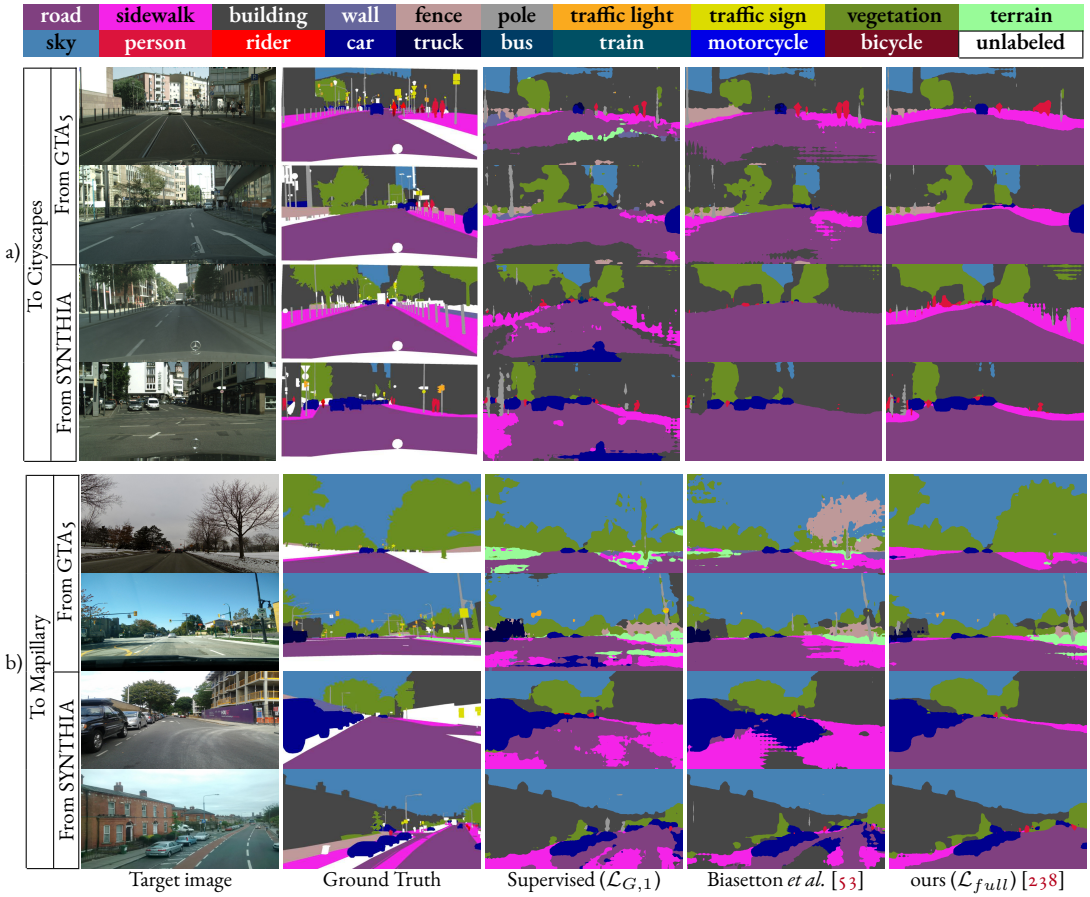


Figure 4.2: Semantic segmentation of sample scenes from the Cityscapes (a) and Mapillary (b) validation sets. Four UDA scenarios are considered, *i.e.*, adaptation from GTA5 or SYNTHIA source datasets to the Cityscapes or Mapillary target counterparts.

profit by the adaptation to the target domain. The complete framework increases the final mIoU to 34.9% with an improvement of 3.8%, consistent with the previous experiments made across different datasets. Remarkable are the percentage gains in the aforementioned classes: for example, the *road* class more than doubles its accuracy and *sidewalk*'s IoU grows by 12.2%. The visual results are reported in the last two rows of Figure 4.2b). Here, for example, we can appreciate that the proposed approach is the only one to achieve an accurate and reliable recognition of *road* and *cars* classes on the shown images, which confirms our previous analysis.

4.4.3 Ablation Study

In this section we present an accurate investigation of the effectiveness of the various modules of the proposed framework. For this study we consider the performances on the Mapillary dataset when adapting from GTA5. We start by evaluating the individual impact of each module: the performance analysis is shown in Table 4.2. Let us recall that the baseline architecture, *i.e.*, the Deeplab-v2 network trained on synthetic data only, achieves a mIoU of 37.8% on real data. Then, we analyze the remaining modules by removing one component at a time (row 2 to 6). We can appreciate that all the components bring a significant contribution to the final mIoU, which in the full version of the approach where all of them are enabled is 41.9%. The impact of $\mathcal{L}^s_{G,1}$, $\mathcal{L}^t_{G,1}$ and $\mathcal{L}_{G,2}$ is clear by looking at Table 4.2: without each of them the accuracy decreases with respect to the complete framework but remains higher than the source supervised case. In particular, we performed a more detailed analysis of the self-training module. Having no self-training leads to 41.1% of mIoU, while having self-training done on all pixels (without thresholding with T_f) or using a fixed confidence threshold (*e.g.*, setting a constant value of $\mathrm{T}_f = 0.2$ as in [53]) leads to 40.6% and 40.9%, respectively. These results show that self-training is not effective if the reliable pixels are not accurately selected, *e.g.*, if performed immutable over the classes and the training steps. On the other side we found that self-training is effective with confidence thresholds variable over classes and over training time.

We also analyzed the behavior of the per-class time-varying confidence values: they are shown in Figure 4.3 for the GTA5 to Mapillary scenario. Here, we can appreciate how the confidence thresholds vary over the training time and typically converge to a value which may be significantly different among the various classes. While previous works [53, 54] fixed the threshold to 0.2, here we can see how the desired value is variable and ranges between 0.04 and 0.4. Highly confident classes typically have high confidence threshold, in order to propagate back very reliable predictions through self-training as for instance *road*, *sky* and *building*. However, notice that *train* and *bike* have high confidence threshold values only because they

Table 4.2: Ablation results on the impact of loss components (GTA5 to Mapillary adaptation).

$\mathcal{L}_{G,0}$	$\mathcal{L}^{s}_{G,1}$	$\mathcal{L}^{t}_{G,1}$	$\mathcal{L}^{t}_{G,2}$	$\mathcal{L}_{G,3}$	T_f	mIoU
✓						37.8
✓		✓	✓	✓	✓	39.9
✓	✓		✓	✓	✓	40.3
✓	✓	✓		✓	✓	40.7
✓	✓	✓	✓			41.1
✓	✓	✓	✓	✓		40.6
✓	✓	✓	✓	✓	fix 0.2	40.9
✓	✓	✓	✓	✓	✓	41.9

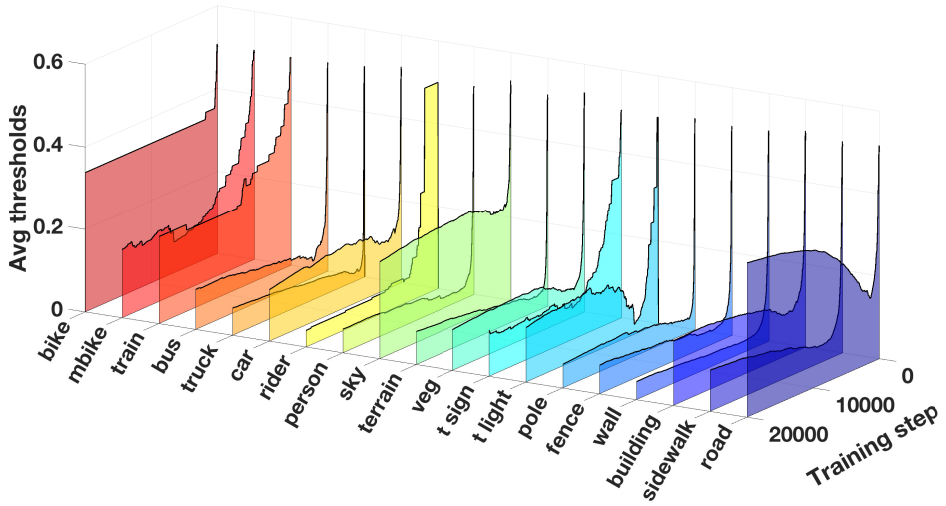


Figure 4.3: Time average over the initial to current step interval of per-class confidence thresholds for different classes and at different training steps (GTA5 to Mapillary adaptation).

are predicted too rarely to be useful for confidence assessment (*i.e.*, the thresholds are piecewise constant functions), hence represent failure modes of the estimate. More challenging classes such as *wall* and *fence*, instead, are characterized by a much lower confidence value. The most important aspect, however, is to verify that our framework can adapt both to the properties of different classes and to the changes in the network behavior during the training procedure. Additionally, we can compute the mean threshold value averaged over all the classes at the end of the training phase and compare it with previous works [53, 54]. The mean is 0.13 in the considered scenario and 0.19 in the adaptation from GTA5 to Cityscapes, consistently with the value of 0.20 used in previous works [53, 54].

5

Class Conditional Feature Adaptation

5.1 Introduction

As thoroughly introduced in Chapter 2, three main levels on which adaptation may occur can be identified [234]: namely, at the input, features or output stages. While in Chapters 3 and 4 we posed the focus on the input and output space adaptations, now we shift our attention to an intermediate level of representation, which usually captures richer and more complex semantic clues. Deep networks, in fact, typically solve complex tasks by building some compact latent representations of the inputs, which are representative of the classifier output. These internal representations are extremely meaningful for the subsequent decision process [94, 95]. Nevertheless, current UDA approaches for semantic segmentation hardly operate at this level due to the high dimensionality of the latent space. Among those that instead focus on feature adaptation, a popular solution has become to bridge the domain gap at an intermediate representation level by means of adversarial techniques [35, 40, 234]. The major drawback of these kind of approaches is that they usually perform a semantically unaware alignment, as they neglect the underlying class-conditional data distribution. Furthermore, the training process they entail is typically unstable, and it generally requires the computation time to reach convergence to be greatly increased.

Differently from those feature-level techniques relying on adversarial schemes, the adaptation strategies of feature representations that we propose are simple and do not require complex adversarial learning frameworks, since they demand only a slight increase of computation time with respect to the sole supervised learning. Those strategies work on the less-explored feature-level: our aim is to reduce the performance discrepancy by employing latent

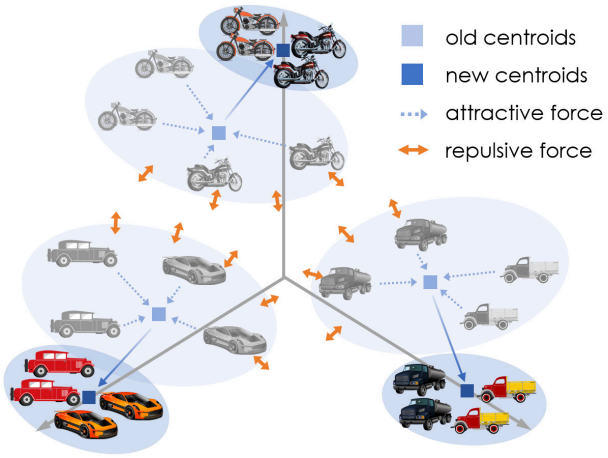


Figure 5.1: The proposed domain adaptation scheme aims to enforce latent space regularization across domains. Feature clustering objectives are proposed, along with orthogonality, sparsity and norm alignment constraints to achieve this goal. By doing so, features in the previous step (in light gray) are driven to new locations (colored) where features of the same class are clustered, while features of distinct classes are pushed away. To further improve the adaptation performance, features of distinct classes are forced to be orthogonal and sparse, and their norm to be consistent across domains.

space-shaping objectives between source and target domains. The main idea is depicted in Figure 5.1: we devise a domain adaptation technique specifically targeted to guide the latent space organization.

The initial framework we devise (named OCE [239]) is driven by 3 main components, by which we aim to provide **O**rthogonal and **C**lustered **E**mbeddings shared by source and target domains. The first is a feature clustering mechanism to group together features of the same class, while pushing apart features belonging to different classes. This constraint, which works simultaneously on both domains, is similar in spirit to the recent progresses in contrastive learning for classification problems [96]; however, it has been developed aiming at a simpler computation, as the number of features per image is significantly larger than in the classification task. The second is a novel orthogonality requirement for the feature space (which targets single pixel-level feature representation), aiming at reducing the cross-talk between features belonging to different classes. Finally, a sparsity constraint is added to reduce the number of active channels for each feature vector: our aim is to enforce the capability of deep learning architectures to learn a compact representation of the scene. The combined effect of these modules allows to regularize the structure of the latent space in order to encompass the source and target domains in a shared representation.

In a second step, we extended the previous work to improve domain adaptation by **L**atent **S**pace **R**egularization, which we called LSR [236, 240]. Once more, a clustering-based objective forces the feature vectors of each class to be closer to the corresponding prototype cen-

troids. While based on source supervision for prototype estimation, its action is delivered to both source and target representations to achieve class-conditional domain alignment. Differently from the previous work, an additional component enforces the perpendicularity directly on class prototypes, thus assembling features into well-distanced class clusters and, at the same time, indirectly promoting disjoint activation sets between semantic categories. Thus, now feature clusters are spaced apart in terms of angular distance. Finally, we account for the fact that, as noticed in [97], feature vectors computed from target domain samples tend to have smaller norms than source domain ones. This latter claim is due to domain-specific features, which the networks rely on to solve the source-supervised classification. Yet, those features may be missing in the target domain and, therefore, may lead to a weakened response of neuron activations in target latent representations. To address this issue, we introduce a regularization objective that promotes uniform vector norms across source and target representations, while jointly inducing progressively increased norm values. Furthermore, the inter-class norm alignment has shown to remove distribution biases towards the most frequent classes, whose higher classification confidence is typically accompanied by bigger feature norms. Since the proposed techniques require to set a strong relationship between predicted segmentation maps and feature representations, we additionally develop a novel strategy to propagate semantic information from the labels to the lower resolution feature space.

Summarizing, the main contributions brought in this chapter are:

(i) We extend class-conditional feature clustering to semantic segmentation, similarly to what achieved by contrastive learning.

(ii) We introduce orthogonality, sparsity and feature norm-alignment objectives to force a regular structure of the embedding space, which is jointly enforced on both source and target domains to reach distribution alignment of latent representations.

(iii) We achieve very competitive results on feature-level adaptation on synthetic-to-real domain adaptation benchmarks via lightweight framework, which we show could be easily integrated with other adaptation strategies.

The reminder of the chapter is structured as follows: Section 5.2 will provide a formal definition of the UDA setting under feature adaptation perspective; then, the proposed OCE and LSR UDA frameworks based on class-conditional feature-level regularization and cross-domain adaptation will be introduced in Sections 5.3 and 5.4. finally, experimental details and evaluation results, along with extensive ablation studies, will be presented for both methods in Sections 5.5 and 5.6.

5.2 Problem Formulation

Formally, we denote the input image space as $\mathcal{X} \subset \mathbb{R}^{H \times W \times 3}$ and the associated output label space as $\mathcal{Y} \subset \mathcal{C}^{H \times W}$, where H and W represent the spatial dimensions and $\mathcal{C}$ the set of classes. Given a first training set $\mathcal{T}^s = \{(\mathbf{X}_n^s, \mathbf{Y}_n^s)\}_{n=1}^{N_s}$, where labeled samples $(\mathbf{X}_n^s, \mathbf{Y}_n^s) \in \mathcal{X}_S \times \mathcal{Y}^s$ originate from a supervised source domain, together with a second set of unlabeled input samples $\mathcal{T}^t = \{\mathbf{X}_n^t\}_{n=1}^{N_t}$, from a target domain ($\mathbf{X}_n^t \in \mathcal{X}_T$), our goal is to transfer knowledge on the segmentation task learned on the source domain to the unsupervised target domain (*i.e.*, without any label on the target set). Superscripts s and t specify the domain: source and target, respectively. In the considered UDA setting, we are provided with plenty of samples $\mathbf{X}_n^s \in \mathbb{R}^{H \times W \times 3}$ from a source dataset, in conjunction with their semantic maps $\mathbf{Y}_n^s \in \mathbb{R}^{H \times W}$. Those semantic maps contain at each spatial location a ground-truth index belonging to the set of possible classes $\mathcal{C}$, which denotes the semantic category of the associated pixel. Concurrently, we have at our disposal target training samples $\mathbf{X}_n^t \in \mathbb{R}^{H \times W \times 3}$ with no label maps (we allow only the availability of a small amount of target labels for validation and testing purposes). Despite sharing similar high-level semantic content, the source and training samples are distributed differently, preventing a source-based model to achieve a satisfying prediction accuracy on target data without adaptation.

We assume that the segmentation network $M = M^d \circ M^e$ is based on an encoder-decoder architecture (as most recent approaches for semantic segmentation), *i.e.*, made by the concatenation of two logical blocks: the encoder network M^e, consisting of the feature extractor, and a decoder network M^d, which is the actual classifier producing the segmentation map. Moreover, we call $M^e(\mathbf{X}_n) = \mathbf{F}_n \in \mathbb{R}_{0+}^{H' \times W' \times K}$ the features extracted from a generic input image $\mathbf{X}_n$, where K denotes the number of channels and $H' \times W'$ denotes the low-dimensional latent spatial resolution. Notice that the proposed method is agnostic to the employed deep learning model, except for the assumption of an encoder-encoder structure and of positive feature values as provided by ReLU activations that are typically placed at the encoder output (as almost all the current state-of-the-art approaches for semantic segmentation).

Given the structure of encoder-decoder based convolutional segmentation networks, we can assume that each class is mapped to a reference representation in the latent space, that should be as invariant as possible to the domain shift. The techniques that will be introduced in Sections 5.3 and 5.4 try to enforce this by comparing the extracted features with some *prototypes* for the various classes.

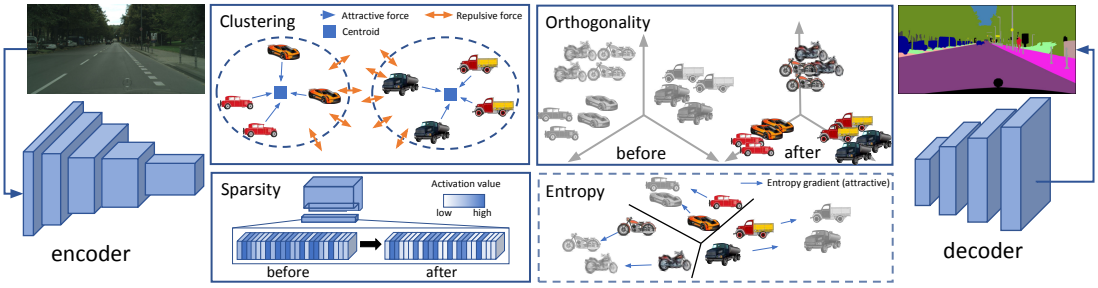


Figure 5.2: Overview of the proposed OCE approach. Features after supervised training on the source domain are represented in light gray, while features of the current step are colored. A set of techniques is employed to better shape the latent feature space spanned by the encoder. Features are clustered and the clusters are forced to be disjoint. At the same time, features belonging to different classes are forced to be orthogonal with respect to each other. Additionally, features are forced to be sparse and an entropy minimization loss could also be added to guide target samples far from the decision boundaries.

5.3 Orthogonal and Clustered Embeddings

In this section, we provide an in depth description of the core modules of the first proposed OCE method [239], *i.e.*, based on clustered and orthogonal embeddings. Our approach leverages a clustering objective applied over the individual feature representations, with novel orthogonality and sparsity constraints. Specifically, inter and intra class alignments are enforced by grouping together features of the same semantic class, while simultaneously pushing away those of different categories. By enforcing the clustering objective on both source and target representations, we drive the model towards feature-level domain alignment. We further regularize the distribution of latent representations by the joint application of an orthogonality and a sparsity losses. The orthogonality module has a two-fold objective: first, it forces feature vectors of kindred semantic connotations to activate the same channels, while turning off the remaining ones; second, it constrains feature vectors of dissimilar semantic connotations to activate different channels, *i.e.*, with no overlap, to reduce cross interference. The sparsity objective further encourages a lower volume of active feature channels from latent representations, *i.e.*, it concentrates the energy of the features on few dimensions.

A graphical outline of the approach with all its components is shown in Figure 5.2: the training objective is given by the combination of the standard supervised loss with the proposed adaptation modules, *i.e.*, it is computed as:

$$\mathcal{L}'_{tot} = \mathcal{L}_{ce} + \lambda_{cl} \cdot \mathcal{L}_{cl} + \lambda_{or} \cdot \mathcal{L}_{or} + \lambda_{sp} \cdot \mathcal{L}_{sp}, \tag{5.1}$$

where $\mathcal{L}_{ce}$ is the standard supervised cross entropy loss. The other components will be detailed in the following sections: the main clustering objective ($\mathcal{L}_{cl}$) is introduced in Section

5.3.1. The orthogonality constraint ($\mathcal{L}_{or}$) is discussed in Section 5.3.2 and finally the sparsity constraint ($\mathcal{L}_{sp}$) is detailed in Section 5.3.3. The λ parameters balance the multiple losses and are experimentally chosen using a validation set.
In addition, we further integrate the proposed adaptation method with an off-the-shelf entropy-minimization like objective ($\mathcal{L}_{em}$), to provide an extra regularizing action over the segmentation feature space and ultimately achieve an improved performance in some evaluation scenarios. In particular, we adopt the simple, yet effective, maximum squares objective of [1], in its *image-wise class-balanced* version. Hence, we can define the ultimate training objective comprising the entropy module as:

$$\mathcal{L}_{tot} = \mathcal{L}'_{tot} + \lambda_{em} \cdot \mathcal{L}_{em}. \tag{5.2}$$

5.3.1 Discriminative Clustering

To bridge the domain gap between the source and target datasets we operate at the feature level. The discrepancy of input statistics across domains is reflected into a shift of feature distribution in the latent space spanned by the feature extractor. This ultimately may cause the source-trained classifier to draw decision boundaries crossing high density regions of the target latent space [55], since it is inherently unaware of the target semantic modes extracted from unlabeled target data. Thus, the classification performance over the target domain is strongly degraded when compared to the upper bound of the source prediction accuracy.

We cope with this performance degradation by resorting to a clustering module, that serves as constraint towards class-conditional feature alignment between domains. Given a batch of source ($\mathbf{X}^s_n$) and target ($\mathbf{X}^t_n$) training images (for ease of notation we pick a single image per domain), we first extract the feature tensors $\mathbf{F}^s_n = M^e(\mathbf{X}^s_n)$ and $\mathbf{F}^t_n = M^e(\mathbf{X}^t_n)$, along with the computed output segmentation maps $\hat{\mathbf{Y}}^s_n = M(\mathbf{X}^s_n)$ and $\hat{\mathbf{Y}}^t_n = M(\mathbf{X}^t_n)$. The clustering loss is then computed as:

$$\mathcal{L}_{cl} = \frac{1}{|\mathbf{F}^{s,t}_n|} \sum_{\substack{\mathbf{f}_i \in \mathbf{F}^{s,t}_n \\ \hat{y}_i \in \mathbf{S}^{s,t}_n}} d(\mathbf{f}_i, \boldsymbol{\pi}_{\hat{y}_i}) - \frac{1}{|\mathcal{C}|(|\mathcal{C}|-1)} \sum_{j \in \mathcal{C}} \sum_{\substack{k \in \mathcal{C} \\ k \neq j}}, d(\boldsymbol{\pi}_j, \boldsymbol{\pi}_k) \tag{5.3}$$

where $\mathbf{f}_i$ is an individual feature vector corresponding to a single spatial location from either source or target domain and $\hat{y}_i$ is the corresponding predicted class (to compute $\hat{y}_i$ the segmentation map $\hat{\mathbf{Y}}^{s,t}_n$ is downsampled to match the feature tensor spatial dimensions). The function $d(\cdot)$ represents a generic distance measure, that we set to the $L1$ norm (we also tried the $L2$ norm but it yielded lower results). Finally, $\boldsymbol{\pi}_j$ denotes the centroid of semantic class

$j \in \mathcal{C}$ computed according to the standard formula:

$$\boldsymbol{\pi}_j = \frac{\sum_{\mathbf{f}_i} \sum_{\hat{y}_i} \delta_{j,\hat{y}_i} \mathbf{f}_i}{\sum_{\hat{y}_i} \delta_{j,\hat{y}_i}}, \quad j \in \mathcal{C} \tag{5.4}$$

where $\delta_{j,\hat{y}_i}$ is equal to 1 if $\hat{y}_i = j$, and to 0 otherwise.

The clustering objective is composed of two terms, the first measures how close features are from their respective centroids and the second how spaced out clusters corresponding to different semantic classes are. Hence, the effect provided by the loss minimization is twofold: firstly, feature vectors from the same class but different domains are tightened around class feature centroids; secondly, features from separate classes are subject to a repulsive force applied to feature centroids, moving them apart.

5.3.2 Orthogonality of Individual Feature Representations

As opposed to previous works on clustering-based adaptation methods for image classification, in semantic segmentation additional complexity is brought by the dense structured classification. To this end, we first introduce an orthogonality constraint in the form of a training objective. More precisely, feature vectors from either domains, but of different semantic classes according to the network predictions, are forced to be orthogonal, meaning that their scalar product should be small. On the contrary, features sharing semantic classification should carry high similarity, *i.e.*, large scalar product. Yet, feature tensors associated to training samples enclose thousands of feature vectors to cover the entire spatial extent of the scene and to reach pixel-level classification. Thus, since measuring pair-wise similarities requires a significant computational effort, we calculate the scalar product between each feature vector and every class centroid $\boldsymbol{\pi}_j$ (centroids are computed using Eq. (5.4)). Inspired by [28, 98], we devise the orthogonality objective as an entropy minimization loss that forces each feature to be orthogonal with respect to all the centroids but one:

$$\mathcal{L}_{or} = - \sum_{\mathbf{f}_i \in F(\mathbf{X}_n^{s,t})} \sum_{j \in \mathcal{C}} p_j(\mathbf{f}_i) \log p_j(\mathbf{f}_i), \tag{5.5}$$

where $\{p_j(\mathbf{f}_i)\}$ denotes a probability distribution derived as:

$$p_j(\mathbf{f}_i) = \frac{e^{\langle \mathbf{f}_i, \boldsymbol{\pi}_j \rangle}}{\sum_{k \in \mathcal{C}} e^{\langle \mathbf{f}_i, \boldsymbol{\pi}_k \rangle}}, \quad j \in \mathcal{C} \tag{5.6}$$

The loss minimization forces a peaked distribution of the probabilities $\{p_j(\mathbf{f}_i)\}$, promoting the orthogonality property as described above, since each feature vector is compelled to carry a high similarity score with a single class centroid. The overall effect of the orthogonality objective is to promote a regularized feature distribution, which should ultimately boost the clustering efficacy in performing domain feature alignment.

5.3.3 Feature Sparsity

To strengthen the regularizing effect brought by the orthogonality constraint, we introduce a further training objective to better shape class-wise feature structures inside the latent space. In particular, we propose a sparsity loss, with the intent of decreasing the number of active feature channels of latent vectors. The objective is defined as follows:

$$\mathcal{L}_{sp} = -\sum_{i \in \mathcal{C}} ||\tilde{\boldsymbol{\pi}}_i - \boldsymbol{\rho}||_2^2, \tag{5.7}$$

where $\tilde{\boldsymbol{\pi}}_i$ stands for the normalized centroid $\boldsymbol{\pi}_i$ in $[0, 1]^D$ and D denotes the number of feature maps in the encoder output. We also empirically set $\boldsymbol{\rho} = [0.5]^D$. It can be noted that the sparsifying action is delivered on class centroids, thus applying an indirect, yet homogeneous, influence over all feature vectors from the same semantic category. The result is a semantically-consistent suppression of weak activations, while rather active ones are jointly raised.

While the orthogonality objective aims at promoting different sets of activations on feature vectors from separate semantic classes, the sparsity loss seeks to narrow those sets to a limited amount of units. Again, the goal is to ease the clustering loss task in creating tight and well distanced aggregations of features of similar semantic connotation from either source and target domains, by providing an improved regularity to the class-conditional semantic structures inside the feature space.

5.4 Latent Space Regularization

In this section, we provide a detailed description of the proposed LSR approach [240], which is based on the idea of aiding the standard cross-entropy loss with additional components to enforce the regularization of the latent space. While the source supervised cross-entropy provides task discriminativeness to the model, the additional objectives jointly imposed on source and target representations drive towards feature-level domain invariance, ultimately reducing domain bias. In particular, we add three feature-space shaping constraints to the

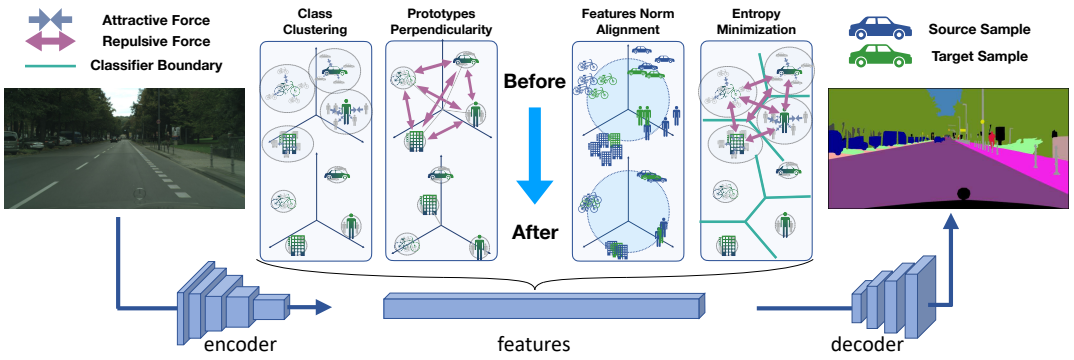


Figure 5.3: Visual summary of the proposed LSR strategy and of the effect of its application on the feature space. The three proposed space shaping constraints are from left to right: Class Clustering (5.4.1), Prototypes Perpendicularity (5.4.2), Norm Alignment and Enhancement (5.4.3). Furthermore, we apply entropy minimization [1].

standard source-supervised cross-entropy loss $\mathcal{L}_{ce}^{s}$, whose combined effect can be mathematically expressed by:

$$\mathcal{L} = \mathcal{L}_{ce}^{s} + \hat{\lambda}_{cl}^{s,t} \cdot \hat{\mathcal{L}}_{cl}^{s,t} + \lambda_{pp}^{s} \cdot \mathcal{L}_{pp}^{s} + \lambda_{na}^{s,t} \cdot \mathcal{L}_{na}^{s,t}. \tag{5.8}$$

Here, $\hat{\mathcal{L}}_{cl}$ is the clustering loss on latent representations (which is of a different from with respect to OCE's one, Section 5.3.1) (Section 5.4.1), $\mathcal{L}_{pp}$ is the perpendicularity of class prototypes loss (Section 5.4.2) and $\mathcal{L}_{na}$ is the norm alignment and enhancement loss (Section 5.4.3). For an improved performance and to show that our approach can be applied on top of existing methods, we also extended our objective with the entropy minimization strategy proposed in [1], leading to $\mathcal{L}^{+} = \mathcal{L} + \lambda_{em} \cdot \mathcal{L}_{em}$.
An overview of the complete approach is reported in Figure 5.3.

5.4.1 Clustering of Latent Representations

The domain shift between source and target data is reflected into a discrepancy in distribution of latent representations from separate domains. Moreover, as the lack of target supervision inherently leads to a bias towards the source domain, it is very likely for the classifier to trace decision boundaries tight around source embeddings, regardless of the disposition of unlabeled target instances. Thus, the misalignment of class-conditional feature statistic inevitably leads the model towards incorrect classification over target representations, in turn degrading the segmentation accuracy on the target domain.

To cope with this issue, we start by introducing a clustering objective over the latent space, in order to achieve class-conditional alignment of feature distribution. Once more we exploit class prototypical feature representations to drive the clusterization. Nonetheless, we mod-

ify the clustering objective proposed in OCE [239] (Section 5.3.1), by focusing solely on grouping feature representations of the same semantics across domains in tight clusters. In addition, we improve the dense feature labeling and prototype computation mechanisms employed in the OCE method 5.3. To this end, we propose an enhanced downsampling method to convert full resolution segmentation maps to feature-level spatial dimensions, along with a more advanced and stable prototype estimation, which will be detailed in the next sections.

Histogram-Aware Downsampling

Since the spatial information of an image is mostly preserved while its content travels through an encoder-decoder network, we can infer a strict relationship between any feature vector and the semantic labeling of the corresponding image region. Therefore, the first step of the extraction process is to identify a way to propagate the labeling information to latent representations (decimation), preserving the semantic content of the image region (window) associated to each feature vector. Otherwise, the generation of erroneous associations would significantly impair the estimation objective. For this task, we design a non-linear pooling function: instead of computing a simple subsampling (*e.g.*, nearest neighbor), we compute a frequency histogram over the labels of all the pixels in each window. Such histograms are then used to select appropriate classification labels for the downsampled windows, producing feature-level label maps $\{\mathbf{I}_n^{s,t}\}_{n=1}^{N_{s,t}}$. Specifically, the choice is made by selecting the label corresponding to the frequency peak in each window, if such peak is distinctive enough, *i.e.*, if any other peak is smaller than T_h times the biggest one (in a similar fashion to the orientation assignment step in the SIFT feature extractor [99]). Empirically, we set $T_h = 0.5$. A key feature of this technique is its ability to introduce void-class samples when a considered window cannot be assigned to a unique class, *i.e.*, it contains mixed classification labels. This procedure can be naturally extended to pseudo-labels (*i.e.*, network-generated segmentation maps) via a confidence measure over the maps that preserves only reliable predictions. In our case, such measure is computed efficiently by average pooling over the map of output probability peaks and used to mask the raw low-resolution pseudo-labels, *i.e.*, we select only confident labeling with average probability value greater than $T_p = 0.5$, empirically.

Prototype Extraction

Once computed, the feature-level label maps $\{\mathbf{I}_n^{s,t}\}_{n=1}^{N_{s,t}}$ can be used to extract the set $\mathcal{F}_c$ of feature vectors belonging to a generic class $c \in \mathcal{C}$ in a training batch $\mathcal{B}$:

$$\mathcal{F}_c^{s,t} = \left\{ \mathbf{F}_n^{s,t}[w,h] \in \mathbb{R}_{0+}^K \mid \mathbf{I}_n^{s,t}[w,h] = c, \forall n \in \mathcal{B} \right\}, \tag{5.9}$$

where (h, w) denote all possible spatial locations over a feature map, *i.e.*, $0 \leq h < H'$ and $0 \leq w < W'$. Exploiting this definition, we can identify the set of all feature vectors in batch $\mathcal{B}$ as the union $\mathcal{F}^{s,t} = (\bigcup_c \mathcal{F}_c^{s,t}) \cup \mathcal{F}_v^{s,t}$ where $\mathcal{F}_v^{s,t}$ are the sets of void-class samples. The class-wise sets are then used to estimate the per-batch class prototypes on labeled source data by simply computing their centroids:

$$\boldsymbol{\pi}_c[i] = \frac{1}{|\mathcal{F}_c^s|} \sum_{\mathbf{f} \in \mathcal{F}_c^s} \mathbf{f}[i] \quad \forall i, \;\; 1 \leq i \leq K. \tag{5.10}$$

Finally, to reduce estimation noise and obtain more stable and reliable prototypes, we apply exponential smoothing:

$$\hat{\boldsymbol{\pi}}_c = \eta \hat{\boldsymbol{\pi}}'_c + (1 - \eta)\boldsymbol{\pi}_c. \tag{5.11}$$

Where $\hat{\boldsymbol{\pi}}_c$ and $\hat{\boldsymbol{\pi}}'_c$ are the estimates of class c prototype respectively at current and previous optimization steps. We initialized $\hat{\boldsymbol{\pi}}_c = \mathbf{0}$ and empirically set $\eta = 0.8$. This strategy allows us to keep track of classes that are not present in the current batch of source samples (in this case we set $\eta = 1$ to propagate the previous estimate), aiding significantly in the unsupervised target tasks.

By exploiting the prototypes, whose computation has been just detailed, and forcing the source and target feature vectors to tightly assemble around them, we regularize the structure of the latent space, adapting representations into a domain independent class-wise distribution. Mathematically, we define the clustering objective as:

$$\hat{\mathcal{L}}_{cl}^{s,t} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{1}{|\mathcal{F}_c^{s,t}|} \sum_{\mathbf{f} \in \mathcal{F}_c^{s,t}} ||\hat{\boldsymbol{\pi}}_c - \mathbf{f}||^2, \tag{5.12}$$

where $||\cdot||$ denotes the L2 norm. This loss has multiple purposes: first, to better cluster representations in the latent space in a supervised manner, thus reducing the probability of erroneous classification. Second, to perform semi-supervised clustering on target samples exploiting network predictions as pseudo-labels. Finally, to improve prototype estimates, since forcing tighter clusters will result in more stable on-batch centroids, which will be closer to the moving-averaged prototype.

5.4.2 Perpendicularity of Latent Representations

We further enhance the space shaping action induced by the clustering objective by introducing a prototype perpendicularity loss. The idea is to improve the segmentation accuracy by better separating the tight and domain-invariant clusters on both domains. By doing so, we

allow the classifiers to increase the margin between decision boundaries and feature clusters, and, consequently, we reduce the likelihood of those boundaries to cross target high-density regions of the feature space (*i.e.*, regions populated by many target samples). Unlike previous works [239], class clusters of latent embeddings are forced to adopt a regular disposition, whilst being spaced out. In fact, we directly encourage a class-wise orthogonality property, by pushing prototypes to be perpendicular. In this way, not only we increase the distance among class clusters, but we jointly regularize the latent space and encourage channel-wise disjoint activations between different semantic categories.

To quantify perpendicularity in the loss value, we exploit the inner product in the euclidean space and its relationship with the angle θ between two vectors $\mathbf{j}$ and $\mathbf{k}$, *i.e.*, $\mathbf{j} \cdot \mathbf{k} = ||\mathbf{j}||\ ||\mathbf{k}|| \cos\theta$. Minimizing their normalized product is equivalent to maximizing the angle between them, since feature vectors have non-negative values. To capture this, we enforce cross-perpendicularity between any couple of prototypes:

$$\mathcal{L}_{pp}^{s} = \frac{1}{|\mathcal{C}|(|\mathcal{C}|-1)} \sum_{c_i, c_j \in \mathcal{C}, i \neq j} \frac{\boldsymbol{\pi}_{c_i}}{||\boldsymbol{\pi}_{c_i}||} \cdot \frac{\boldsymbol{\pi}_{c_j}}{||\boldsymbol{\pi}_{c_j}||}, \tag{5.13}$$

where the sum of the cosines over the set of all couples of non-void classes is computed. We use the per-batch computation of prototypes on source samples $\boldsymbol{\pi}_c$ (notice the missing hat on the prototype symbols, see Eq. (5.10)), guaranteeing a stronger gradient flow through the network. In addition, thanks to the tight geometric relation between prototype estimates and feature vectors enforced by $\hat{\mathcal{L}}_{cl}^{s,t}$, the effect induced by the orthogonality constraint on the prototypes is propagated to the vectors associated to them from either domains. The net result is the application of the shaping action to all feature vectors of each class, thus promoting perpendicularity between all individual components of distinct clusters.

The loss seeks to increase the angular distance between latent representations of separate classes, which is achieved when distinct sets of active feature channels are associated to distinct semantic categories. A similar orthogonality constraint has been proposed in [239]. Differently from [239] here we directly enforce the perpendicularity property between clusters, whereas in [239] each feature vector is considered independently without accounting for its semantic labeling, as the constraint is imposed in an unsupervised fashion.

5.4.3 Latent Norm Alignment Constraint

The last constraint we propose acts on the norm of source and target feature vectors. In particular, we promote the extraction of latent representations with uniform norm values across domains. Our objective is twofold. First, we aim at increasing the classification con-

fidence during target prediction, similarly to what achieved by adaptation strategies based on entropy minimization over the output space [55]. In particular, recent studies in image classification [97] highlight how the norm of target feature vectors tends towards smaller values than source ones, generally leading to reduced prediction confidence and potentially erroneous classifications. Second, we assist the perpendicularity loss by reducing the number of domain-specific feature channels exploited to perform classification. We argue, in fact, that by forcing the network to produce consistent feature norms, we reduce the number of channel activations switched on for only one of the two domains, as they would cause norm discrepancies. Formally, we define two separate objectives for source and target domains:

$$\mathcal{L}_{na}^{s} = \frac{1}{|\mathcal{F}^s|} \sum_{\mathbf{f} \in \mathcal{F}^s} \big|(\bar{f}_s + \Delta_f) - ||\mathbf{f}||\big|, \tag{5.14}$$

$$\mathcal{L}_{na}^{t} = \frac{1}{|\mathcal{F}^t|} \sum_{\mathbf{f} \in \mathcal{F}^t} \max(0, (\bar{f}_s + \Delta_f) - ||\mathbf{f}||), \tag{5.15}$$

where $\bar{f}_s$ is the mean of the feature vector norms computed from source samples in the previous optimization step, Δ_f dictates the enhancement step (we experimentally tuned it, *e.g.*, $\Delta_f = 0.002$). Feature vectors are pushed towards the same global average norm value, regardless of their labeling. This removes any bias generated by heterogeneous pixel-class distribution in semantic labels, which, for example, would cause the most frequent classes to show larger norm than the average. The source-specific constraint (Eq. (5.14)) forces both the inter-class alignment and enhancement step, *i.e.*, it ensures that norms are progressively increased throughout the training process, towards a common value for all the classes. On the other hand, the target objective (Eq. (5.15)) focuses on domain-alignment, by enforcing the target norms to be similar to the source ones. Furthermore, since target features have typically smaller norms, we do not penalize target norms exceeding the current source reference value.

5.5 Experimental Setup

In this section we describe the experimental setup used to evaluate OCE and LSR methods.

Datasets We test the proposed OCE [239] and LSR [240] methods on synthetic-to-real UDA on road-view semantic segmentation. Supervised training is performed on the synthetic datasets GTA5 [18] and SYNTHIA [19]. We employ Cityscapes [14] as target domain. We perform training in a closed-set [234] setup, *i.e.*, source and target class sets coincide. Therefore, we use the 19 and 16 common classes for GTA5 and SYNTHIA, respectively.

Network Implementation The modules introduced in this work are agnostic to the underlying network architecture and can be extended to other scenarios. For fair comparison with previous works [1, 48, 55] we employ the DeepLab-V2, a fully convolutional segmentation network with ResNet-101 [100] or VGG-16 [101] as backbones. Further details on the segmentation network architecture can be found in [48, 55], as we follow the same implementation adopted in those works. We initialize the two encoder networks with ImageNet [26] pretrained weights. In addition, prior to the actual adaptation phase, we supervisedly train the segmentation network on source data.

Training Details The model is trained with the starting learning rate set to $2.5\text{e-}4$ and decreased with a polynomial decay rule of power 0.9. We employ weight decay regularization of $5\text{e-}4$. Following [1], we also randomly apply mirroring and gaussian blurring for data augmentation during the training stage. To accommodate for GPU memory limitations, we resize images from the GTA5 dataset up to a resolution of 1280×720 px, as done by [48]. SYNTHIA images are instead kept to the original size of 1280×780 px. As for the target Cityscapes dataset, training unlabeled images are resized to 1024×512 px, whereas the results of the testing stage are reported at the original image resolution (2048×1024 px). As evaluation metric, we employ the mean Intersection over Union (mIoU). We use a batch size of 2 (1 source and 1 target samples), training the network for $27,450$ steps (*i.e.*, 10 epochs of the Cityscapes [14] dataset) and employing early stopping based on a validation set (subset of the original training set), which was also exploited for the hyper-parameters search in our loss terms. The entire model is developed using PyTorch and trained with a single GPU.

Finally, with LSR we followed an enhanced pre-trained strategy: the model is trained on source-only samples with a batch size of 10 for more stable optimization, and using patches of 512×512 px and data augmentation to remove visual biases introduced by the running mean components of batch-normalization layers when full images are employed. As an example, a dark patch on the bottom half of the image will often be interpreted as *road*, while a light patch on the top half will often be interpreted as *sky*, which is not always true (see the random camera angles in the SYNTHIA dataset) and preserving such behavior may be detrimental for some applications. This allows to obtain a model better performing on the target dataset, even before the actual domain adaptation phase.

5.6 Experimental Results

We evaluate the performance of the proposed OCE [239] and LSR [240] approaches on two widely used synthetic-to-real adaptation scenarios, namely the GTA5 $\rightarrow$ Cityscapes and

SYNTHIA → Cityscapes benchmarks. Tables 5.1 and 5.2 reports the numerical results of the experimental evaluation. We compare the proposed OCE [239] and LSR [240] frameworks to several state-of-the-art methods, which, similarly to our approach, resort to a direct or indirect form of feature-level regularization and distribution alignment to achieve domain adaptation [1, 35, 40, 48, 55, 78, 102, 103]. With *source only* we indicate the naïve fine-tuning approach, in which no form of target adaptation assists the standard source supervision.

An ablation study and a discussion of the effects brought by the proposed loss terms is also included. Notice that our methods are trained end-to-end, thus we can seamlessly add other adaptation techniques, *e.g.*, entropy minimization/adversarial output-level approaches or generative-adversarial input-level solutions. To prove such compatibility, we introduce an entropy-minimization loss [1] to our frameworks.

Relative Performance Evaluation

For a better evaluation, we introduce a novel measure, called mASR (*mean Adapted-to-Supervised Ratio*), to capture the relative performance between an adapted architecture and its target supervised counterpart (higher means better). We compare the IoU score of the adapted network for each class $c \in \mathcal{C}$ (IoU^c_{adapt}) with the results of supervised training on target data (IoU^c_{sup}, that is a reasonable upper bound estimate) and we average the scores:

$$\text{mASR} = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \text{ASR}^c, \quad \text{ASR}^c \stackrel{def}{=} \frac{\text{IoU}^c_{adapt}}{\text{IoU}^c_{sup}}. \tag{5.16}$$

In this metric the contribution of each class is inversely proportional to the capacity of the segmentation model to learn it in the supervised reference scenario, thus emphasizing the most challenging semantic categories. Results are reported in Table 5.2.

5.6.1 Adaptation from GTA5 to Cityscapes

Analysis of OCE method

For the GTA5 → Cityscapes and ResNet-101 configuration, our OCE approach shows state-of-the-art performances in feature-level UDA for semantic segmentation, achieving 45.3% of mIoU, which is further boosted up to 45.9% by the entropy minimization objective. By looking at Table 5.1, we observe about 9% increase over the *source only* baseline, with the improvement well distributed over all the classes. A similar behavior can be noted when switching to the less performing VGG-16 backbone: we achieve 33.7% mean IoU with $\mathcal{L}'_{tot}$ (*i.e.*, without the entropy minimization objective) and 34.2% with $\mathcal{L}_{tot}$ (*i.e.*, with all compo-

Table 5.1: Numerical results of the GTA5 and SYNTHIA to Cityscapes adaptation scenarios in terms of per-class and mean IoU. Evaluations are performed on the validation set of the Cityscapes dataset. In all the experiments, the DeepLab-V2 segmentation network is employed, with VGG-16 (top) or ResNet-101 (bottom) backbones. The mIoU* results in the last column refer to the 13-classes configuration, *i.e.*, classes marked with * are ignored. MaxSquares IW(r) denotes our re-implementation, as original results are provided only for the ResNet-101 backbone. The highest values have been highlighted in bold.

	Method	Road	Sidewalk	Building	Wall	Fence	Pole	T. Light	T. Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorbike	Bicycle	mIoU (all)
VGG16	Source Only	26.5	13.3	45.1	6.0	15.2	16.5	21.3	8.5	78.0	8.3	59.7	45.0	10.5	69.1	22.8	17.9	0.0	16.4	2.7	25.4
	FCNs ITW [35]	70.4	**32.4**	62.1	14.9	5.4	10.9	14.2	2.7	79.2	21.3	64.6	44.1	4.2	70.4	8.0	7.3	0.0	3.5	0.0	27.1
	CyCADA (f) [40]	85.6	30.7	74.7	14.4	13.0	17.6	13.7	5.8	74.6	15.8	69.9	38.2	3.5	72.3	16.0	5.0	0.1	3.6	0.0	29.2
	CBST [78]	66.7	26.8	73.7	14.8	9.5	**28.3**	25.9	10.1	75.5	15.7	51.6	**47.2**	6.2	71.9	3.7	2.2	**5.4**	**18.9**	**32.4**	30.9
	MinEnt [55]	85.1	18.9	76.3	**32.4**	**19.7**	19.9	21.0	8.9	76.3	**26.2**	63.1	42.8	5.9	**80.8**	**20.2**	9.8	0.0	14.8	0.6	32.8
	MS-IW (r) [1]	81.4	20.0	75.4	19.4	19.1	16.1	24.4	7.9	78.8	22.9	65.9	45.0	12.3	74.6	16.1	10.3	0.2	11.3	1.0	31.7
	OCE ($\mathcal{L}'_{tot}$) [239]	83.6	16.6	79.0	19.8	18.7	21.5	27.3	**15.9**	80.2	14.3	**72.6**	47.0	17.5	76.8	16.6	**13.9**	0.1	16.0	3.4	33.7
	OCE ($\mathcal{L}_{tot}$) [239]	**86.0**	13.5	**79.4**	20.4	18.5	21.5	**27.6**	15.2	**80.8**	21.9	**72.6**	46.3	**18.1**	80.0	16.9	13.1	1.0	14.6	2.0	**34.2**
ResNet101	Source Only	81.8	16.3	74.4	18.6	12.7	23.5	29.3	18.1	73.5	21.4	77.6	55.6	25.6	74.1	28.6	10.2	3.0	25.8	32.7	37.0
	ASN (f) [48]	83.7	27.6	75.5	20.3	19.9	27.4	28.3	27.4	79.0	28.4	70.1	55.1	20.2	72.9	22.5	35.7	**8.3**	20.6	23.0	39.3
	MinEnt [55]	84.4	18.7	80.6	23.8	23.2	28.4	36.9	23.4	83.2	25.2	**79.4**	59.0	**29.9**	78.5	33.7	29.6	1.7	29.9	33.6	42.3
	SAPNet [103]	88.4	38.7	79.5	**29.4**	**24.7**	27.3	32.6	20.4	82.2	32.9	73.3	55.5	26.9	82.4	31.8	41.8	2.4	26.5	24.1	43.2
	MS-IW [1]	89.3	**40.5**	81.2	29.0	20.4	25.6	34.4	19.0	83.6	34.4	76.5	59.2	27.4	83.8	**38.4**	43.6	7.1	**32.2**	32.5	45.2
	OCE ($\mathcal{L}'_{tot}$) [239]	88.7	32.2	81.8	24.1	22.1	**30.8**	**37.6**	**32.8**	83.4	36.3	76.0	60.0	27.0	81.0	34.2	43.0	8.0	23.4	38.1	45.3
	OCE ($\mathcal{L}_{tot}$) [239]	**89.4**	30.7	**82.1**	23.0	22.0	29.2	**37.6**	31.7	**83.9**	**37.9**	78.3	**60.7**	27.4	**84.6**	37.6	**44.7**	7.3	26.0	**38.9**	**45.9**
	LSR [240]	87.7	32.6	82.6	29.1	23.0	28.5	36.1	28.5	**84.8**	**41.8**	**80.1**	59.4	23.8	76.5	**38.4**	**45.8**	7.1	28.5	40.1	**46.0**

(a) GTA5 → Cityscapes

	Method	Road	Sidewalk	Building	Wall*	Fence*	Pole*	T. Light	T. Sign	Vegetation	Sky	Person	Rider	Car	Bus	Motorbike	Bicycle	mIoU (all)	mIoU* (13-cl)
VGG16	Source Only	7.8	13.7	66.6	2.2	0.0	23.9	4.8	13.3	71.2	76.5	49.2	12.1	67.1	24.5	9.8	9.2	28.3	32.8
	FCNs ITW [35]	11.5	19.6	30.8	4.4	0.0	20.3	0.1	11.7	42.3	68.7	51.2	3.8	54.0	3.2	0.2	0.6	20.2	22.9
	Cross-City [102]	62.7	25.6	**78.3**	-	-	-	1.2	5.4	81.3	81.0	37.4	6.4	63.5	16.1	1.2	4.6	-	35.7
	CBST [78]	69.6	28.7	69.5	**12.1**	**0.1**	**25.4**	11.9	13.6	**82.0**	**81.9**	49.1	14.5	66.0	6.6	3.7	**32.4**	35.4	36.1
	MinEnt [55]	37.8	18.2	65.8	2.0	0.0	15.5	0.0	0.0	76.0	73.9	45.7	11.3	66.6	13.3	1.5	13.1	27.5	32.5
	MS-IW (r) [1]	9.1	12.7	72.5	1.0	0.0	22.3	7.0	8.4	80.0	77.9	49.4	10.0	71.8	23.8	6.0	13.5	29.1	34.0
	OCE ($\mathcal{L}'_{tot}$) [239]	**78.5**	29.9	77.7	1.2	**0.1**	24.1	11.9	**15.0**	78.7	78.5	51.0	15.4	73.7	**24.7**	**10.1**	23.5	**37.1**	**43.7**
	OCE ($\mathcal{L}_{tot}$) [239]	78.3	**30.1**	78.0	1.7	**0.1**	24.1	**12.0**	14.6	79.7	79.1	**51.4**	**15.5**	**74.4**	23.7	9.1	22.7	**37.1**	**43.7**
ResNet101	Source Only	39.5	18.1	75.5	10.5	0.1	26.3	9.0	11.7	78.6	81.6	57.7	21.0	59.9	30.1	15.7	28.2	35.2	40.5
	ASN (f) [48]	62.4	21.9	76.3	-	-	-	11.7	11.4	75.3	80.9	53.7	18.5	59.7	13.7	**20.6**	24.0	-	40.8
	MinEnt [55]	73.5	29.2	77.1	7.7	0.2	27.0	7.1	11.4	76.7	82.1	57.2	**21.3**	69.4	29.2	12.9	27.9	38.1	44.2
	SAPNet [103]	81.7	33.5	75.9	-	-	-	7.0	6.3	74.8	78.9	52.1	**21.3**	75.7	**30.6**	10.8	28.0	-	44.3
	MS-IW [1]	78.5	34.7	76.3	6.5	0.1	**30.4**	12.4	12.2	**82.2**	**84.3**	**59.9**	17.9	80.6	24.1	15.2	31.2	40.4	46.9
	OCE ($\mathcal{L}'_{tot}$) [239]	64.4	25.5	77.3	**14.3**	**0.9**	29.6	**21.2**	**24.2**	76.6	79.7	53.7	15.5	79.7	11.0	11.0	**35.2**	38.7	44.2
	OCE ($\mathcal{L}_{tot}$) [239]	**88.3**	**42.2**	**79.1**	7.1	0.2	24.4	16.8	16.5	80.0	**84.3**	56.2	15.0	**83.5**	27.2	6.3	30.7	**41.1**	**48.2**
	LSR [240]	81.0	36.9	**79.5**	13.4	0.2	**28.7**	9.0	16.1	79.1	81.7	**57.9**	**21.6**	77.2	35.3	14.2	**35.4**	**41.7**	48.1

(b) SYNTHIA → Cityscapes

nents enabled) starting from the 25.4% of the baseline scenario without adaptation. When compared with other approaches, our method performs better than standard feature-level adversarial techniques [35, 40, 48, 103]. For example, with the VGG-16 backbone there is a gain of 4.5% w.r.t. [40]. This proves that a more effective class-conditional alignment has been ultimately achieved in the latent space by our approach. Due to the similar regularizing effect over feature distribution and comparable ease of implementation, we also compare our framework with some entropy minimization and self-training techniques [1, 55, 78], further showing the effectiveness of our adaptation strategy even if the gap here is a bit more limited. With both backbones, our novel feature level modules ($\mathcal{L}'_{tot}$) perform better than MaxSquare IW [1]. Moreover, adding the entropy minimization objective from [1] to $\mathcal{L}'_{tot}$ provides a slight but consistent improvement. It is worth noting that our method does not rely on additional trainable modules (*e.g.*, adversarial discriminators [35, 40, 48, 103]) and the whole adaptation process is end-to-end, not requiring multiple separate steps to be re-iterated (*e.g.*, pseudo-labeling in self-training [78]). Moreover, being focused solely on feature level adaptation, it could be easily integrated with other adaptation techniques working at different network levels, such as the input (*e.g.*, generative approaches) or output (*e.g.*, self-training), as shown by the addition of the output-level entropy-minimization loss.

Figure 5.4 displays some qualitative results on the Cityscapes validation set of the adaptation process when the ResNet-101 backbone is used. We observe that the introduction of the clustering module is beneficial to the target segmentation accuracy w.r.t. the *source only* case. Some small prediction inaccuracies remain, that are corrected with the introduction of the orthogonality, sparsity and entropy modules in the complete framework. By looking at the last two columns, we also notice that our entire framework shows an improvement over the individual entropy-minimization like objective from [1], which is reflected in a better detection accuracy both on frequent (*e.g.*, *road*, *vegetation*) and less frequent (*e.g.*, *traffic sign*, *bus*) classes.

Analysis of LSR method

In the GTA5 $\rightarrow$ Cityscapes setup our LSR approach achieves a score of 46.0% mIoU, with a gain of 9.1% compared to the baseline. In addition, LSR outperforms all competitors, with only [1] and OCE [239] able to get close to our result, while there is a quite relevant gap compared to other methods. Such improvement is quite stable across most single-class IoU scores, and is particularly evident in difficult classes, such as *terrain*, where our strategy shows the highest percentage gains. As an index of robustness and performance balance, we use the standard deviation of the per-class IoUs. LSR reduces it by 0.8 compared to OCE [239] (from 24.8 to 24.0). Furthermore, LSR surpasses the same strategy by 0.4% in terms of

Table 5.2: Comparison of adaptation strategies in terms of IoU, mIou and mASR (%) (Section 5.6). The mIoU[1] and mASR[1] restrict to 13 classes, ignoring the ones with same superscript. The highest values have been highlighted in bold.

Setup	Method	mIoU	mASR
	Target Only	64.8	100
GTAV→Cityscapes	Baseline [239]	36.9	54.0
	ASN (feat) [48]	39.0	56.9
	MinEnt [55]	42.3	61.9
	SAPNet [103]	43.2	63.1
	MaxSquareIW [1]	45.5	62.2
	OCE [239]	45.9	67.3
	LSR [240]	**46.0**	**67.7**

Setup	Method	mIoU	mIoU[1]	mASR	mASR[1]
	Target Only	64.8	-	100	100
SYNTHIA→Cityscapes	Baseline [239]	30.1	34.3	41.7	44.6
	ASN (feat) [48]	-	40.8	-	52.5
	MinEnt [55]	38.1	44.2	51.1	56.3
	SAPNet [103]	-	44.3	-	56.0
	MaxSquareIW [1]	39.4	45.2	53.8	58.3
	OCE [239]	41.1	**48.2**	54.3	60.9
	LSR [240]	**41.7**	48.1	**56.5**	**61.6**

mASR (Table 5.2), reaching 67.7%, meaning that our approach shows improved accuracy over more challenging classes, thanks to the enhanced latent space regularization.

Some qualitative results are reported in the top half of Figure 5.5. From visual inspection, we can verify the increased precision of *t. sign*, *t. light*, *pole* and *person* borders in multiple images. Furthermore, LSR correctly classifies the *wall* in the first image (labeled as *building* by competitors) and the *building* in the second image (confused for *fence* by competitors).

5.6.2 Adaptation from SYNTHIA to Cityscapes

Analysis of OCE method

To further prove the efficacy of the OCE method, we evaluate it on the more challenging SYNTHIA → Cityscapes benchmark, where a larger domain gap exists. Once more, our approach proves to be successful in performing domain alignment with both ResNet-101 and VGG-16 backbones, reaching state-of-the-art results for feature-level UDA in both configurations (see Table 5.1). When ResNet-101 is used, the mIoU* on the 13 classes setting is pushed up to 48.2% from the original 40.5% of *source only*, while the VGG-16 scenario witnesses an even more improved performance gain of almost 11% over the no adaptation baseline till a final value of 43.7%. Differently from the GTA5 → Cityscapes case, here the contribution of the entropy-minimization module varies for the two backbones. The induced benefit is absent with VGG-16, since the clustering, orthogonality and sparsity jointly enforced already carry the whole adaptation effort. Besides, even the $\mathcal{L}_{em}$ objective alone (*i.e.*, MaxSquares IW $^{(r)}$ [1]) displays quite limited gain over the no adaptation baseline. On the contrary, the regularizing effect of the entropy objective is strongly valuable in case the ResNet-101 backbone is used. Yet, the combination of all modules together actually pro-

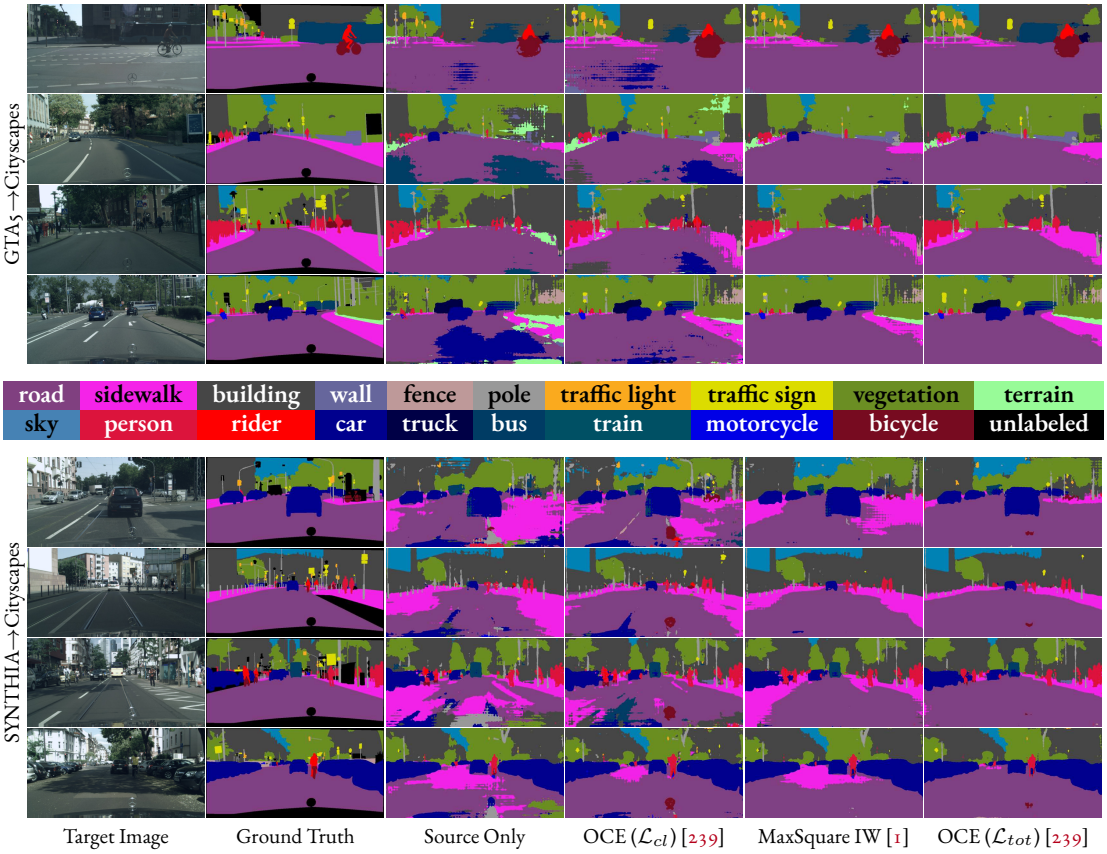


Figure 5.4: Semantic segmentation of sample scenes from the Cityscapes validation set when adaptation is performed from the GTA5 and SYNTHIA source datasets and the DeepLab-V2 with ResNet-101 backbone is employed.

vides a noticeable boost over both the entropy and feature-level modules separately applied. As for the GTA5 scenario, our model shows better performance than feature-level adversarial adaptation [35, 48, 102, 103] and output-level approaches [55, 78] comparable in computational ease. By looking at qualitative results present in Figure 5.4 on the Cityscapes validation set when adapting from SYNTHIA, we once more notice overall improved segmentation accuracy across multiple semantic classes.

Analysis of LSR method

In the SYNTHIA → Cityscapes setup LSR surpasses all the competitors in the 16-classes configuration, reaching 41.7% of mIoU, with a gap compared to the best competing approach larger than in the previous setting. Once more, our method reduces the standard deviation of the IoU distribution (27.7 compared to 29.7 of OCE [239]). At the same time, LSR shows the highest mASR, surpassing the second best approach (on 16 classes) by 2.2%

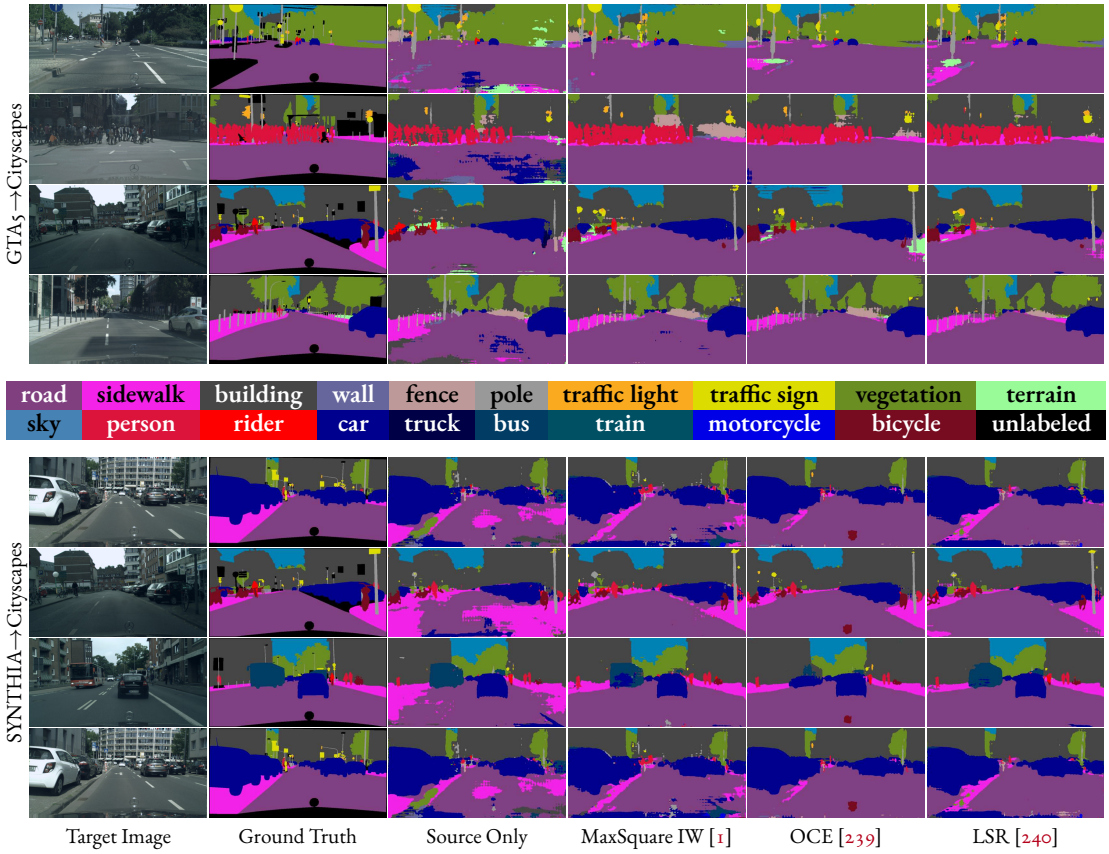


Figure 5.5: Semantic segmentation of sample scenes from the Cityscapes validation dataset when adaptation is performed from the GTA5 and SYNTHIA source datasets and the DeepLab-V2 with ResNet-101 backbone is employed.

(Table 5.2). On the 13-classes setup our strategy outperforms all competitors in terms of mASR with only a marginal loss in mIoU score, confirming our previous claim of better performance balance across classes and reduced gap with respect to supervised learning.

Similarly to the GTA5 case, the performance gain can be noticed also from the qualitative results reported in the bottom half of Figure 5.5. The segmentation maps show an overall improvement in the shape of *sidewalk* and *pole* classes. Our LSR method is able to correctly detect the *car* and *person* behind the *pole* in the last image, which are missed or wrongly classified by competing strategies (*e.g.*, the *car* is confused as the *person* in [1]), and can accurately predict the *t. sign*, missed by some strategies. In addition, we observe improved segmentation of the *bus* class, which OCE, in turn, struggles to correctly segment. Finally, we remark that LSR is able to correct the region around the car logo (bottom-part of each figure), which is often confused with *bicycle*, *car* or *bus* by competitors.

Table 5.3: Ablation results on the contribution of each adaptation module in the GTA5 to Cityscapes scenario and with ResNet-101 as backbone.

$\mathcal{L}_{cl}$	$\mathcal{L}_{or}$	$\mathcal{L}_{sp}$	$\mathcal{L}_{em}$	mIoU
				37.0
✓				42.3
	✓			43.2
		✓		43.7
			✓	44.8
✓	✓	✓		45.3
✓	✓	✓	✓	45.9

5.6.3 Orthogonal and Clustered Embeddings: Ablation Study

To verify the robustness of the OCE framework, we perform an extensive ablation study on the adaptation from GTA5 to Cityscapes with ResNet-101 as backbone. First, we examine the contribution of each loss to the final mIoU; then, we investigate the effect of each novel loss component.

The contribution of each loss to the adaptation module is shown in Table 5.3. Every loss component largely improves the final mIoU results from 37.0% of the *source only* scenario up to a maximum of 44.8%. Combining the 3 novel modules of this work, we achieve a mIoU of 45.3%, which is higher than all the losses alone, but lower than our complete framework with all the losses enabled (45.9%).

Discriminative Clustering. To investigate the effect of the OCE's clustering module ($\mathcal{L}_{cl}$) we show a t-SNE [104] plot of features extracted from a sample image of the Cityscapes validation set. In particular, we provide a comparison of the discovered low-dimensional feature distribution for two distinct training settings, *i.e.*, *source only* and adaptation with $\mathcal{L}_{cl}$ only. The results are reported in Figure 5.6, where sparse points in the *source only* plot turn out more tightly clustered and spaced apart in the $\mathcal{L}_{cl}$ approach (*e.g.*, look at the *person* class).

Feature Orthogonality. We analyze the orthogonality constraint ($\mathcal{L}_{or}$) via a similarity score defined as an average class-wise cosine similarity measure (Figure 5.7). The cosine distance is first computed for every pair of feature vectors from a single target image. Then, the average values are taken over all features from the same class to get a score for each pair of semantic classes. The final values are computed by averaging over all images from the Cityscapes validation set. The score computation is performed for two different configurations, *i.e.*, $\mathcal{L}_{cl}+\mathcal{L}_{or}+\mathcal{L}_{sp}$ and $\mathcal{L}_{cl}+\mathcal{L}_{sp}$, to highlight the effect induced by the orthogonality module. The results in Figure 5.7 show that $\mathcal{L}_{or}$ causes the similarity score to significantly

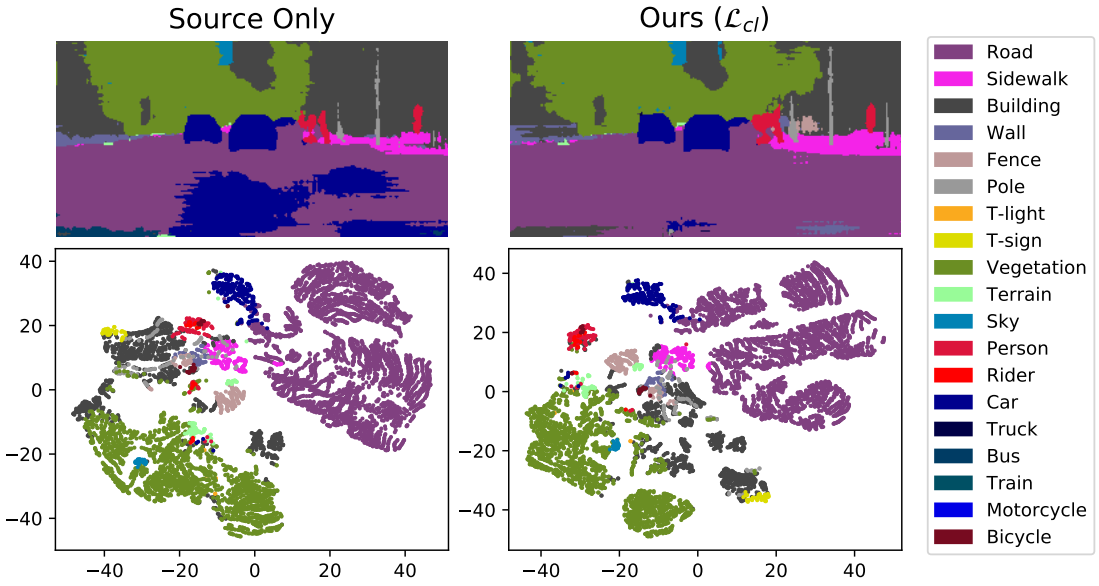


Figure 5.6: T-SNE computed over features of a single image of the Cityscapes validation set when adapting from GTA5.

increase within almost all the classes.

Feature Sparsity. To understand the efficacy of the sparsity loss ($\mathcal{L}_{sp}$) we compute the sparsity scores as the fraction of activations in normalized feature vectors close to 0 or 1 (Figure 5.8). Closeness is quantified as being distant from 0 or 1 less than a threshold, which we set to $1e{-}4$. As for the similarity scores, the sparsity measures for a single target image are obtained through averaging over all feature vectors from the same class. The final results correspond to the mean values over the entire Cityscapes validation set. We compute the scores for two different configurations, *i.e.*, $\mathcal{L}_{cl} + \mathcal{L}_{or} + \mathcal{L}_{sp}$ and $\mathcal{L}_{cl} + \mathcal{L}_{or}$, so that we can inspect the effect that the sparsity module is providing on feature distribution. From Figure 5.8 we can appreciate that $\mathcal{L}_{sp}$ effectively achieves higher values of sparseness for all the classes.

5.6.4 Latent Space Regularization: Ablation Study

In this section, we evaluate the impact of each constraint on the final accuracy. Quantitative results are reported in Figure 5.9a, where we evaluate our strategy by removing each constraint independently and evaluating the impact on the final accuracy. In particular, we show how the absence of each of our losses reduces the final performance by a minimum of 0.8% mIoU and an average of 1% mIoU. Each module brings a significant improvement in terms of accuracy and all the components are needed for the best results. In addition, we remark the better performance achieved before the adaptation phase by the enhanced pre-training, compared to that achieved on OCE with standard source-based pre-training (Table

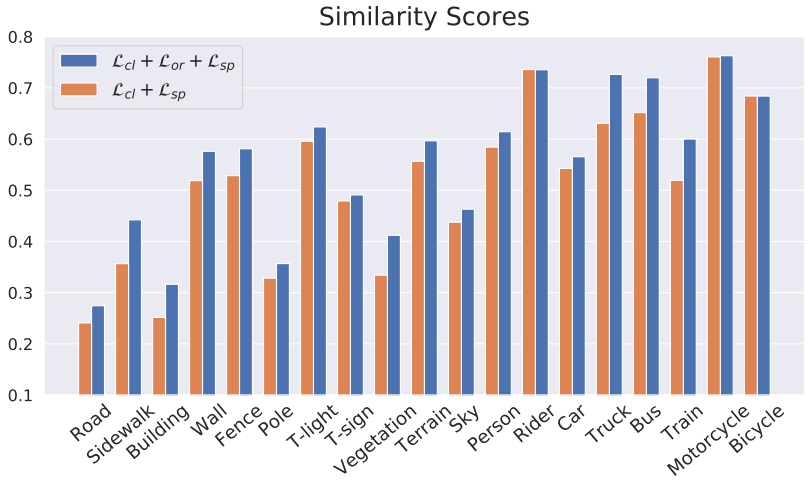


Figure 5.7: Similarity scores computed over images of the Cityscapes validation set when adapting from GTA5.

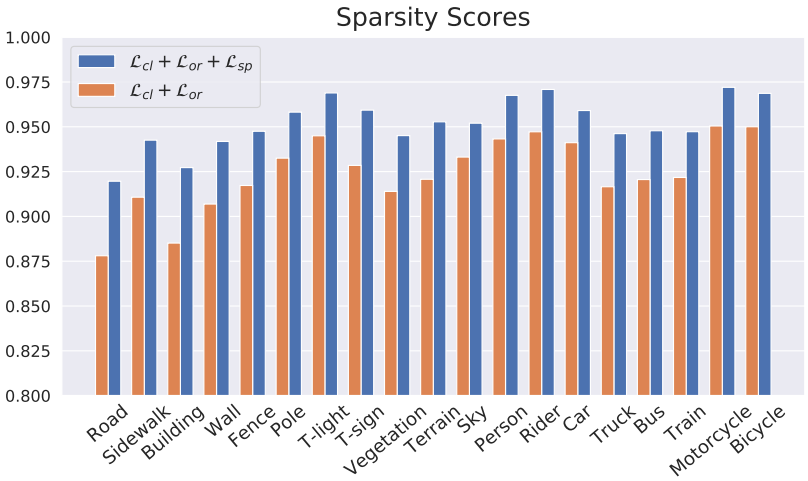


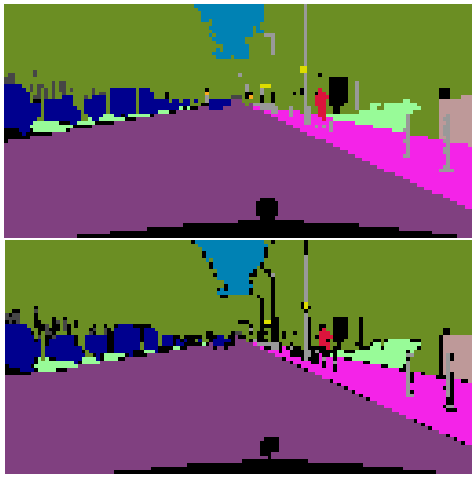
Figure 5.8: Sparsity scores computed over images of the Cityscapes validation set when adapting from GTA5.

5.3). Finally, the $\hat{\mathcal{L}}_{cl} + \mathcal{L}_{pp} + \mathcal{L}_{na}$ combination (*i.e.*, without entropy minimization) performs worse than OCE's counterpart (*i.e.*, $\mathcal{L}_{cl} + \mathcal{L}_{or} + \mathcal{L}_{sp}$), but shows to work better in combination with $\mathcal{L}_{em}$, for slightly better overall performance.

Histogram Downsampling. As for the novel downsampling scheme (Section 5.4.1), the goal of the proposed frequency-aware setup is to label only feature locations with a clear class assignment. This aims to reduce cross-talk between neighboring features of different classes, thus improving class discriminativeness at the latent space. We can observe this phenomenon in Figure 5.9b, where the label map downsampled via our frequency-aware scheme (bottom) marks some features close to the edges of objects as *unlabeled*. This is confirmed by the class distribution of the downsampled segmentation maps (*i.e.*, to match the spatial resolution of the feature level), reported in Figure 5.10 for both the histogram-aware scheme (ours) or

$\hat{\mathcal{L}}_{cl}$	$\mathcal{L}_{pp}$	$\mathcal{L}_{na}$	$\mathcal{L}_{em}$	mIoU
				42.8
	✓	✓	✓	44.8
✓		✓	✓	44.9
✓	✓		✓	45.2
✓	✓	✓		44.2
✓	✓	✓	✓	46.0

(a)



(b)

Figure 5.9: (a) Ablation analysis on impact of different loss components. (b) Example of semantic label downsampled via nearest (top) or by frequency-aware (bottom) approaches.

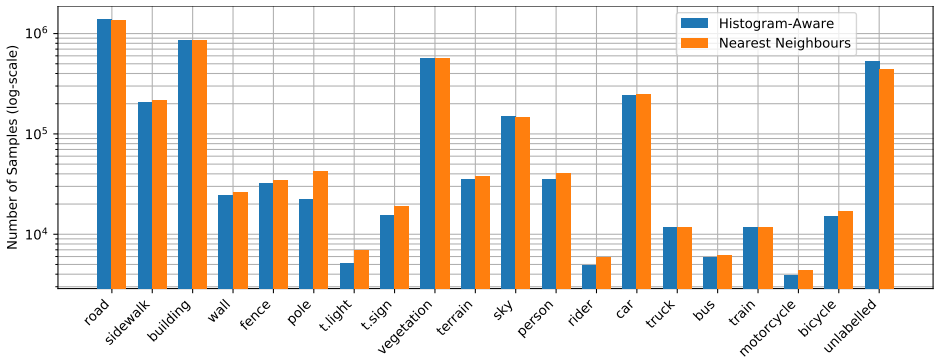
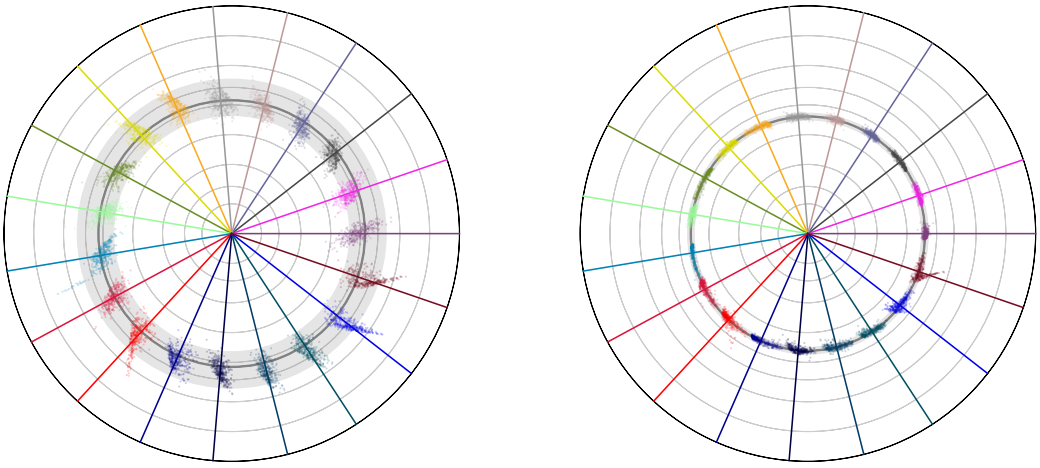


Figure 5.10: Class distribution of segmentation maps downsampled either via histogram-aware or via nearest neighbor.

the standard nearest neighbors one. In particular, the histogram-aware scheme generally seldom preserves small object classes, promoting *unlabeled* classification when discrimination between classes is uncertain.

Norm Alignment. We analyze the effect of the norm alignment constraint in Figure 5.11, where we show a plot of some feature vectors after projecting them to a 2D space for better visualization. This and the subsequent plots were produced using a balanced subset of feature vectors (350 vectors per class) extracted from the *Cityscapes* validation set for a fair comparative analysis across the classes.

In Figure 5.11 the norm of each vector is represented in the radial axis in log scale, the inner angle between any point and its prototype's direction is represented in the angular axis. The original direction in the high dimensional space of each centroid is ignored, as a meaningful



(a) Baseline. **(b)** LSR.

Figure 5.11: Feature distribution before and after adaptation.

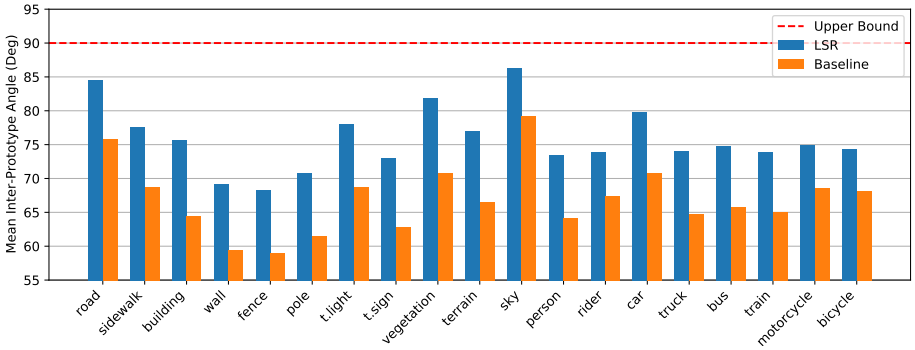
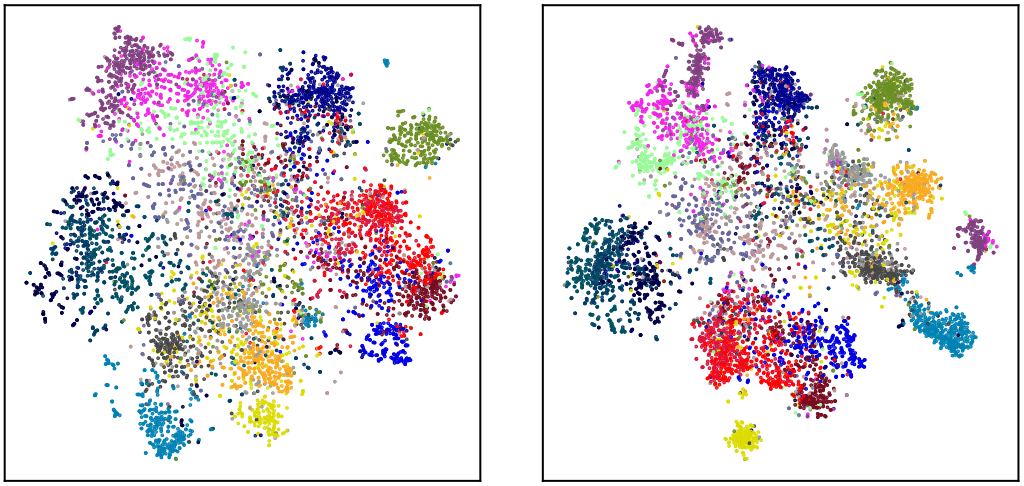


Figure 5.12: Average inter-prototype angle, comparison between Source Only and LSR.

representation would be very difficult to achieve in 2D. Instead, we assign to each centroid a reference angle (as shown by the colored lines) and plot the associated feature vectors centered on it. The plots also reports a confidence interval for the global average norm, to highlight how the proposed norm alignment constraint (Section 5.4.3) effectively promotes uniform norm values: the dark gray line represents the median of the distribution and the shaded gray area represents the 95% confidence interval. The effect of our space shaping strategy is clearly visible, in that the norms align very tightly around the global mean value (smaller shaded region around the unique gray line). The distribution of the points around each prototype is shrunk, thanks to the clustering objective, while the centroids themselves are pushed away from each other, thanks to the perpendicularity constraint (not visible from this plot, but appreciable in Figure 5.12 as we discuss next).



(a) Baseline. **(b)** LSR.

Figure 5.13: T-SNE plots comparing class-wise feature distribution before and after adaptation. Points are color coded according to the legend of Figure 5.5.

Prototype Perpendicularity. To analyze the effect of the perpendicularity constraint, Figure 5.12 shows the distribution of the average inner angle between a prototype's direction and the direction of each other prototype. Ideally, we aim at producing as perpendicular prototypes as possible, in order to reduce the overlap of different semantic classes over feature channels (*i.e.*, cross-talk). The red dashed line at 90 degrees shows the target value for perpendicularity, which is also the upper bound for the angle, as our feature vectors have all non-negative coordinates. From the figure, it emerges clearly that our strategy results in an average increase of more than 5 degrees.

Feature Clustering. Finally, we analyze the LSR's clustering objective by means of a t-SNE [104] embedding produced on the normalized features (to remove the norm information, enhancing the angular one) and we report it in Figure 5.13. Our strategy increases significantly the cluster separation in the high dimensional space and the spacing between clusters belonging to different classes. As a side effect, this also reduces the probability of confusing visually similar classes (*e.g.*, the *truck* class with the *bus* and *train* ones).

Part II

Learning under Task Shift

6 Transfer Learning Across Tasks

6.1 Continual Task Learning

Recently, deep learning algorithms have evolved rapidly to tackle a wide variety of tasks previously considered extremely challenging, particularly in the field of computer vision, where deep models have been shown to achieve human-like performance in many contexts. Deep learning techniques have matured along the way and the transition from academic research to various practical and industrial applications has begun to be successful. However, practical applications soon raised the need for solutions able to improve the learned knowledge over time in order to accomplish new tasks without forgetting previous information. This represents the building paradigm of continual learning in its essence. In other words, when deep learning models are deployed into the real-world, we would like to have the possibility to expand the their prediction proficiency to new tasks (leveraging new training data), without retraining them from scratch.

In general, the main issue with these computational models is that they are prone to *catastrophic forgetting* [105, 106], *i.e.*, training them with new information interferes with previously learned knowledge and typically greatly degrades the overall performance. Deep learning algorithms, in particular, perform the best when all data samples are available during the training phase and, therefore, require their training to be performed on the entire dataset in order to adapt to changes in the data distribution. When experiencing sequential tasks with samples progressively available over time, their performance significantly decreases on previously learned tasks, as the network parameters are optimized for the newly introduced task without accounting for former ones, if no ad-hoc provisions are employed [107, 108].

Continual learning (also called incremental learning, lifelong learning or never ending learning), then, is the set of techniques designed to face this challenging scenario in which a sequence of tasks comes in succession. Despite being a long-standing problem in computational models [106], in deep learning it has been tackled with some successes only recently. To further prove its relevance, it is worthwhile to consider an analogy between machine learning and human learning. Indeed, humans encounter a continual stream of learning tasks and are able to generalize to similar unseen tasks [109]. To understand the brain mechanisms and to translate them into computational models, many connectionist and biologically-inspired attempts have been made throughout the years [110, 111].

Recently, the problem has been actively investigated in some image-level visual tasks (*i.e.*, with one or few labels per each image) such as image classification [112–114] and object detection [115]. In dense labeling tasks such as semantic segmentation, the problem has been faced only very recently [3, 4, 116–125] due to the inherent increased complexity. Before digging into the definition of continual learning and exploring its application to dense labeling tasks, we point out some general reviews on the topic, not directly dealing with semantic segmentation. We refer to [126] for catastrophic forgetting in connectionist models, while [108] is the first review to critically compare recent works about the phenomenon in deep learning models. In [127] many approaches are compared into a common framework and in [128] the challenges of continual learning are described with a special focus on robotics.

In this dissertation we will propose two frameworks to address continual learning for image classification (Chapter 7) and semantic segmentation (Chapter 8). First, we will describe an approach [242] targeting class incremental image classification based on feature-level replay by modeling representation drift. In particular, the goal is to retrieve an estimate of the feature distribution associated to past tasks, in order to reiterate latent representations of no longer available training data and thus mitigate forgetting. Then, we will introduce RECALL [241] for class incremental segmentation, based on a replay mechanism at the input level, resorting to image generation and web-crawling.

The reminder of this chapter comprises a formal introduction and problem formulation of Continual Learning under task shift, with a special focus to Class Incremental Learning (Section 6.2); then, an overview of CIL techniques that can be found in literature will be presented, both for whole-image classification and semantic segmentation as target tasks (Section 6.3); we will finally conclude the chapter with a description of the most popular experimental setups and benchmarks exploited for the evaluation of CIL methods (Section 6.4), which will be re-encountered when validating the proposed CIL frameworks in Chapter 7 and Chapter 8.

Table 6.1: Formal definition of notation used throughout Part II of the dissertation.

Symbol	Definition
$t \in \{0, 1, 2, ..., T\}$	Task index
$\mathcal{X}_t$	Input space of the t-th task
$\mathcal{Y}_t$	Label space of the t-th task
$\mathcal{D}_t = \{\mathcal{X}_t, \mathbb{P}(\mathcal{X}_t)\}$	Domain of the t-th task
$\mathbb{P}(\mathcal{X}_t, \mathcal{Y}_t)$	Distribution of the t-th task
$\mathcal{C}_t$	Class space of the t-th task
$\mathcal{C}_{0:t-1}$	Class space up to the t-th task ($t > 0$, t excluded)
$\mathcal{T}_t$	Training set of the t-th task

6.2 Problem Formulation

Continual learning (CL) under the task shift perspective could be regarded as a particular case of transfer learning, where the target distribution changes at every incremental step and the model should perform well on all the tasks observed up to the current point. It is also connected to the domain adaptation problem but, in this case, the focus is devoted toward data annotations, whose distributions change over time and their number may be increased as well (*i.e.*, more classes to be recognized).

On the other hand, we remark that the focus of domain adaptation, as discussed in the previous chapters, was devoted toward the input domain distributions where, typically, a single domain shift is experienced. Hybrid approaches combining DA and CL are emerging to overcome the need for multiple changes in domains and tasks [129–131]. Nonetheless, this enhanced learning setting is still mostly unexplored for incremental semantic segmentation. We will provide a discussion on this subject and then propose a novel approach addressing the hybrid generalized task and domain setup for semantic segmentation in the last part of the dissertation (*i.e.*, Part III).

In **class incremental learning**, tasks are received one at the time and training is performed on the available training data. We remark that in CIL *tasks* are specifically intended as solving a *classification problem*. In the next chapters, we will focus on task incremental learning under the class-incremental perspective. In addition, CIL represents a mitigation of the true generalized continual learning system where data distribution is shifting both in terms of input and output variables, which is more likely to be encountered in practice [127]. The learned model is updated to recognize new classes, whilst preserving knowledge about previous ones. An overview of this setup is reported in Figure 6.1. Key notation is reported and defined in Table 6.1. In more formal terms, we consider the t-th incremental step (with

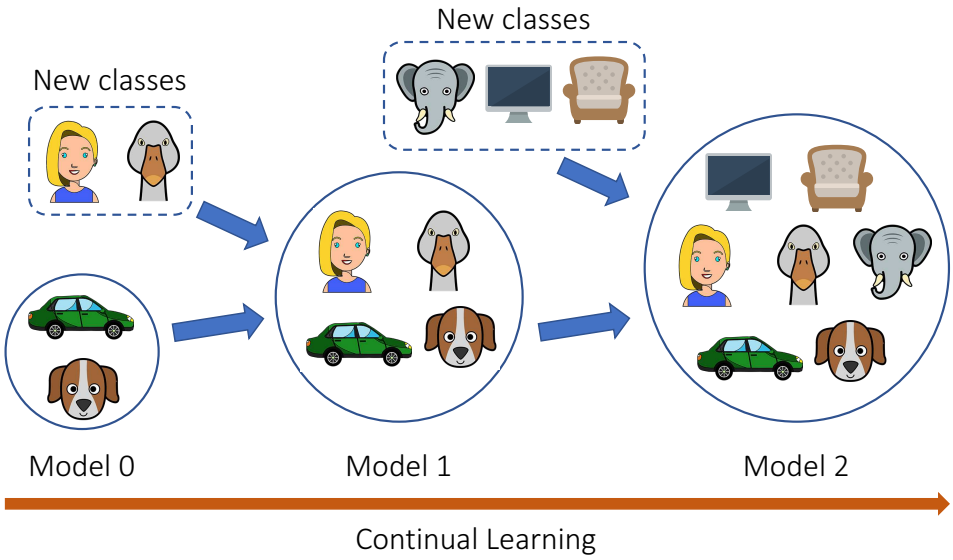


Figure 6.1: Graphical representation of the class incremental continual learning framework. The model is updated to recognize new classes over time without forgetting previously learned ones.

$t = 0, 1, 2, ..., T$) and we are given the previous model M_{t-1} and two sets of data $(\mathcal{X}_t, \mathcal{Y}_t)$ randomly drawn from the distribution $\mathbb{P}(\mathcal{X}_t, \mathcal{Y}_t)$, which is an observation (or subset) of the complete domain $(\mathcal{X}, \mathcal{Y}) \sim \mathbb{P}(\mathcal{X}, \mathcal{Y})$. Here, $\mathcal{X}_t$ denotes a set of data samples for step t and $\mathcal{Y}_t$ denotes the corresponding ground-truth annotations (*i.e.*, a single label for the image classification problem or a dense labeling map for the semantic segmentation problem). In the considered class incremental setup, we assume that each step corresponds to a different learning task. Yet, no constraint is usually posed on domain (*i.e.*, $\mathcal{D}_t = \{\mathcal{X}_t, \mathbb{P}(\mathcal{X}_t)\}$) diversity across steps.

To mimic what happens in many real-world scenarios and to reduce the need for storage or privacy limitations, many frameworks do not store any sample of data $(\mathcal{X}_{t'}, \mathcal{Y}_{t'})$ for any step t' preceding the current step t. Hence, the problem becomes even more challenging as the goal is to control an objective function comprising of all seen tasks without having access to previous samples. More formally, the empirical risk minimization framework translates into the research of the optimal parameters θ^* by optimizing:

$$\text{argmin}_\theta \sum_{t=0}^{T} \mathbb{E}_{(\mathcal{X}_t, \mathcal{Y}_t)} \left[\mathcal{L} \left(M_t \left(\mathcal{X}_t; \theta \right), \mathcal{Y}_t \right) \right], \tag{6.1}$$

with model's parameters θ, loss function $\mathcal{L}$, T incremental tasks seen so far, and M_t the model function at step t. We remark, however, that this objective function cannot be optimized directly as old samples may not be present at all or may be very limited (depending

on the continual learning scenario, see Section 6.4.1). We refer to the case in which all samples are available from the beginning as *joint training*, representing an upper bound of the performance of a continual learning system (*i.e.*, a single stage of training with all samples).

Furthermore, it is useful to gain insight of the problem in terms of marginal output and input distributions, *i.e.*, $\mathbb{P}(\mathcal{Y}_t)$ and $\mathbb{P}(\mathcal{X}_t)$ respectively, of a generic step t. In general, task incremental learning considers that $\mathbb{P}(\mathcal{Y}_{t+1}) \neq \mathbb{P}(\mathcal{Y}_t)$ due to the task output spaces differing over time, *i.e.*, $\mathcal{Y}_{t+1} \neq \mathcal{Y}_t$. On the other hand, incremental input domains $\mathcal{D}_t$ are assumed to not vary across steps, *i.e.*, all input samples belong to the same domain but their task supervision in changing. In Domain Adaptation, instead, input and label distribution roles are generally inverted, being $\mathcal{D}_S \neq \mathcal{D}_T$ and $\mathcal{Y}_S = \mathcal{Y}_T$ (see Section 2.2). In addition, in standard DA there is no continual learning, as adaptation is performed in a single phase with all domains simultaneously available. Nonetheless, recently some works have addressed dynamic adaptation setups, with continually changing input distributions [132–134].

Finally, we remark that ideal continual learning setups consider infinite and continuous stream of training data and at each step the system receives some new samples drawn non-i.i.d. from the current distribution $\mathbb{P}(\mathcal{X}_t, \mathcal{Y}_t)$ that could itself experience sudden or gradual changes with no notification. This is what methods developed in the future should aim to tackle and realize.

6.3 Continual Task Learning Techniques

6.3.1 Class Incremental Learning

Class incremental learning in the computer vision field has been investigated primarily in relation to the image classification task, which enables more fundamental research due to its simplicity if compared with more challenging tasks as semantic image segmentation, which introduces additional complexities to achieve dense semantic recognition. It is possible to group most of the works present in literature into three major families, according to how the task specific information is captured and reiterated during the incremental learning process [135], *i.e.*, (i) *Replay* methods, (ii) *Regularization-based* methods and (iii) *Parameter isolation* methods. We remark that categorization is not mutually exclusive, as different techniques could be jointly used to reach improved performance.

Replay Methods

Replay methods rely on re-introducing data samples of former tasks when novel tasks are learned, in order to mitigate catastrophic forgetting. Training data from past learning steps

can just be stored and rehearsed, or pseudo-samples can be retrieved resorting to generative models.

Rehearsal methods [112, 136–139] are focused on selecting and storing data samples (exemplars) in a memory bank, which is usually assumed to have a fixed size. For example, the popular iCaRL [112] saves exemplars which are closest to the feature means of their classes. Other methods have improved the exemplar selection process, such as in [138], where a reservoir-based more efficient sampling scheme is used. The stored samples are then used to retrain the prediction model on them in later learning steps, while jointly performing optimization on new tasks with the novel training data.

Constrained methods [140–143] leverage the memory bank of stored exemplars to constrain the optimization of new tasks. By just retraining on a limited set of training data from the past, it is likely to end up overfitting on these data. Therefore, by a constrained optimization the goal is to learn the new task with minimal interference with respect to past tasks. This could be achieved, for instance, by looking at gradient directions generated for new and past data [140, 142].
The major issue with exemplar-based class incremental learning methods is that they may not scale well with a large number of classes experienced sequentially or when progressing through multiple learning steps, especially if the memory size is kept fixed. Moreover, they may be limited by privacy concerns, since practical applications could require that raw input data samples are not shared across multiple learning phases.

Pseudo Rehearsal methods [144–152, 242] propose to generate rather than store and rehearse samples of old tasks (at input or feature level). These replay approaches rely on generative models typically trained on the same data distribution (as the main prediction task), and those models are later used to generate artificial samples injected into the training process to preserve previous knowledge. Generative models are usually GANs [144, 147, 149] or auto-encoders [148]. Despite not necessarily requiring exemplars from past training sets, most of the aforementioned pseudo rehearsal approaches introduce novel challenges related to a complex generative side process, which should sustain the same continual learning procedure as the main prediction task.

Proposing a simpler solution without an additional generative framework, in [151] class prototypes were used to model feature distribution for past learning steps, and then inject past knowledge into the current learning step via a sampling process in the latent space according to that distribution. By following this approach, no exemplars have to be stored and replayed. Although showing promising results, [151] fail to capture the representation drift that is present while incrementally training models. This is because old task feature representations, which are computed when corresponding data is available and then kept fixed for the rest of the training, are getting constantly staler and more outdated as the learning

progresses and models are updated.

A different work [152] proposes to dynamically estimate the change of feature prototypes of old tasks while learning new tasks. However, they do not directly capture the relationship between semantic representations of old and new classes, rather focusing on estimating the change of drifts across new and old class sets and neglect the joint evolution of model features during single task training (*i.e.*, they treat feature representation drift independently for each feature channel). Furthermore, they limit their scope to embedding learning and devise a non-learnable module to estimate prototype shift.

In Chapter 7 we will describe our work [242] based on modeling representation drift within the feature space, and on replaying feature representations of past tasks generated according to an up-to-date estimate of the feature distribution. The proposed class incremental method does not rely on memory exemplars from past learning steps. At the same time, it allows to replay data sampled from an updated distribution obtained via a lightweight generative mechanism that does not involve complex side generative models, which can also suffer from catastrophic forgetting.

Regularization Based Methods

This class of approaches introduces additional optimization constraints to preserve knowledge of past tasks, leveraging on the training data that is currently available.

Data-focused methods [114, 153–164] mainly exploit previous model representations to inject knowledge of previous tasks when learning the new one. These works are inspired by the popular *knowledge distillation* mechanism [165], where, in its original form, previous model outputs are used as soft labels in an additional optimization constraint. Diverse distillation schemes have been proposed to tackle incremental image classification, targeting feature representations [159], applied task-wise [160], based on an attention maps [158], low-dimensional manifolds [162] or causal frameworks [163], or combined with self-supervised representation learning [161, 164]. More details will be provided in Section 6.3.2, with a special focus on incremental semantic segmentation, where the distillation mechanism has been extended successfully.
Nonetheless, the distillation approach may suffer from domain shift, since the previous model is fed with the currently available training data that it has never experienced before, which could lead to noisy pseudo supervision introduced by the distillation objective. This issue will be addressed in detail in Part III of the dissertation, where a generalized incremental learning framework under domain and task shifts will be investigated.

We finally remark that most of the successful CIL methods in this category use exemplars of old classes $\mathcal{C}_{old}$ to aid the distillation scheme in rehearsing past knowledge [114, 157–163],

even if not strictly required by the distillation method in its original form.

Prior-focused methods [166–171] focus on estimating a distribution over model parameters across multiple tasks. Parameter relevance is thus investigated for incremental tasks, and then used as prior when undertaking a novel learning step. Weight importance drives the learning process, in order for changes of important parameters to be penalized. Very popular is the EWC method [166], which slows down learning on certain weights based on how important they are to previously seen tasks, and does so by exploiting the Fisher information matrix.

Parameter Isolation Methods

This line of research is populated by works proposing to assign different sets of model weights to different tasks. By freezing previous parameters and growing new model branches to accommodate for new task knowledge, catastrophic forgetting can be effectively counteracted.

Fixed Network methods [172–175] develop static architectures, where task specific fixed sections of the prediction model are disjoint. For example, PackNet [172] resort to iterative pruning and re-training, by which multiple tasks are progressively associated with disjoint sets of model weights. This allows to avoid catastrophic forgetting, as long as the original model has space for a new task to be *packed*. The task-specific fixed-network solution has the downside of needing a task oracle at test time to select which task (*i.e.*, model branch) to activate in order to perform model prediction over an input sample.

Dynamic Architecture methods [176–182] instead propose dynamically changing networks, which evolve during the incremental learning process, by for instance growing new branches [179, 180, 182]. A recent method by Yan *et al.* [182] introduces dynamically expandable representations to achieve better stability-plasticity trade-off. They freeze and expand the feature extractor with additional feature dimensions to accommodate for new visual concepts to be acquired by the prediction model, while they control model growth by a channel-wise pruning scheme for efficient incremental increase of the model size.

6.3.2 Incremental Semantic Segmentation

Even if many of the works detailed in the previous section propose techniques which could in principle be generalized to various vision tasks (such as the popular knowledge distillation mechanism [113, 115, 165]), when facing the semantic segmentation additional complexity, which are not present in case of whole-image classification, must be taken into account. In this section we are going to review the main methods to tackle task-incremental semantic segmentation grouped by employed technique.

Knowledge Distillation

The first family of approaches we present is the most commonly employed one thanks to its simplicity and efficacy, *i.e.*, knowledge distillation. This technique was originally proposed by [183] and [165] to preserve the output of a complex ensemble of networks when adopting a simpler network for more efficient deployment. The idea was adapted to maintain unchanged the responses of the network on the old tasks whilst updating it with new training samples typically associated to new tasks. This is typically performed applying a constraint (*e.g.*, a loss function) in order to mimic the responses of the previous model in the current one. Its main effect is to act as a powerful regularization term during the learning process of the current classes, often leading to better performance on both previous and current classes (by preserving the capability of recognizing the former set and avoiding the overestimation of the latter set).

Knowledge distillation has been explored in different setups and it is somehow a prerequisite for successful task incremental learning algorithms. In sparse tasks, many algorithms use knowledge distillation in different flavors [114, 153–164] (we refer to Section 6.3).

These techniques have been found to be extremely effective and reliable also in dense tasks. [116] extend the image classification model of [113] to segmentation simply constructing a knowledge distillation loss as the cross entropy between previous and current model's output probabilities. The authors also devise a strategy to select relevant samples of old data for rehearsal, that improve the performance, but violate the assumption used in many scenarios of avoiding previous data storage. [117] apply knowledge distillation via cross entropy between previous and current model's output probabilities for each class, as the model predicts binary segmentation maps for each class separately. [118] evaluate on a standard semantic segmentation benchmark and propose to apply knowledge distillation not only at the output level but also at the intermediate feature space to preserve the geometrical relationships of the extracted features. The work is extended in [119] that introduces and compares many knowledge distillation techniques. In particular, distillation on the output layer is enriched by temperature scaling (*i.e.*, rescaling softmax probabilities by a so-called temperature factor) to consider also the uncertainty of the estimations of previous models. Distillation on intermediate feature level is extended to multiple decoding stages and a scheme inspired by Similarity-Preserving Knowledge Distillation [184] is also proposed. [3] propose a revisited distillation loss on the output level which accounts for the fact that a previous model could have already seen previous classes labeled as *background*. [4] propose a masked and weighted distillation loss on the output level to improve the accuracy on small or under-represented classes within the dataset. [120] applies a matching distillation to retain both long-range and short-range statistics at different feature levels and at different scales between the old and

current model by pooling representations. [121] introduce a measure of task similarity as a weighting factor in the distillation objective. [122] resort to a structured self-attention approach for preserve relevant knowledge. Finally, [123] extend the popular contrastive learning paradigm to incremental semantic segmentation to improve class discriminability in the feature space.

Parameter Freezing

One of the major achievements of the early connectionist works is that they identified one main strategy to address catastrophic forgetting: *i.e.*, by freezing part of the network weights [112]. This technique has been applied by a large number of contemporary approaches as a regularization attempt to prevent knowledge degradation caused by upcoming tasks. For instance, [115] experiment on freezing either all the layers (except for the last) or part of them. [185] tries to identify and freeze a compact subset of features (nodes) in the hidden layers, that are crucial for the current task, thus preventing forgetting in the future. Similarly, [166] remember old tasks by slowing down the learning process on the relevant weights for those tasks. [154] try to maintain the performance on old tasks by freezing the final layer and discouraging the change of shared weights in feature extraction layers.

Also in the dense labeling task, parameter freezing has been proposed as a way to prevent forgetting. [118] propose to freeze all the layers of the encoder in order to preserve unaltered the feature extraction capabilities and only train the decoding parameters. [119] propose to freeze only the first couple of layers of the encoder, to preserve the most task-agnostic part of the feature extractor. However, the choice of which layers to freeze remains an open question and there is an intrinsic trade-off between the capability of efficiently learning new tasks and the preservation of the acquired knowledge. A first attempt of automatic selection of which layers to freeze has been recently introduced by [186] checking the most plastic layers of the network.

Geometrical Feature-Level

The analysis of the latent space organization is becoming crucial towards understanding and improvement of deep neural networks [94, 95, 187, 188]. Recently, some attention has been devoted to latent regularization in continual image classification [189, 190] and in unsupervised domain adaptation [239, 240]. The key idea of these approaches is to disentangle the intermediate feature space in different ways, to space apart features of different classes. In continual learning this can reduce the overlap when future classes are introduced in the model.

Moving to dense tasks, we find the work of Michieli *et al.* [2], where the latent space is constrained to reduce forgetting whilst improving the recognition of novel classes. The framework is driven by three main components: first, prototype matching enforces latent space consistency on old classes, constraining the encoder to produce similar latent representation for previously seen classes in the subsequent steps; second, feature sparsification allows to make room in the latent space to accommodate novel classes; third, contrastive learning is employed to cluster features according to their semantics while tearing apart those of different classes. More recently, Yang *et al.* [123] resort to a contrastive paradigm to enforce similarity between latent representations of all the pixels from the same classes, and space apart those corresponding to pixels from different classes. In order to alleviate catastrophic forgetting, representations of the new model and representations extracted by a frozen model from the previous learning step are jointly exploited in a contrastive scheme.

New Directions

Other novel ideas have been proposed in the continual learning literature both for dense and sparse tasks. Here, we present and discuss some of the most promising research directions.

Weights Initialization has been employed in [3] to deal with the atypical behavior of the background class in the *disjoint* and *overlapped* scenarios. The authors initialize the classifier's parameters for the novel classes in such a way that the probability of the background is uniformly spread among the novel classes, preventing the model to be biased toward the background class when dealing with unseen classes.

Generative Replay methods train generative models on the current data distribution; afterwards, it is possible to sample data from past experience when learning on new data (we refer to Section 6.3). By learning on actual data mixed with artificially generated past data, they try to preserve past knowledge while learning the new task. The generative model is generally a GAN [31] as in [147] and [144] or an auto-encoder as in [191] and [148].

In Chapter 8, we propose RECALL [241], an incremental learning framework for semantic segmentation where two kinds of generative replays are exploited: either resorting to a standard pre-trained GAN or to web-crawled images to avoid forgetting, without storing any of the samples related to previous tasks. When using the generative model, differently from previous works on continual image classification, we do not select real exemplars as anchor to support the learned distribution [149] nor we train or fine-tune the GAN architecture on the current data distribution [144, 147, 149], thus reducing memory and computation time. In particular, only the weak classification labeling is available for generated data and some pseudo-labels need to be estimated for segmentation. We will detail the devised approach to retrieve both image data and dense annotations for past tasks.

Webly-based Learning models [192, 193], *i.e.*, models that learn from samples acquired via web searches, could be an extremely powerful tool to retrieve faithful past examples using as queries the label names of the old classes to preserve. This approach is proposed in RECALL [241] as an alternative to GAN-based image generation, where novel samples are interleaved with web-crawled ones (querying the class names to drive the search). This simple method outperforms by a large margin competing approaches with the (minimal) requirement of searching for pictures on the Web. Also in this case, only weak classification labels are available and a pseudo-labeling scheme needs to be introduced.

6.4 Experimental Setups

6.4.1 Task Incremental Setups

Continual task learning in class incremental fashion comes in different flavors. In particular, existing works differ in the consideration of the domain distributions $\mathbb{P}(\mathcal{X}_t, \mathcal{Y}_t)$ and of the data sampling $(\mathcal{X}_t, \mathcal{Y}_t)$. The different choices emerge from different target applications. Let us denote with $\mathcal{C}_{0:t\text{-}1}$ the previous label set, which is expanded with a set of new classes $\mathcal{C}_t$ at step t, yielding a new label set $\mathcal{C}_{0:t} = \mathcal{C}_{0:t\text{-}1} \cup \mathcal{C}_t$.

As typically assumed in task incremental settings, the sets of new labels discovered at each step are disjoint. Nonetheless, a different discussion about incremental class progression should be made for global and dense classification problems. While for whole-image classification the generation of incremental data sets is straightforward once the class order has been determined (due to the one-to-one image-label association), dense prediction problems, such as semantic segmentation, allows for pixels with diverse semantic significance to be found in a single image, leading to data samples with multi-class labels. In addition, there usually exists a special *background* or *void* class that is present in every incremental step, and which behavior and meaning depends on the selected scenario. These features entail that many possible incremental scenarios can be followed in case of dense classification problems, and one of the key differences lies in the way the background class is considered, which is typical of many semantic segmentation benchmarks. The most common settings for continual semantic segmentation are the following:

(i) **Sequential masked.** This setup reflects the simplest idea on continual semantic segmentation; *i.e.*, each learning step contains a unique set of images, whose pixels belong either to novel classes or to a void class, which is not predicted by the model and it is masked out from both the results and the training procedure. This setup has been used in [4, 117].

(ii) **Sequential.** This setup has been proposed in [118, 119]. Each learning step contains a unique set of images, whose pixels belong to classes seen either in the current or in the previous learning steps. At each step, labels for pixels of both novel classes and old ones are present; however, the specific occurrence of a particular old class is highly correlated to the set of classes being added. For example, if the set of all old classes is $\mathcal{C}_{0:t\text{-}1} = \{chair, airplane\}$ and the set of classes being added is $\mathcal{C}_t = \{dining\ table\}$, then it is reasonable to expect that $(\mathcal{X}_t, \mathcal{Y}_t)$ contains some images with the *chair* class, that typically appears together with the *dining table*, while the class *airplane* is extremely unlikely to occur.

(iii) **Disjoint.** This setup has been proposed in [3, 119]. At each learning step, the unique set of images is identical to the sequential setup. The difference with respect to the sequential setup lies in the set of labels. At each step, only labels for pixels of novel classes are present, while the old ones are labeled as background in the segmentation maps (this causes the *background* class to change distribution at each step).

(iv) **Overlapped.** This setup moves from the work of [115] for object detection and has been addressed in [2, 3, 120] for semantic segmentation. In this setup, each training step contains all the images that have at least one pixel of a novel set of classes, with only the classes of the set annotated and the rest set to *background*. Differently from the other settings, in this scenario images may contain pixels of classes that will be learned in the future, but labeled as *background* in the current step; for this reason, as in the previous setting, the *background* class changes distribution at every incremental step.

A few examples of the different semantic map annotations are given in Figure 6.2. Although being subcategories of the same problem they lead to substantially different setups requiring different strategies to be tackled.

This articulated scenario is getting even more articulated as there exist many different ways of sampling the sets $\mathcal{C}_t$ of unseen classes and of selecting its cardinality $|\mathcal{C}_t|$, leading to completely different experiments. For instance, let us consider one of the most widely used benchmarks for semantic segmentation, *i.e.*, the Pascal VOC2012 dataset [20], which is composed by 21 semantic classes (*background* included). Regarding the first aspect, one possibility is to sort the classes using the order provided by the dataset (*e.g.*, the alphabetical ordering for VOC2012) as done in [2, 3, 115, 118–120]. Another possibility is to sort the classes based on their occurrence inside the dataset [119], to reflect the idea that, in real-world applications, it is more likely to start from common classes and introduce rarer ones later. With respect to the second aspect, one may add a single class, a batch of classes or multiple classes sequentially one after the other [2, 3, 118–120]. All these possibilities open up a very varie-

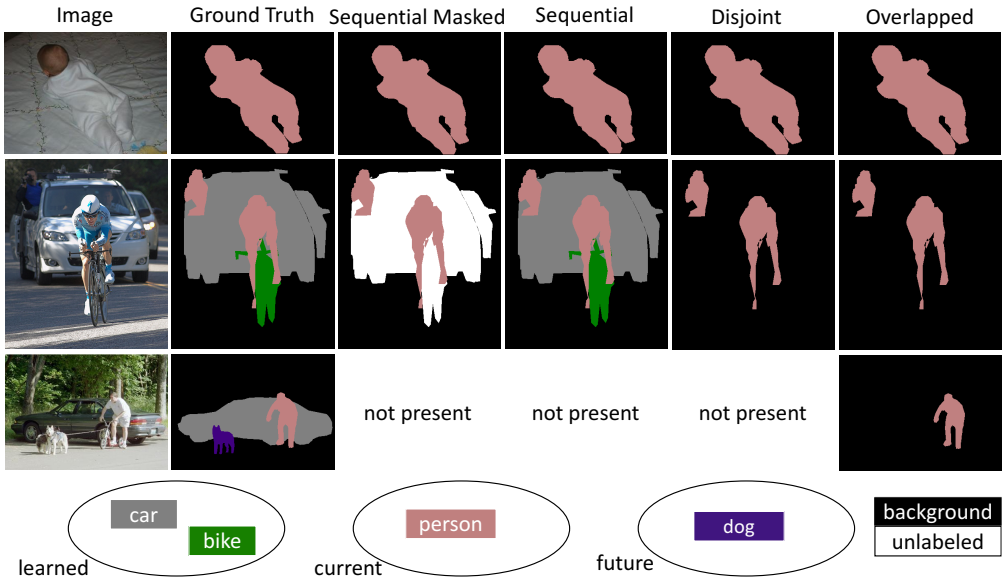


Figure 6.2: Overview of the different setups for class incremental continual learning in semantic segmentation. The black class represents the $background$ class and the white one represents the *void/unlabeled*.

gate picture, which is being explored only recently, hence many research directions remain still unexplored.

6.4.2 Employed Datasets

In this section we describe the experimental benchmarks commonly used in class incremental learning, which will be exploited to validate techniques presented in Chapters 7 and 8.

Image Classification

There exist multiple standard CIL benchmarks, such as the CIFAR100 [194], TinyImageNet [195] and CUB200-2011 [196] datasets. In addition, the large-scale ImageNet-Subset benchmark [26] is commonly employed to validate incremental frameworks in a more challenging setup. To simulate the task incremental framework different class incremental protocols can be used. Generally, first the training is performed on large set of classes, such as half of the available semantic classes, in order to get a robust model before undertaking the incremental phases. Then, the remaining class set is evenly divided into multiple steps (such as 5, 10 or 20 incremental steps), each with an equal number of new classes being introduced.

Semantic Segmentation

One of the most widely used benchmarks for incremental semantic segmentation is the Pascal VOC2012 dataset [197]. It contains 11,530 training and validation images of a variable size and with semantic labels corresponding to 20 classes plus background. Disjoint, overlapped and sequential experimental settings are commonly analyzed [3, 115, 118, 119]. As discussed in Section 6.4.1, in the first setting each learning step contains a unique set of images, whose pixels belong to classes seen either in the current or in the previous learning steps. In other words, each image is seen in only one learning step. On the contrary, in the overlapped setting, a few classes appear on both tasks. Finally, the sequential setup allows to get annotations for past classes if re-occurring in novel step images. Additionally, different class incremental procedures can be investigated, varying the size of class set seen at the first learning step and of number of incremental steps.

7

Continual Learning by Feature-level Replay

7.1 Introduction

As detailed in Chapter 6, continual learning has been extensively studied in a class incremental fashion [108, 127, 198]. In Class Incremental Learning, a model is employed with sequential tasks, where classes to be learned progressively change (Figure 7.1). Yet, the change of distribution of training data in the form of semantic drift (*i.e.*, due to change of experienced class set) leads to forgetting, where the bias towards new data causes past information to be gradually erased and learned representations to be constantly updated (*i.e.*, feature drift) focusing on new tasks. Our goal is to retrieve and replay up-to-date feature-level representations from former tasks without accessing training samples from the past. The contributions in this chapter, introduced to overcome the aforementioned limitations and perform feature-level replay, with focus on the *image classification* problem, can be summarized as follows:

(i) To expound the forgetting phenomena in CIL, we explore the dynamics of incrementally learned classifiers using a probabilistic approach. Our investigation (Figure 7.1) suggests that a source of the forgetting is the evanescence of representations learned using old classes and the unavailability of their distribution in incremental steps.

(ii) To revive the evanescent representations (ERs), we devise a framework which enables to model different types of representation drifts modularly. In the framework (Figure 7.1, 7.2 and 7.3), we first define the change of feature representations by feature drift (*i.e.*, due to constantly evolving feature representations of different patterns learned from data in CIL) and propose an effective method to model it. Next, we de-

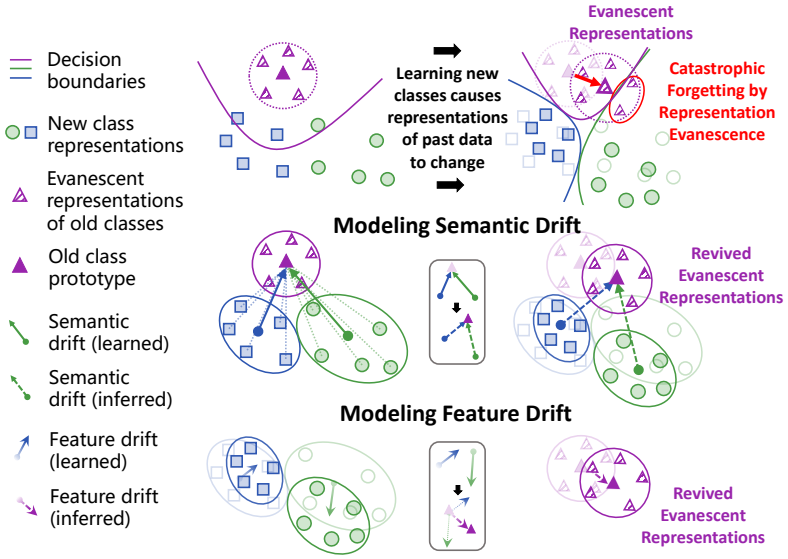


Figure 7.1: In CIL, the process of training models on new classes causes representations of past categories to constantly change. Yet, unavailability of data of former classes prevents from tracking their evolution in feature spaces, leading to *evanescence* of their representations and, in turn, to catastrophic forgetting. We propose to model representation drift on a semantic level (*i.e.*, the relationship among novel and past classes) and on a feature level (*i.e.*, the combined evolution of features learned by a classification model), and exploit it to infer up-to-date representations of past classes. By injecting old-class knowledge into the learning process, we counteract forgetting.

fine the change of representations of classes by semantic drift (*i.e.*, due to the change of semantic categories learned at different incremental steps) and propose an effective method to model it.

(iii) We propose to train semantic and feature drift models together with feature learning and classification models. The devised method integrates learning and inference in training: it is used to estimate old-class distributions, to be exploited for preserving knowledge of former classes, while learning new representations for new classes.

(iv) In the experimental analyses, our methods outperform state-of-the-art exemplar-free competitors on various benchmarks. We also provide a detailed ablation study of geometric and statistical properties of drift models. Our experimental results explicate the nontrivial relationships between accuracy of models and distribution of evanescent and revived representations in class incremental learning.

The chapter is organized as follows: first the CIL problem under representation learning perspective will be formally introduced (Section 7.2); next, the core of the proposed CIL strategy will be described, *i.e.*, we will discuss how to model representation drift, both in its feature and semantic forms, in order to estimate the feature distribution of former classes

Table 7.1: Formal definition of key notation used throughout Chapter 7.

Symbol	Definition
$t \in \{0, 1, 2, ..., T\}$	Step index
$n \in \{0, 1, 2, ..., N\}$	Stage index
$\mathcal{T}_t$	Training set at step t
$\mathcal{F}_{old}$	Set of *old class* feature representations at the current step
$\mathcal{F}_t^0$, $\mathcal{F}_t^n$	Set of *new class* feature representations at step t, respectively at stages 0 and n
$\Pi_{old}^{t,0}$, $\Pi_{old}^{t,n}$	Set of *old class* feature prototypes at step t, respectively at stages 0 and n
Π_{new}^t	Set of *new class* feature prototypes computed at the end of step t
f_{θ_t}	Feature extractor at step t
h_{ϕ_t}	Classifier at step t
$g_t = h_{\phi_t} \circ f_{\theta_t}$	Classification model at step t
$\Gamma_{\gamma_t^n}$	Feature drift model at step t and stage n
$\Psi_{\psi_t^n}$	Semantic drift model at step t and stage n

(Section 7.3); we will additionally detail the overall incremental training, that merges drift modeling to the image classification task learning, and the core architectures underlying drift models (Section 7.4 and 7.5); lastly, we will provide extensive experimental studies to validate the proposed framework (Sections 7.6 and 7.7).

7.2 Problem Formulation

In the considered learning setting, for each incremental training task and step t, the training set is composed of images belonging to the current class set $\mathcal{C}_t$, whereas past semantic categories $\mathcal{C}_{old} \triangleq \{\mathcal{C}_{t'}\}_{t'=0}^{t-1}$ lack any training sample. The goal of the model is to maximize the generalization (classification) accuracy on all the classes observed up to the current step.

More formally, at each step $t \in [T] = \{0, 1, \ldots, T\}$ of CIL, we are given a dataset $\mathcal{T}_t = (\mathcal{X}_t, \mathcal{Y}_t)$, where $\mathcal{X}_t = \{\mathbf{X}_{t,j}\}_{j=1}^{N_t}$ is the set of RGB image samples and $\mathcal{Y}_t = \{y_{t,j} \in \mathcal{C}_t\}_{j=1}^{N_t}$ is the set of their labels. $\mathcal{C}_t$ is the set of class labels observed at this step, and $\mathcal{C}_t \cap \mathcal{C}_{t'} = \emptyset, \forall t \neq t'$. Popularly employed CIL models for whole-image classification [151] are composed of a feature extraction module f_θ and a classifier h_ϕ with parameters $\theta \in \Theta$ and $\phi \in \Phi$. At each t-th step, training is performed by solving:

$$\arg\min_{\theta_t, \phi_t, \epsilon} \mathcal{L}_{cc}^t + \sum_{t'=0}^{t-1} \epsilon_{t'}, \tag{7.1}$$

where $\mathcal{L}_{cc}^{t} \triangleq \mathcal{L}(\mathcal{T}_t; \theta_t, \phi_t)$ is the expected loss of the full classification model $g_t = h_{\phi_t} \circ f_{\theta_t}$ on $\mathcal{T}_t$ at step t, while $\epsilon_{t'}$ controls the *forgetting* of representations of old classes [151]. In other words, we would like to optimize $\mathcal{L}_{cc}^{t}$ in order to perform well on the new set of classes, still retaining the satisfactory performance on old tasks that should have been previously achieved. The $\epsilon_{t'}$ term, in fact, is supposed to measure the performance deviation suffered by the model from the configuration attained in past steps relative to the old classes.

To address the classification optimization problem, we can identify different strategies. On one side, generative classifiers implementing h_ϕ optimize Eq. (7.1) to model the joint probability distribution $p(C, F; \mathcal{P})$ where $\mathcal{P} = \Theta \cup \Phi$. On the other side, discriminative classifiers, such as softmax classifier, instead optimize Eq. (7.1) to model the conditional distribution $p(C|F; \mathcal{P})$, defining the loss by a function (*e.g.*, a cross-entropy objective) of $p(C|F; \mathcal{P})$. We will focus on the second category.
In general, however, when dealing with the classification task in an incremental fashion, methods aim to model $p(C, F|\mathcal{P}_t)$ without using $\{\mathcal{T}_{t'}\}_{t'=0}^{t}$ at step t, where

- $\mathcal{P}_t = \Theta_t \cup \Phi_t$, $\Theta_t = \{\theta_{t'}\}_{t'=0}^{t}$ and $\Phi_t = \{\phi_{t'}\}_{t'=0}^{t}$ are the model's current set of parameters*,
- $F \in \mathcal{F} = \mathcal{F}_t \cup \mathcal{F}_{old}$ is the random variable taking values from the set of feature representations $\mathcal{F}_t$ learned at step t on current training data $\mathcal{T}_t$ and from old representations $\mathcal{F}_{old} = \{\mathcal{F}_{t'}\}_{t'=0}^{t-1}$, and
- $C \in \mathcal{C} = \mathcal{C}_t \cup \mathcal{C}_{old}$ is a random variable of semantic (class) representations, where $\mathcal{C}_{old} = \{\mathcal{C}_{t'}\}_{t'=0}^{t-1}$.

Deep learning models have been employed to model probability distributions implicitly by learning representations of old and new classes, and make predictions for new classes:
(i) class posterior probabilities for old ($p(C \in \mathcal{C}_{old}|\bullet)$) and new classes ($p(C \in \mathcal{C}_t|\bullet)$) are computed using classifiers $h_{\Phi_{t'}}, \forall t' \in \{0, 1, \ldots, t\}$;
(ii) feature representations $\mathcal{F}_t$ of new classes $\mathcal{C}_t$ are learned by updating the feature extractor $f_{\Theta_{t'}}, \forall t' \in \{0, 1, \ldots, t\}$.

Evanescent Representations At the t-th step, we do not have access to old samples $\mathcal{T}_{old} = \{\mathcal{T}_{t'}\}_{t'=0}^{t-1}$. Thus, features $\mathcal{F}_{old}$ cannot be extracted from $\mathcal{T}_{old}$, leaving us with no direct way to minimize Eq. (7.1) by modeling feature distributions for old and new classes jointly.
To address this problem and bring the *evanescent representations* $\mathcal{F}_{old}$ to life, we exploit class prototypes $\pi \in \Pi_{old}$, which we obtain by computing the class-wise mean of features [151].

*In some CIL methods, models trained at earlier steps $t' < t$ are frozen and re-used at consecutive steps, while the other methods incrementally update the models at each step. In the latter case, $\mathcal{P}_t = \{\theta_t\} \cup \{\phi_t\}$.

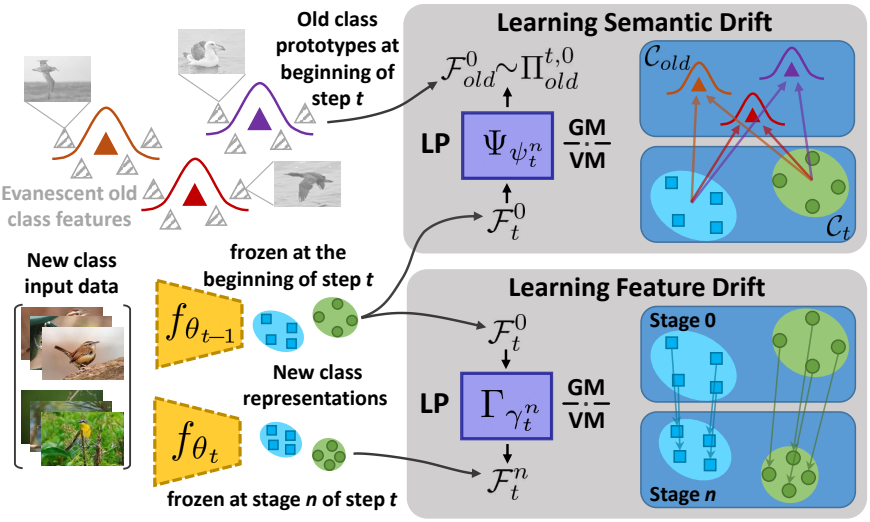


Figure 7.2: Illustration of the learning phase (LP) of semantic and feature drift models. SD (top): we model the relationship between representations of new and old classes at the beginning of the incremental step. FD (bottom): we capture the evolution of representations of novel classes within the incremental step.

We leverage prototypes at the beginning of an incremental step to model old-class feature distribution $p(F \in \mathcal{F}_{old} | \pi_c \in \Pi_{old})$ (denoted by $\mathcal{F}_{old} \sim \Pi_{old}$ in Figure 7.2 and Figure 7.3). Then, we update $p(F \in \mathcal{F}_{old})$ throughout the incremental step by modeling the representation drift to *revive* evanescent representations. In the next section, we will provide a detailed description of the proposed approach.

7.3 Modeling Representation Drift

Let $\mathcal{F}_t^0$ denote the set of features extracted using a feature extractor f_{θ_t} at the beginning of the incremental step t, and $\mathcal{F}_t^n$ denote the set of features extracted using f_{θ_t} updated with $n > 0$ optimization stages from the dataset $\mathcal{T}_t$. Since only $\mathcal{T}_t$ is available at the step t, $\mathcal{F}_t^0$ and $\mathcal{F}_t^n$ contain only representations of new classes $\mathcal{C}_t$. Similarly, $\Pi_{old}^{t,0}$ and $\Pi_{old}^{t,n}$ are the set of semantic representations (prototypes) of old classes $\mathcal{C}_{old}$, respectively available at the beginning of the step t and updated at the n^{th} stage ($n > 0$) of the same step. Moreover, we remark that storing prototypes is very memory efficient and compliant to privacy requirements [151], since a very limited amount of processed data has to be stored (*i.e.*, comparable to the size of a classifier).

We propose modeling the drift of feature and semantic representations using two models: (i) Γ_γ parameterized by γ, and (ii) Ψ_ψ parameterized by ψ in two phases:

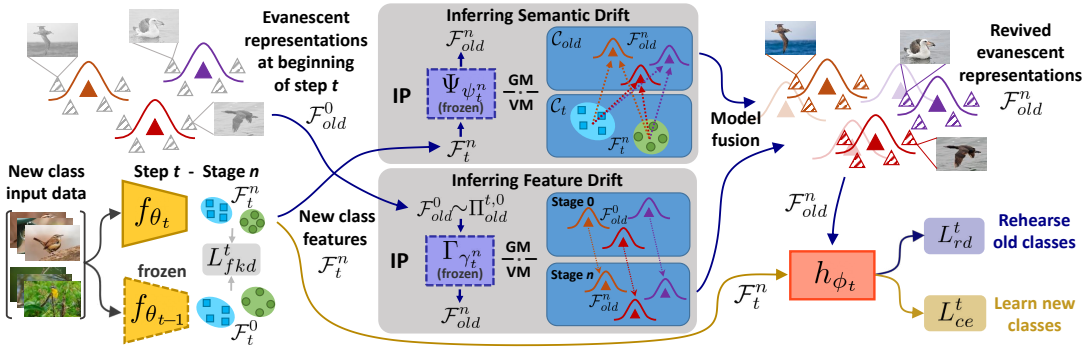


Figure 7.3: An outline of our CIL framework augmented with the proposed representation drift models. A classifier is trained on the dataset $\mathcal{T}_t$ at step t (L^t_{ce}). Feature knowledge distillation (L^t_{fkd}) is used to reduce feature drift. We exploit the drift models learned in LP (Figure 7.2) to infer revived evanescent representations (RERs), which are leveraged by L^t_{rd} to inject past knowledge into the current training procedure.

1. In the **Learning Phase (LP);** the parameters γ and ψ are optimized to estimate the relationship among representations available at an incremental step t (Figure 7.2).

2. In the **Inference Phase (IP);** the learned models Γ_γ and Ψ_ψ are used to infer $\mathcal{F}^n_{old}$ which are the *evanescent representations revived at the step* t (Figure 7.3).

The following subsections detail the methods proposed for modeling drifts. To identify and train Γ_γ and Ψ_ψ, we propose to leverage Gaussian (GM) and Variational (VM) models parameterized by deep neural networks (DNNs). Although Gaussian processes can be used for VMs [199], we consider GMs and VMs individually to explicate variational structure of VMs. We will detail the training algorithms we use to learn these models in Section 7.4. In addition, the analyses given in Section 7.7 will show that Gaussian models can provide a more robust accuracy, since they do not suffer from pathologies of variational models. [200, 201]. On the other hand, variational models can provide higher accuracy when limited data is available, *i.e.*, for small $\mathcal{T}_t$.

7.3.1 Modeling Feature Drift

Learning Phase

We aim at modeling the feature drift (FD) experienced on current data $\mathcal{T}_t$ as representations revive and evolve throughout the present step $t > 0$, as depicted in Figure 7.2. In particular, at stage $n > 0$, we extract $\mathcal{F}^n_t$ and train $\Gamma_{\gamma^n_t}$ to learn the relationship between $\mathcal{F}^0_t$ and $\mathcal{F}^n_t$, *i.e.*, between new-class representations captured at different stages.

GM: To employ Gaussian models, we first identify $\Gamma_{\gamma_t^n} : \mathcal{F}_t^0 \rightarrow \mathcal{F}_t^n$ by a DNN, *e.g.*, a multilayer perceptron (MLP). Then, $\Gamma_{\gamma_t^n}$ is trained to track and model the evolution of revived evanescent representations (RERs) from stage 0 to n.

VM: We consider $\Gamma_{\gamma_t^n}$ as a stochastic map and identify it by a variational model such as a variational auto-encoder (VAE). The variational model is trained by maximizing the likelihood $p(F \in \mathcal{F}_t^n | F \in \mathcal{F}_t^0; \gamma_t^n)$. Thereby, we can statistically model the feature drift across different stages $[n]$ at a given step t.

Inference Phase

In the learning phase, the model $\Gamma_{\gamma_t^n}$ is trained to learn the drift experienced by features associated to $\mathcal{C}_t$. In the inference phase (Figure 7.3), instead, we exploit the trained $\Gamma_{\gamma_t^n}$ to infer the feature drift undergone by features of $\mathcal{C}_{old}$ and, consequently, the distribution $p(F \in \mathcal{F}_{old}^n)$ of revived evanescent representations $\mathcal{F}_{old}^n$.

GM: We use $\Gamma_{\gamma_t^n}$ to infer representations of old class prototypes $\Pi_{old}^{t,n}$ at stage $n > 0$ under the feature drift. That is, $\Gamma_{\gamma_t^n}$ enables us to directly track the trajectory of Π_{old}^t from stage 0 to n, and thus map $\Pi_{old}^{t,0}$ to $\Pi_{old,f}^{t,n}$. If $n = 0$, then $\Gamma_{\gamma_t^0}$ is an identity map. Finally, we approximate the distribution $p(F \in \mathcal{F}_{old}^n)$ of revived evanescent representations at stage n by a Gaussian distribution $p(F \in \mathcal{F}_{old}^n; \pi_c \in \Pi_{old,f}^{t,n}) \sim \mathcal{N}(\pi_c, \sigma_c)$. For both Gaussian and variational models, the standard deviation σ_c is estimated at step t' when $c \in \mathcal{C}_{t'}$, and kept fixed at every step $t'' > t'$.

VM: The trained model $\Gamma_{\gamma_t^n}$ provides an approximation of $p(F \in \mathcal{F}_{old}^n | F \in \mathcal{F}_{old}^0)$. At stage $n = 0$, we resort to $p(F \in \mathcal{F}_{old}^0)$, as no feature drift has to be estimated, and we model the distribution of evanescent representations by $p(F \in \mathcal{F}_{old}^0) \propto p(F \in \mathcal{F}_{old}^0; \pi_c \in \Pi_{old}^{t,0}) \sim \mathcal{N}(\pi_c, \sigma_c)$. At $n > 0$, training features are sampled from the distribution $p(F \in \mathcal{F}_{old}^n | F \in \mathcal{F}_{old}^0; \gamma_t^n) \cdot p(F \in \mathcal{F}_{old}^0)$.

7.3.2 Modeling Semantic Drift

Learning Phase

We aim at capturing the semantic drift (SD) experienced by representations at each incremental step. To this end, at the beginning of each step, we extract $\mathcal{F}_t^0$ and train a network $\Psi_{\psi_t^0}$ with parameters ψ_t^0 to model the relationship between $\mathcal{F}_t^0$ and $\mathcal{F}_{old}^0$. We employ prototypes $\pi \in \Pi_{old}^{t,0}$ to model $p(F \in \mathcal{F}_{old}^0) \propto p(F \in \mathcal{F}_{old}^0; \pi_c \in \Pi_{old}^{t,0}) \sim \mathcal{N}(\pi_c, \sigma_c)$. As opposed to Γ_γ, Ψ_ψ captures the semantic drift observed at each new step. Therefore, an individual model Ψ_ψ is optimized at the start, and fixed for the rest of the step (*i.e.*, $\psi_t^n = \psi_t^0$).

When model fusion (see Section 7.4.2) is adopted, instead, Ψ_{ψ} is re-trained once per stage , to account for the drift estimated by Γ_{γ} (Section 7.4.2).

GM: We first identify $\Psi_{\psi_t^n} : \mathcal{F}_t^0 \rightarrow \Pi_{old}^{t,0}$ by a deep neural network (*e.g.*, an MLP). Then, $\Psi_{\psi_t^n}$ is trained to model the semantic drift between representations for classes available at the current step ($\mathcal{C}_t$) and those experienced in the past ($\mathcal{C}_{old}$).

VM: We first approximate the distribution $p(F \in \mathcal{F}_{old}^0)$ of evanescent representations revived at stage $n = 0$ by $p(F \in \mathcal{F}_{old}^0; \pi_c \in \Pi_{old}^{t,0}) \sim \mathcal{N}(\pi_c, \sigma_c)$. Then, a conditional VM (*e.g.*, a VAE) is trained by maximizing the likelihood $p(F \in \mathcal{F}_{old}^0 | F \in \mathcal{F}_t^0; \psi_t^n)$ to learn the representations shared among old and new classes.

Inference Phase

In the learning phase, $\Psi_{\psi_t^n}$ is trained to learn the semantic drift experienced while moving from $\mathcal{C}_{old}$ to $\mathcal{C}_t$. The drift is captured at the beginning of the current step t, when up-to-date representations of both sets are available. We now exploit the trained $\Psi_{\psi_t^n}$ to infer the semantic undergone at every stage $n > 0$, and estimate the distribution $p(F \in \mathcal{F}_{old}^n)$ of evanescent representations.

GM: We use the trained $\Psi_{\psi_t^n}$ to infer representations of old class prototypes $\Pi_{old}^{t,n}, \forall n \geq 0$ under semantic drift. That is, $\Psi_{\psi_t^n} : \mathcal{F}_t^n \rightarrow \Pi_{old,s}^{t,n}$ is trained to estimate the relationship between feature and prototypical representations at stage n. We then approximate $p(F \in \mathcal{F}_{old}^n)$ by $p(F \in \mathcal{F}_{old}^n; \pi_c \in \Pi_{old,s}^{t,n}) \sim \mathcal{N}(\pi_c, \sigma_c)$.

VM: The trained model $\Psi_{\psi_t^n}$ provides an approximation of the feature distribution $p(F \in \mathcal{F}_{old}^n | F \in \mathcal{F}_t^n), \forall n \geq 0$. To generate feature samples to be leveraged for training purposes, we perform inference using the estimated distribution $p(F \in \mathcal{F}_{old}^n | F \in \mathcal{F}_t^n; \psi_t^n) \cdot p(F \in \mathcal{F}_t^n)$, where $\mathcal{F}_t^n$ is provided by the feature extractor f_{θ_t} trained for n optimization stages and applied on samples from $\mathcal{T}_t$.

7.4 Training Models of Representation Drift

7.4.1 Training Classification Models

At each step t, we train $g_t = h_{\phi_t} \circ f_{\theta_t}$ on $\mathcal{T}_t$ using a cross-entropy loss $L_{ce}^t \triangleq L_{ce}(\mathcal{T}_t)$.

When $t > 0$, to mitigate forgetting of previous tasks, we generate features of $\mathcal{C}_{old}$ by modeling the drift and estimating the distribution $p(F \in \mathcal{F}_{old})$. Thus, we compute:

$$L_{rd}^t \triangleq L_{rd}(\mathcal{F}_{old}^{n,t}) = \sum_{F \in \mathcal{F}_{old}^{n,t}} y_F \log h_{\phi_t}(F), \tag{7.2}$$

Algorithm 7.1 Training Models.

Input: $\{\mathcal{T}_t\}_{t=0}^{T}$ (datasets), N (num. of stages per step).
Output: $g_T = h_{\phi_T} \circ f_{\theta_T}$.
Train f_{θ_0} and h_{ϕ_0} with L_{cc}^0, and compute Π_{new}^0.
Initialize $\Pi_{old}^{t=1,0}$ as Π_{new}^0.
for each incremental step $t \leftarrow 1$ to T **do**
  for each optimization stage $n \leftarrow 0$ to $N-1$ **do**
    LP: Train $\Gamma_{\gamma_t^n}$ and $\Psi_{\psi_t^n}$ by solving (7.5).
    IP: Estimate $p(\mathcal{F}_{old}^n)$ and $\Pi_{old}^{t,n}$.
    Train f_{θ_t} and h_{ϕ_t} by solving (7.6).
  end for
  LP: Train $\Gamma_{\gamma_t^N}$ and $\Psi_{\psi_t^N}$ by solving (7.5).
  IP: Estimate $\Pi_{old}^{t,N}$ and Π_{new}^t.
  Initialize $\Pi_{old}^{t+1,0} := \Pi_{new}^t \cup \Pi_{old}^{t,N}$.
end for

where y_F is the one-hot label vector of the class $c_F \in \mathcal{C}_{old}$ and $\mathcal{F}_{old}^{n,t}$ is the set of revived evanescent representations sampled from the estimated distribution using the updated prototypes of $\mathcal{C}_{old}$. The loss L_{rd}^t approximates $\mathcal{L}(\mathcal{T}_{t'}; \theta_t, \phi_t) \leq \epsilon_{t'}$ (Eq. 7.1) of g_t on the previous datasets $\{\mathcal{T}_{t'}\}_{t'=0}^{t-1}$ using their inferred representations.

We enhance the training objective by a distillation loss $L_{fkd}^t \triangleq L_{fkd}(\mathcal{T}_t)$ [158] to reduce the entity of representation drift across incremental tasks. L_{fkd}^t is defined by the ℓ_2 distance between representations extracted from $\mathcal{T}_t$ using f_{θ_t} and $f_{\theta_{t-1}}$, the latter inherited from the previous step and kept fixed. Thereby, L_{fkd}^t approximates the difference between $\mathcal{L}_{cc}^t$ and the loss $\mathcal{L}_{pc}^t$ of the previous model $f_{\theta_{t-1}}$ on $\mathcal{T}_t$. Although $\mathcal{L}_{pc}^t$ is not explicitly defined in Eq. (7.1), it provides information regarding shareability of representations among consecutive steps $t-1$ and t. Therefore, models optimizing L_{fkd}^t can make use of the feature shareability for learning drifts. Then, the overall classification objective L_{cc}^t computed at each step t is $L_{cc}^t = L_{ce}^t + \lambda_{rd} L_{rd}^t + \lambda_{fkd} L_{fkd}^t$, where L_{rd}^t and L_{fkd}^t are used only for $t > 0$ with loss balancing parameters $\lambda_{rd} > 0$ and $\lambda_{fkd} > 0$.

7.4.2 Training Representation Drift Models

The loss functions $L_f^t \triangleq L_f(\mathcal{F}_t^0, \mathcal{F}_t^n; \gamma_t^n)$ and $L_s^t \triangleq L_s(\mathcal{F}_t^0, \Pi_{old}^{t,0}; \psi_t^n)$ denote the objectives used to individually train feature and semantic drift models $\Gamma_{\gamma_t^n}$ and $\Psi_{\psi_t^n}$, respectively. The exact form of the aforementioned objectives depends on the employed network architecture identifying $\Gamma_{\gamma_t^n}$ and $\Psi_{\psi_t^n}$. A more detailed description is provided in Section 7.6.1.

Model Fusion (MF): To estimate $p(F \in \mathcal{F}_{old}^n)$ using **GM** and **VM**, we fuse the output of $\Gamma_{\gamma_t^n}$ and $\Psi_{\psi_t^n}$ by jointly training them. For this purpose, we optimize model parameters by minimizing a measure of discrepancy between the estimated distributions $p(F \in \mathcal{F}_{old}^n; \gamma_t^n)$ and $p(F \in \mathcal{F}_{old}^n; \psi_t^n)$ employing the training objective:

$$L_{fus}^t = \|\Pi_{old,s}^{t,n} - \Pi_{old,f}^{t,n}\|_2^2 + \lambda_{corr}\|\rho(\Pi_{old,s}^{t,n}) - \rho(\Pi_{old,f}^{t,n})\|_2^2, \tag{7.3}$$

where the subscript s and f denotes the updated prototypes of old classes estimated by the semantic and feature drift models, respectively, $\|\cdot\|_2^2$ is the squared ℓ_2 norm, $\lambda_{corr} > 0$ is the regularization parameter and $\rho(\Pi)$ is the normalized correlation matrix of Π [202]. Finally, the renovated distributions $p(F \in \mathcal{F}_{old}^n; \gamma_t^n)$ and $p(F \in \mathcal{F}_{old}^n; \psi_t^n)$ are linearly combined with equal weights to obtain $p(F \in \mathcal{F}_{old}^n)$.

Then, the overall objective used to learn representation drift is defined by:

$$L_{drift}^t = L_s^t + L_f^t + \lambda_{fus} L_{fus}^t, \tag{7.4}$$

where $\lambda_{fus} > 0$ is the loss balancing parameter. We note that L_{drift}^t measures the loss of current models on inferred representations of old classes. Thereby, we aim at reducing forgetting (ϵ_t), *i.e.*, the discrepancy between RERs as estimated by drift models and their evanescent (unavailable) counterparts by training models optimizing L_{drift}^t.

7.4.3 Optimization of Model Parameters

In the previous subsections, we designed the loss functions to capture losses induced by representation drift in CIL while training classification models. Consequently, we consider training models by minimizing $L_{cc}^t + L_{drift}^t, \forall t$. At the incremental step $t = 0$, we train f_{θ_0} and h_{ϕ_0} with $L_{ce}(\mathcal{T}_0)$. At each step $t > 0$, we train classification and drift models in an alternate fashion as follows:

- $\mathcal{F}_t^n$ is extracted from $\mathcal{T}_t$ using f_{θ_t}, and $\Gamma_{\gamma_t^n}$ and $\Psi_{\psi_t^n}$ are trained until convergence by solving:
$$\arg\min_{\gamma_t^n, \psi_t^n} L_{drift}^t(\Pi_{old}^{t,0}, \Pi_{old,\{f,s\}}^{t,n}, \mathcal{F}_t^0, \mathcal{F}_t^n; \gamma_t^n, \psi_t^n). \tag{7.5}$$
- First, $\mathcal{F}_{old}^n$ is estimated by drift models $\Gamma_{\gamma_t^n}$ and $\Psi_{\psi_t^n}$ (employed individually or fused). Then, $\Pi_{old}^{t,n}$ is computed by class-wise averaging features sampled from $p(F \in \mathcal{F}_{old}^n)$. Finally, f_{θ_t} and h_{ϕ_t} are trained on $\tilde{\mathcal{T}}_t = \mathcal{T}_t \cup \mathcal{F}_{old}^n$ by:
$$\arg\min_{\theta_t, \phi_t} L_{cc}^t(\tilde{\mathcal{T}}_t; \theta_t, \phi_t). \tag{7.6}$$

The convergence criterion for a model is early stopping the optimization of model parameters if the training loss does not change for τ steps. At the end of step $t \geq 0$, we compute $\Pi^t_{new} = \{\pi_c, c \in \mathcal{C}_t\}$ by class-wise average of feature representations $f_{\theta_t}(\mathcal{T}_t)$ of input samples and initialize $\Pi^{t+1,0}_{old} = \Pi^{t,n}_{old} \cup \Pi^t_{new}$, where $\Pi^{t,n}_{old} = \emptyset$ for $t = 0$. A detailed description of the training procedure is given in Algorithm 7.1.

7.5 Design of Representation Drift Models

In the following, we detail the design of the core architectures employed to capture feature and semantic drifts for both Gaussian and variational models.

7.5.1 Modeling Feature Drift

GM: We use a simple and lightweight multilayer perceptron to implement $\Gamma_{\gamma^n_t} : \mathcal{F}^0_t \rightarrow \mathcal{F}^n_t$, composed of two fully-connected (FC) layers and a ReLU activation between them (Table 7.2). Input and output variables of $\Gamma_{\gamma^n_t}$ correspond to sets of B feature vectors of dimension D (number of feature channels at the output of the feature extractor, which is set to 512 in all experiments), both arranged in a $B \times D$ matrix. In addition, the number of output channels of the first FC layer is set to $2 * D$ and B is set equal to the cardinality of $\mathcal{T}_t$ (*i.e.*, the available training set at step t).

VM: We use a lightweight conditional variational auto-encoder [203] to implement $\Gamma_{\gamma^n_t}$. The task of the cVAE is to learn a generative function of feature representations of training samples at stage n ($\mathcal{F}^n_t$), conditioned on the representations of the same samples at the beginning of the current incremental step ($\mathcal{F}^0_t$). The encoder and decoders are composed of two FC layers each. We refer to Table 7.3 for a more detailed description of the employed architectures. We perform conditioning in input and latent spaces by concatenation along the channel dimension. Input, output and conditioning variables of $\Gamma_{\gamma^n_t}$ correspond to sets of B feature vectors of dimension D, all arranged in $B \times D$ matrices. We set B equal to the cardinality of $\mathcal{T}_t$.

7.5.2 Modeling Semantic Drift

GM: We use a simple and lightweight multilayer perceptron to implement $\Psi_{\psi^n_t} : \mathcal{F}^n_0 \rightarrow \Pi^{t,0}_{old}$, composed of two FC layers and a ReLU activation between them (Table 7.2). Input and output variables correspond, respectively, to sets of B and C^t_{old} feature vectors of dimension D, arranged in $D \times B$ and $D \times C^t_{old}$ matrices, where C^t_{old} denotes the number of past

Table 7.2: Architecture of the Gaussian Model (GM) identified by MLP.

Feature Drift			Semantic Drift		
Input	Operator	Output	Input	Operator	Output
$B \times D$	FC layer	$B \times 2D$	$D \times B$	FC layer	$D \times 2B$
$B \times 2D$	FC layer	$B \times D$	$D \times 2B$	FC layer	$D \times C$

We set $D = 512$ and $B = |\mathcal{T}_t|$

Table 7.3: Architecture of the Variational Model (VM) identified by conditional VAE.

	Feature Drift			Semantic Drift		
	Input	Operator	Output	Input	Operator	Output
Encoder	$B \times 2D$	FC layer	$B \times 4D$	$D \times (C+B)$	FC layer	$D \times 2(C+B)$
	$B \times 4D$	FC layer	$B \times 2D$	$D \times 2(C+B)$	FC layer	$D \times 2$
Decoder	$B \times 2D$	FC layer	$B \times 4D$	$D \times (1+B)$	FC layer	$D \times 2(1+B)$
	$B \times 4D$	FC layer	$B \times 2D$	$D \times 2(1+B)$	FC layer	$D \times C$

We set $D = 512$, $B = |\mathcal{T}_t|$ and $C = |\mathcal{C}^t_{old}|$

classes present at the current incremental step $t > 0$ and D the number of feature channels at the output of the feature extractor. The number of output channels of the first FC layer is set to $2 * B$ and B is set equal to the cardinality of $\mathcal{T}_t$.

VM: We use a lightweight conditional variational auto-encoder to implement $\Psi_{\psi^n_t}$. The task of the cVAE is to learn a generative function of feature representations of old classes $\mathcal{F}^0_{old}$ (whose distribution is approximated as $p(\mathcal{F}^0_{old}; \Pi^{t,0}_{old}) \sim \mathcal{N}(\pi_c, \sigma_c)$), conditioned on those of new classes $\mathcal{F}^0_t$ (which can be extracted from the available training set $\mathcal{T}_t$). Each encoder and decoder is composed of two FC layers. Once more, we refer to Table 7.3 for further details about the employed architectures. We perform conditioning in input and latent spaces by concatenation along the channel dimension. Input and output variables of $\Gamma_{\gamma^n_t}$ correspond to sets C^t_{old} of feature vectors of dimension D, arranged in $D \times C^t_{old}$ matrices. The conditioning variable is a set of B feature vectors of dimension D, arranged in $D \times B$ matrices. We set B equal to the cardinality of $\mathcal{T}_t$.

7.6 Experimental Setup

7.6.1 Training Details

In the following, we detail the optimization objectives and hyper-parameters used in the implementation of the proposed class incremental learning framework.

Modeling Feature Drift

GM: We learn $\Gamma_{\gamma_t^n}$ by minimizing the mean squared error between $\Gamma_{\gamma_t^n}(\mathcal{F}_t^0)$ and $\mathcal{F}_t^n$, *i.e.*, $L_f^t = ||\Gamma_{\gamma_t^n}(\mathcal{F}_t^0) - \mathcal{F}_t^n||_2^2$.

VM: We optimize the variational model (conditional VAE) following the objective proposed in [204]. The learning objective is composed of a reconstruction constraint and a regularization term measuring the KL divergence between the posterior distribution modeled by the encoder and the standard normal prior, plus an additional term to maximize the mutual information between input and latent variables. Thus, the objective is of the form $L_f^t = \beta L_{rec,f}^t + (1-\alpha)L_{kl,f}^t + (\alpha - \lambda_{info} - 1)L_{info,f}^t$, where $L_{rec,f}^t$ is the reconstruction loss, $L_{kl,f}^t$ is the KL divergence loss and $L_{info,f}^t$ is the loss of the InfoVAE. We set $\beta = 1\text{e}1$ in all experiments, $\alpha = -1\text{e}1$ and $\lambda_{info} = 1\text{e}1$ on CIFAR100, and $\alpha = -1\text{e}2$ and $\lambda_{info} = 1\text{e}2$ on TinyImageNet and CUB200.

Modeling Semantic Drift

GM: We learn $\Psi_{\psi_t^n}$ by minimizing the mean squared error between $\Psi_{\psi_t^n}(\mathcal{F}_t^0)$ and $\Pi_{old}^{t,0}$, *i.e.*, $L_s^t = ||\Psi_{\psi_t^n}(\mathcal{F}_t^0) - \Pi_{old}^{t,0}||_2^2$.

VM: We optimize the variational model (conditional VAE) by maximizing the ELBO (Evidence Lower BOund) [205]. The learning objective is thus composed of a reconstruction constraint and a regularization term measuring the KL divergence between the posterior distribution modeled by the encoder and the standard normal prior, *i.e.*, $L_s^t = L_{rec,s}^t + \lambda_{kld,s}L_{kd,s}^t$. We set $\lambda_{kd,s} = 1$ in all experiments.

Model Fusion

We experimentally fine-tuned the values of λ_{fus} and λ_{corr} for each drift model configuration, dataset and incremental setup. In particular, we perform grid-search such that $\lambda_{fus}, \lambda_{corr} \in \{1\text{e}2, 1\text{e}1, 1\text{e}0, 1\text{e-}1, 1\text{e-}2, 1\text{e-}3, 1\text{e-}4, 1\text{e-}5\}$ and select the best value combination.

In all the aforementioned setups, we employ the Adam optimizer [91] with fixed learning rate η, and train until convergence by performing early-stopping, that is, the model is trained until the loss function does not change for a predefined constant number of steps $\tau = 25$. We experimentally fine-tuned the value of learning rate $\eta \in \{1\text{e-}3, 1\text{e-}4, 1\text{e-}5\}$ for each drift model configuration, dataset and incremental setup.

Finally, we apply weight normalization [206] to $\Psi_{\psi_t^n}$ (both for GM and VM implementations) and spectral normalization [207] to weights of $\Gamma_{\gamma_t^n}$ (both for GM and VM implementations), since we observed an improvement in robustness of training convergence.

7.6.2 Implementation Details

Datasets We evaluate our approach on multiple standard CIL benchmarks, that is, CIFAR100 [194], TinyImageNet [195] and CUB200-2011 [196] datasets. We devise 3 class-incremental setups; first, the framework is trained on half of the available semantic classes (except for one setup on CIFAR100, where only 40 classes are selected as the first task); then, the remaining class set is evenly divided into respectively 5, 10 or 20 incremental steps. Class order is selected randomly and then fixed at every class split.

Hyper-parameters ResNet-18 [100, 151] is used as a backbone. The model is trained for 100 epochs (*i.e.*, $N = 100$ and each stage corresponds to one epoch over $\mathcal{T}_t$) at each incremental step with Adam optimizer. Learning rate is initialised to $1\text{e}{-3}$ for CIFAR100 and TinyImageNet, and to $1\text{e}{-4}$ for CUB200-2011. It is decreased by a factor of 0.1 after 45 and 90 epochs [151]. Images are cropped to 32×32, 64×64 and 256×256 for CIFAR100, TinyImageNet and CUB200-2011 respectively, and randomly flipped. We apply input and label augmentation [151]. We set batch size to 64, and $\lambda_{fkd} = 10$ and $\lambda_{rd} = 10$ in all experiments. We employed lightweight DNNs to identify networks of Γ_γ and Ψ_ψ to model representation drifts. In particular, we investigate the use of GMs with an MLP, and we use a conditional VAE [203, 204] to implement VMs, where hyperparameters are experimentally tuned. We implement the fusion loss (7.3) using two methods for an ablation: we define ρ by (i) a normalized feature kernel matrix [202], and (ii) an identity map, *i.e.*, $A = \rho(A)$. The results obtained by (ii) are marked by † in the tables.

Comparisons We compare our approach with several CIL methods storing exemplars of old classes (EEIL [114], iCarl [112], UCIR [157], DER [182]) and other SotA exemplar-free methods (EWC [166], LwF [153], LwM [158], PASS [151], SDC [152]). As for exemplar-based methods [112, 114, 157, 158], we store 20 samples with *herd selection* [112, 157]. All methods are evaluated with the ResNet18 image classification model and batch size of 64 [151]. We evaluate the SDC [152] method by employing the prototype drift compensation proposed in [152] to update prototypes of past classes, and model old-class feature distribution by Gaussians as discussed in Section 7.3. We employ the original code of [152] to evaluate and compensate for the feature drift of old-class prototypes (*i.e.*, in place of the proposed semantic and feature representation drift models), and we use the estimated up-to-date representations $\Pi_{old}^{t,n}$ by SDC [152] to approximate feature distribution of old classes with a parametric Gaussian model, *i.e.*, $p(F \in \mathcal{F}_{old}^n; \pi_c \in \Pi_{old}^{t,n}) \sim \mathcal{N}(\pi_c, \sigma_c)$. Computation of σ_c was explained in Section 7.3.1. In Section 7.7, we will show how our methods outperform SotA exemplar-free frameworks, while surpassing some approaches that use exemplars.

7.6.3 Evaluation metrics

We evaluate the performance of different methods using the standard top-1 accuracy. Firstly, we resort to a per-step metric [151] (Tables 7.4, 7.5 and 7.6, and Fig 7.4), defined as the average top-1 classification accuracy over all classes observed up to the current incremental step k

$$\bar{a}^k = \frac{1}{|\mathcal{C}_{0:k}|} \sum_{c \in \mathcal{C}_{0:k}} a_c^k, \quad \mathcal{C}_{0:k} = \bigcup_{t=0}^{k} \mathcal{C}_t, \tag{7.7}$$

where a_c^k denotes the accuracy for class c attained at step k. Accuracy results in Table 7.4, 7.5 and 7.6 are computed at the end of the last incremental step.
We additionally make use of the step-average incremental accuracy measure proposed in [112] (Table 7.6), defined as:

$$\hat{a}^k = \frac{1}{k+1} \sum_{k'=0}^{k} \bar{a}^{k'}, \tag{7.8}$$

where we take into account the evolution of the per-step accuracy, as computed in Eq. (7.7), up to the current step k.

Finally, we report a measure of forgetting [169] computed for each past class c at step $t > 0$ by:

$$f_t^c = \max_{k<t} (a_k^c - a_t^c), \ \forall c \in \bigcup_{k<t} \mathcal{C}_k, \tag{7.9}$$

where a_k^c denotes the top-1 accuracy for class c attained at the step $k < t$. We then compute the class-wise average of forgetting measures over all past classes at each step (Figure 7.5a), as well as the task-wise average at the end of incremental training (Figure 7.5b) (*i.e.*, we compute averages of forgetting measures at the final step over classes belonging to the same task).

7.7 Experimental Results

7.7.1 Comparison with the State-of-the-Art

CIFAR100 Results given in Table 7.4 show that our models (with the best achieved accuracy) outperform the closest SotA (SDC) by 1.75%, 4.6% and 3.07% for 5, 10 and 20 steps. In Figure 7.4, results show the improved accuracy achieved by our models with respect to the competitors throughout the incremental steps.

Table 7.4: Per-class average top-1 accuracy (%) on CIFAR100 and TinyImageNet, with different incremental setups. The highest values have been highlighted in bold.

Method	CIFAR100			TinyImageNet		
	5 Steps	10 Steps	20 Steps	5 Steps	10 Steps	20 Steps
Fine-tuning	9.09	4.49	2.76	8.12	4.34	2.33
Joint	72.24	72.24	72.24	58.19	58.19	58.19
EWC [166]	26.26	19.92	3.82	14.63	6.73	3.62
LwF [153]	39.51	18.00	12.58	40.62	24.43	22.62
LwM [158]	40.49	38.39	33.65	28.39	27.18	23.55
EEIL [114]	45.26	41.36	34.84	32.03	28.93	27.25
iCarl [112]	54.06	51.11	41.20	41.81	41.39	38.68
UCIR [157]	51.13	46.00	38.31	35.73	32.95	29.23
DER [182]*	66.33	65.76	-	-	-	-
PASS [151]	56.53	47.54	47.30	47.00	41.50	29.04
SDC [152]	57.62	52.26	48.84	47.89	45.41	41.46
Feat. Drift (GM-MLP)	57.91	54.45	50.63	47.48	45.19	40.56
Sem. Drift (GM-MLP)	58.33	54.15	50.85	47.92	46.21	42.43
Fusion† (GM-MLP)	58.89	55.95	51.61	47.95	46.36	42.43
Fusion (GM-MLP)	**59.37**	55.99	**51.91**	48.56	46.50	42.81
Feat. Drift (VM-VAE)	56.99	53.69	51.09	47.88	44.67	41.05
Sem. Drift (VM-VAE)	58.17	55.38	51.65	48.60	46.24	43.44
Fusion† (VM-VAE)	58.76	55.50	51.72	**48.74**	46.46	42.72
Fusion (VM-VAE)	58.72	**56.86**	51.75	48.57	**46.92**	**44.61**

† Without using the correlation objective [202] in fusion loss.
* Numerical values were directly taken from [182].

TinyImageNet Table 7.4 shows that our framework outperforms exemplar-based competitors and the SotA methods not using exemplars [151, 152]. In particular, our drift models yield superior performance with respect to SDC [152]. This is especially true when semantic and feature representation drifts are jointly taken into account, showing that they both individually model crucial and complementary information by model fusion, which is not fully captured by SDC [152].

CUB200-2011 Table 7.5 shows that non-exemplar methods provide quite low results, especially when the number of incremental steps is increased. Adopting the method proposed in [152] to compensate for modeling shift of prototypes using a softmax classifier seems to have no beneficial effect, showing that it fails to adequately model semantic drift in a fine-grained classification setup with high semantic similarity among classes. On the other end, our framework demonstrates to successfully capture and model representation drift; by injecting up-to-date knowledge of old classes, in fact, we manage to more effectively mitigate catastrophic forgetting. Per-step accuracy values displayed in Figure 7.4 corroborate accuracy results given in Table 7.5.

Table 7.5: Per-class average top-1 accuracy (%) on CUB200, with different incremental setups. The highest values have been highlighted in bold.

Method	CUB200 5 Steps	CUB200 10 Steps	CUB200 20 Steps
Fine-tuning	10.93	7.10	4.63
Joint	74.67	74.67	74.67
EWC [166]	10.63	6.43	4.65
LwF [153]	26.40	13.65	7.89
PASS [151]	52.14	37.97	18.29
SDC [152]	52.30	38.30	18.17
Feat. Drift (GM-MLP)	55.87	50.67	31.36
Sem. Drift (GM-MLP)	56.51	47.89	32.50
Fusion† (GM-MLP)	56.20	52.07	36.67
Fusion (GM-MLP)	56.28	51.82	37.99
Feat. Drift (VM-VAE)	57.39	51.29	32.72
Sem. Drift (VM-VAE)	**57.34**	51.88	33.34
Fusion† (VM-VAE)	56.59	52.00	36.80
Fusion (VM-VAE)	56.97	**52.58**	**38.26**

† Without using the correlation objective [202] in fusion loss.

Table 7.6: *acc1*:*acc2* top-1 accuracy (%) where *acc1* is the class-wise average accuracy (Eq. (7.7)) and *acc2* is the class- and step- wise average accuracy (Eq. (7.8)). Both measures are computed at the end of the last incremental step.

Method	CIFAR100 ResNet18 [151] 5 Steps	CIFAR100 ResNet18 [151] 10 Steps	CIFAR100 ResNet18 [151] 20 Steps	CIFAR100 ResNet32 [112] 5 Steps	CIFAR100 ResNet32 [112] 10 Steps	Imagenet-Subset ResNet18 [151] 10 Steps	Imagenet-Subset ResNet18 [151] 20 Steps
iCarl [112]	54.1:64.8	51.1:62.3	41.2:56.3	-:-	-:-	54.7:68.6	48.7:63.9
UCIR [157]	51.1:61.4	46.0:57.5	38.3:51.3	-:63.4*	-:60.2*	57.5:66.6	45.5:57.3
PASS [151]	56.5:65.1	47.6:60.8	47.3:58.7	51.1:58.9	44.4:53.2	58.1:68.2	47.2:61.4
SDC [152]	57.6:66.2	52.3:62.7	48.8:59.2	51.4:59.4	47.0:54.4	58.6:68.6	47.1:61.0
DER [182]*	-:73.2	-:72.8	-:-	-:68.5	-:67.1	74.9:78.2	-:-
Feat. Drift (GM-MLP)	57.9:66.3	54.5:63.7	50.6:61.2	53.1:60.2	50.9:58.3	59.1:68.1	50.6:62.6
Sem. Drift (GM-MLP)	58.3:66.0	54.2:63.4	50.9:60.8	53.3:60.3	51.2:58.4	59.3:69.2	50.4:63.4
Fusion (GM-MLP)	59.4:66.9	56.0:64.8	51.9:61.5	53.9:60.9	52.0:58.5	59.7:69.3	51.5:63.6
Feat. Drift (VM-VAE)	57.0:65.8	53.7:63.0	51.1:60.9	53.8:60.2	50.9:58.3	59.3:68.8	50.9:62.8
Sem. Drift (VM-VAE)	58.2:66.7	55.4:63.3	51.7:61.6	54.1:60.9	51.3:58.9	59.5:69.3	51.0:63.1
Fusion (VM-VAE)	58.7:66.8	56.9:65.1	51.8:61.6	54.2:61.3	52.1:59.1	60.2:70.0	51.6:63.7

* Numerical values were directly taken from [182].

7.7.2 Additional Results

To validate robustness of our approach w.r.t. evaluation metric, in Table 7.6 we report additional results in the form of the step-average incremental accuracy from [112] (Eq. (7.8)). We notice that we still outperform most of the competitors. This holds also for the large-scale ImageNet-Subset benchmark, where our approach attains better performance than SotA

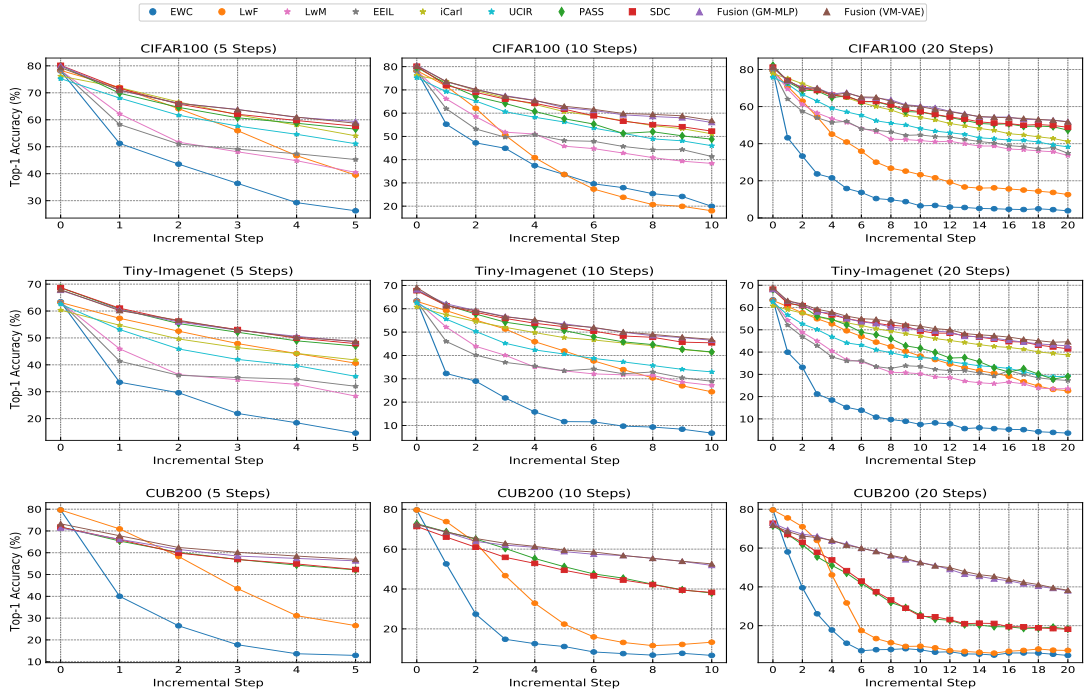


Figure 7.4: Per step average top-1 accuracy (%) on the CIFAR100, CUB200-2011 and TinyImageNet datasets.

exemplar-free methods and some exemplar-based frameworks, demonstrating resilience to change of data settings. In Table 7.6 we further report evaluation results on Cifar100 when a modified 32-layers ResNet [112] is employed as classification network. Once more, we observe that we surpass PASS [151] and SDC [152] exemplar-free SotA competitors by a large margin, especially when 10 incremental steps are performed. This indicates robustness to change of classification backbone.

Furthermore, we evaluate the CIL methods considered by computing a measure of forgetting (Eq. (7.9)). In Figure 7.5a, we analyze how the class-wise average forgetting evolves throughout incremental training when different methods are employed. We observe that the proposed CIL approach based on representation drift modeling mitigates forgetting more efficiently than the majority of the competitors (only iCARL [112] shows superior performance). Nonetheless, we remark that iCARL [112] leverages replay data from the past to address forgetting. Furthermore, iCARL [112] yields a lower or comparable classification accuracy with respect to our approach, suggesting that a focus of [112] on preserving past knowledge (*i.e.*, *stability*) is accompanied by a less efficient learning of novel classes (*i.e.*, *plasticity*). Finally, we propose an analysis using task forgetting computed at the end of incremental training (Figure 7.5b). We notice how our CIL method causes overall less forgetting

than most of the competing approaches, providing an improved performance equally shared among all tasks. This is especially noticeable when evaluation is done on CUB200, where even the SotA PASS and SDC methods induce almost total forgetting of the oldest tasks.

7.7.3 Ablation Study

Modeling drifts with GMs and VMs

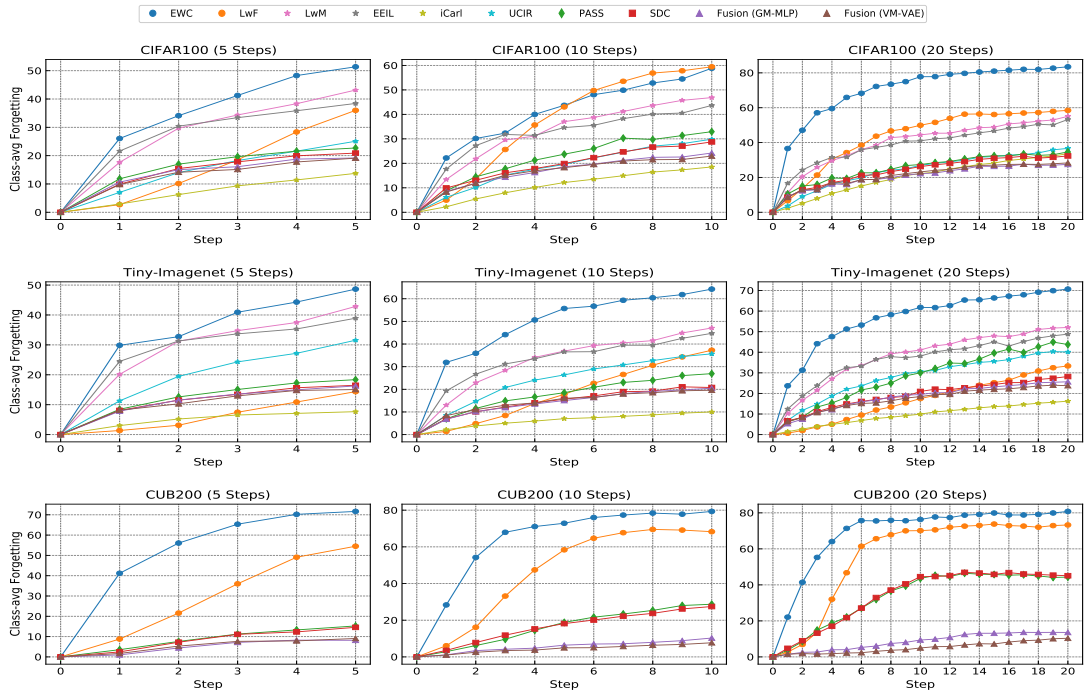
Our framework enables implementation of drift models using different GMs and VMs. We studied the accuracy of a GM (MLP) and VM (VAE) for modeling different drifts and their fusion in Section 7.7.1. Results suggest that the accuracy of GMs and VMs depends on statistical sufficiency of data which affects *capacity* of f_θ and learned representations as follows:

- On the Cifar100 dataset, the GM (MLP) outperforms the VM (VAE) for smaller (*e.g.*, 5) steps, where more classes are observed at each step, compared to the larger (*e.g.*, 20) steps. We conjecture that this result can be attributed to training models using statistically insufficient data representing all classes at each step.
- On TinyImageNet, containing larger images than Cifar100, the VM (VAE) performs on par with and slightly outperforms the GM (MLP) for smaller steps.
- On CUB200, which comprises the largest images, the variational model (VAE) outperforms the Gaussian model (MLP) for all steps.

Analysing semantic drift

We study how the proposed framework captures and preserves semantic relationships between representation of old and new classes by modeling semantic drift. For this purpose, we compute prototypical representations of novel classes on the training data available at an incremental step t, and estimate the *revived* prototypes of old categories by modeling semantic and feature drifts (employed individually or fused). Then, we express inter-class relationships in the form of Euclidean distance between prototypes of past and new classes, and observe the evolution of such distance throughout an incremental step t. In Figure 7.6, we report distance values computed at the beginning and end of an incremental step, together with their difference (distances are normalized along new-class axis). Results are presented for the analyses carried out when performing 20 incremental training steps on the CIFAR100, TinyImageNet and CUB200 datasets. We focus on analyzing the change of semantic relationship during the first incremental step (*i.e.*, $t = 1$).

(a) Per step average class forgetting (%) on the CIFAR100, CUB200-2011 and TinyImageNet datasets.



(b) Per task average forgetting (%) at the end of training on the CIFAR100, CUB200-2011 and TinyImageNet datasets.

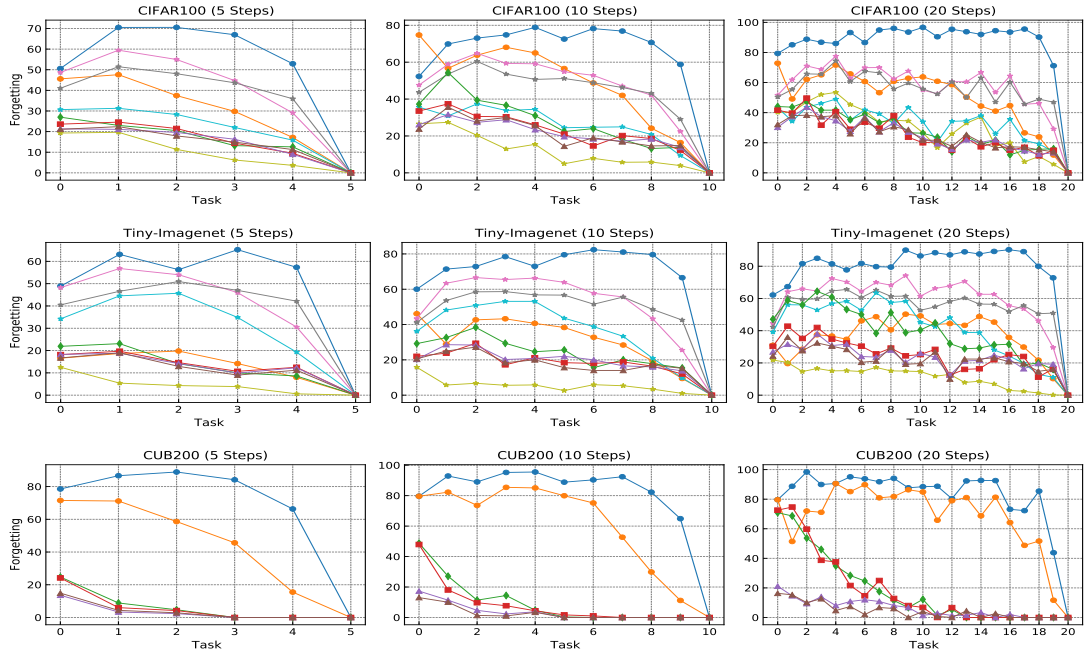
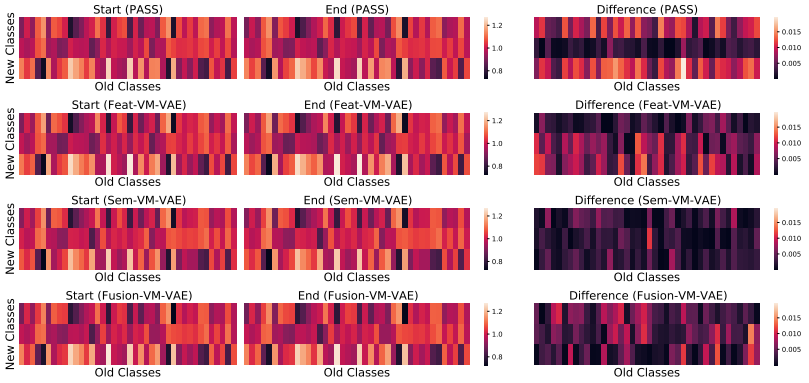
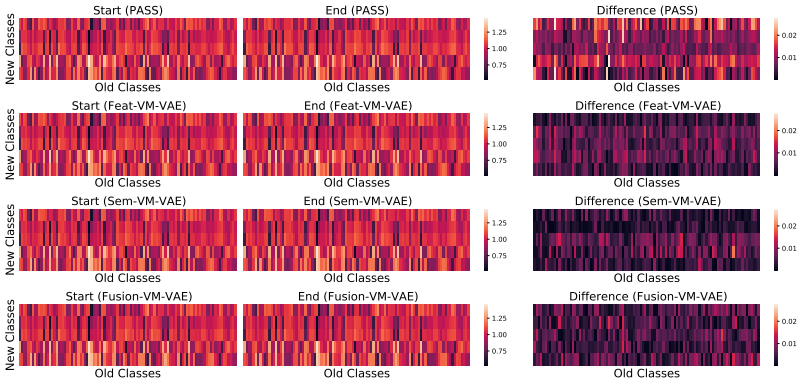


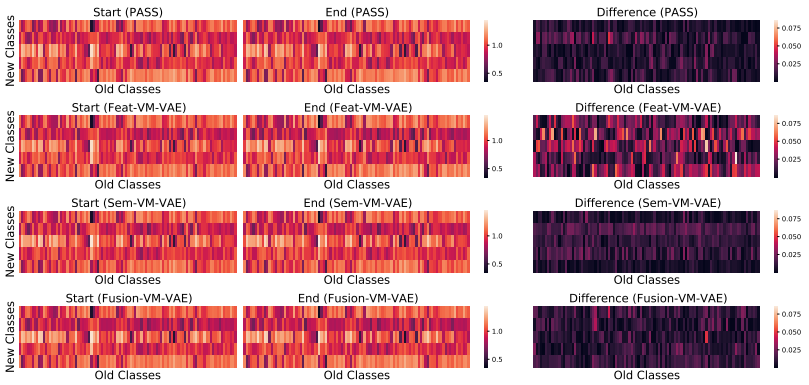
Figure 7.5: Per step and per-task average forgetting (%) curves.



(a) CIFAR100



(b) TinyImageNet



(c) CUB200

Figure 7.6: Analysis of normalised distance between estimated prototypes of classes seen at steps $t = 0$ and $t = 1$, captured at the beginning (left) and end (mid) of step $t = 1$. We report the absolute value of the difference of the two measures (right). We replicate the analysis for the 20-step incremental setup, over the CIFAR100 (top), TinyImageNet (middle) and CUB200 (bottom) datasets.

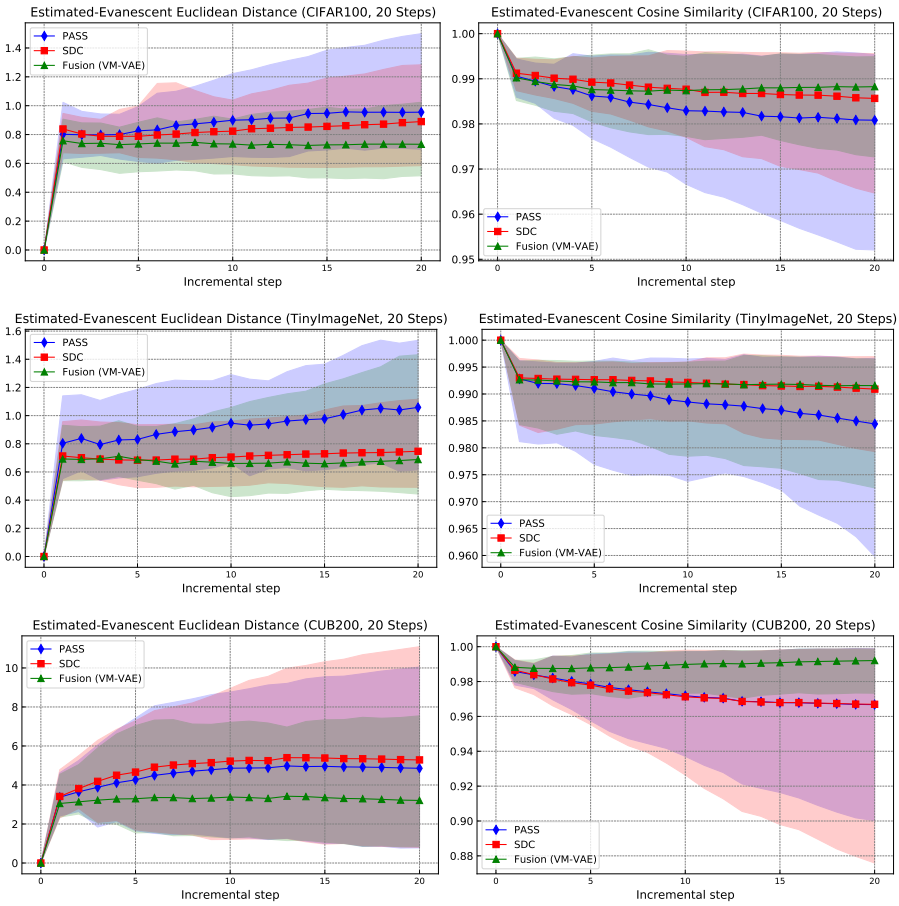


Figure 7.7: Analysis of average distance between estimated (*revived*) and evanescent prototypes of old classes.

We notice how prototypes estimated by leveraging the modeled semantic representation drift tend to more effectively preserve inter-class relationships with respect to novel classes. By utilizing feature drift alone, in fact, we notice that class representations tend to modify their interconnections. Nonetheless, the proposed model fusion allows to better identify and retain inter-class relationships, whereas keeping prototypes fixed (as in PASS) causes a greater impairment of inter-class relationships as new representations are learned.

Analysing feature drift of evanescent representations

We analyze the efficacy of the proposed model in estimating the feature drift undergone by evanescent representations of old classes when novel classes are learned. To this end, we compute the Euclidean and cosine distances between estimated (*revived*) prototypes of old classes and their reference (*i.e.*, evanescent) representations at each incremental step (Figure 7.7).

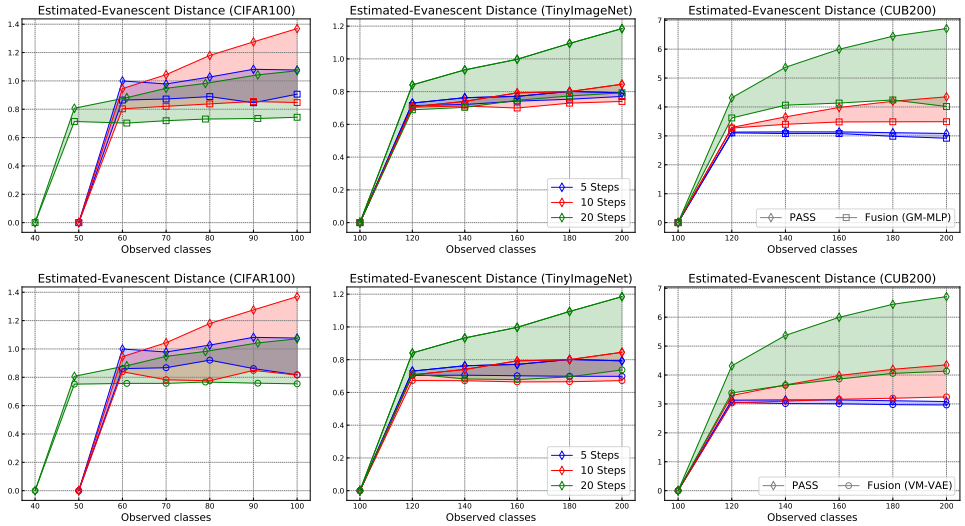


Figure 7.8: Average Euclidean distance between the estimated (*revived*) and evanescent prototypes of old classes.

We provide per-step class-wise average distance values (main curves), as well as the maximum and minimum class values that identify, respectively, the upper and lower bound of the shaded regions for each setup. Feature prototypes of old classes are estimated by computing class wise averages of feature representations over training data when they are available at an incremental step, which can then be simply fixed for the rest of the training (as done in PASS [151]), or can be updated by SDC [152] or by the proposed drift models. Evanescent prototypical representations of the same old classes are instead computed over the test set (unavailable during training). We replicate the analysis on the CIFAR100, TinyImageNet and CUB200 datasets to provide a robust evaluation. The results show that our proposed methods can track the trajectory of evanescent prototypes more efficiently (in terms of geometric distances) compared to the SotA PASS and SDC methods, by modeling the evolution of the representations (*i.e.*, feature drift). Furthermore, we observe that the improved accuracy with respect to the SotA is shared among the three datasets chosen for evaluation. In particular, we remark the noticeable improvement experienced on the CUB200 dataset. Our method, in fact, provides much lower Euclidean distance between revived and evanescent representations, whose average value is kept almost constant as incremental training progresses and new classes are introduced. The same trend can be observed for the cosine similarity of the estimated and evanescent representations, whose value tends to be steady over the incremental training and much closer to the upper bound when our method is adopted.

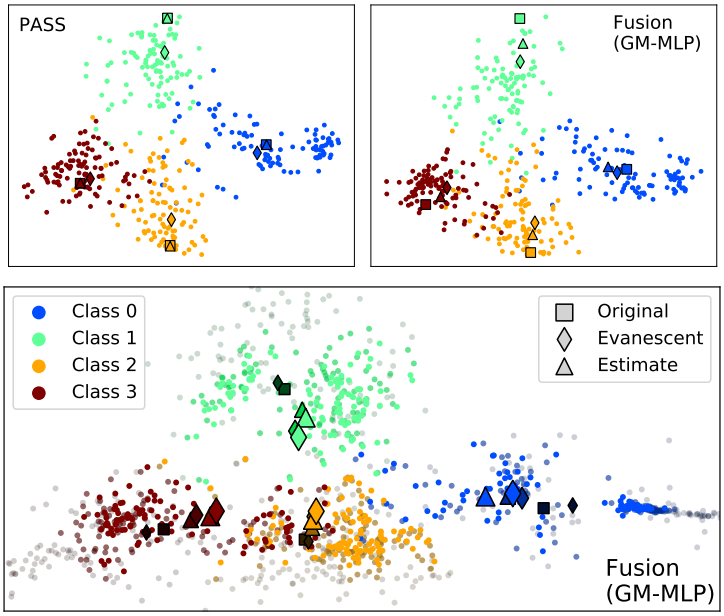
In Figure 7.8, we compare the Euclidean distance between the estimated and evanescent prototypes for different number of total incremental steps, on the CIFAR100, TinyImageNet and CUB200 datasets, when employing model fusion with both GM and VM. We

observe that our method always outperforms PASS, in all the evaluation setups. In addition, we notice that by employing VM to identify representation drifts, we reach the largest improvement over PASS, which translates into the revived evanescent representations closer to their unknown reference version. We also highlight once more the noticeable accuracy provided by our model for the 20-step setup on the CUB200, where our approach jointly shows the largest improvement over the SotA accuracy by $\sim 20\%$.

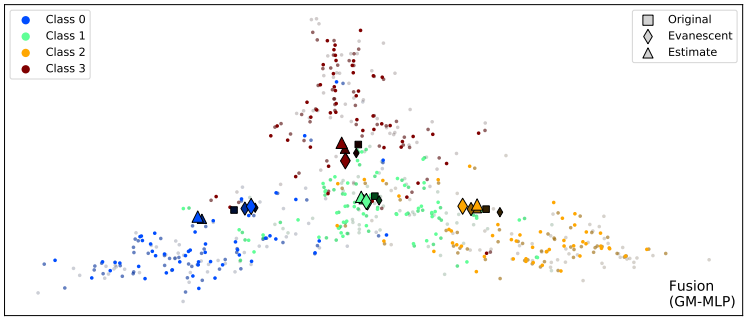
Finally, we visualize 2D embeddings of feature vectors of the first four observed classes as computed by the feature extractor on the same set of input samples at different incremental steps. To project high-dimensional feature vectors to a 2D space, we use Isomap [208]. Results are reported in Figure 7.9 for the CIFAR100, TinyImageNet and CUB200 datasets. We observe that our proposed method allows to estimate revived prototypical representations that tend to be projected closer to their evanescent versions compared to PASS (especially on the CIFAR100 and CUB200 benchmarks). This shows that we can approximate the trajectory of evanescent representations of old classes, without having access to training data of such categories, by modeling representation drift.

Analysing how learned evanescent representations affect classification accuracy

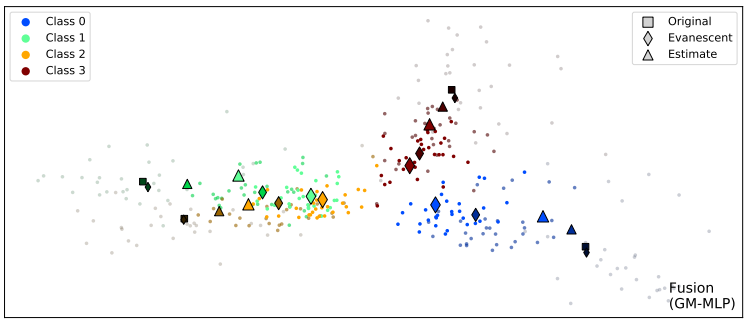
In this section, we investigate the relationship tying the accuracy of the classification model and the *normalized* distance between the revived and evanescent prototypes of old classes. Normalization is performed by dividing individual distances computed for single past classes by the average distance among all the past classes. We then provide the average of normalized distance values at each incremental step. We evaluate the accuracy of the proposed method when fusing semantic and feature drift models and adopting GM to identify representation drift, alongside with that of the SDC and PASS. Results are reported in Figure 7.10. We observe that classification accuracy measured at each incremental step and distance between the estimated and evanescent prototypes are negatively correlated, with similar trends shared by the different methods being analyzed, across the CIFAR100, TinyImageNet and CUB200 benchmarks. It is worth noting that our method and SDC display very similar correlation patterns, whilst the latter reaching lower accuracy and higher distance values at the final incremental step. Therefore, state-of-the-art PASS and SDC methods seem to overfit to training data as the models are trained incrementally. For instance, on CIFAR100 the accuracy of PASS and SDC continues to decrease below 50% and their produced prototypical representations diverge from the reference prototype as the incremental steps increase. However, the proposed methods limit the distance measure between prototypical representations by 0.3 and the accuracy by 50%. Thereby, we argue that our method yields superior performance that the SotA PASS and SDC by more accurately tracking evanescent old-class prototypes.



(a) CIFAR100



(b) TinyImageNet



(c) CUB200

Figure 7.9: Feature representations of the first four learned classes (CIFAR100, 20 steps) extracted from samples of test set (dots), along with their prototypes computed over available training data (squares), over test data (diamonds) and estimated prototypes (diamonds). In the lower plot, decrease in transparency and increase in brightness indicate that representations are extracted at progressively increasing incremental steps (*i.e.*, at steps 0, 10 and 20).

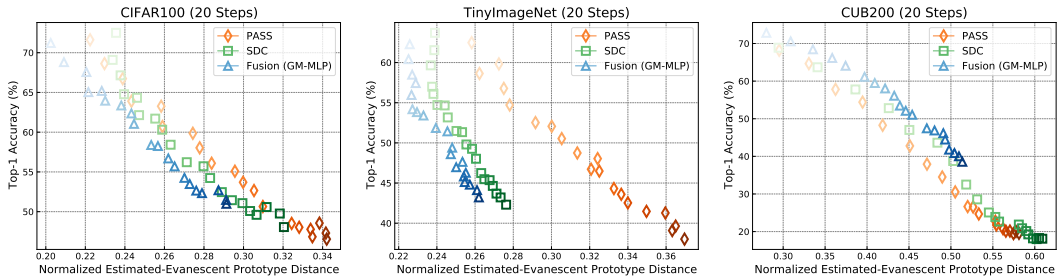


Figure 7.10: Relationship between top-1 accuracy (%) and *normalized* Euclidean distance between estimated and evanescent old-class prototypes. Each point depicts a single training phase, and the decrease in transparency indicates progressively increasing incremental steps. For each step, accuracy values have been averaged over all classes observed so far, and distances are averaged over all past classes.

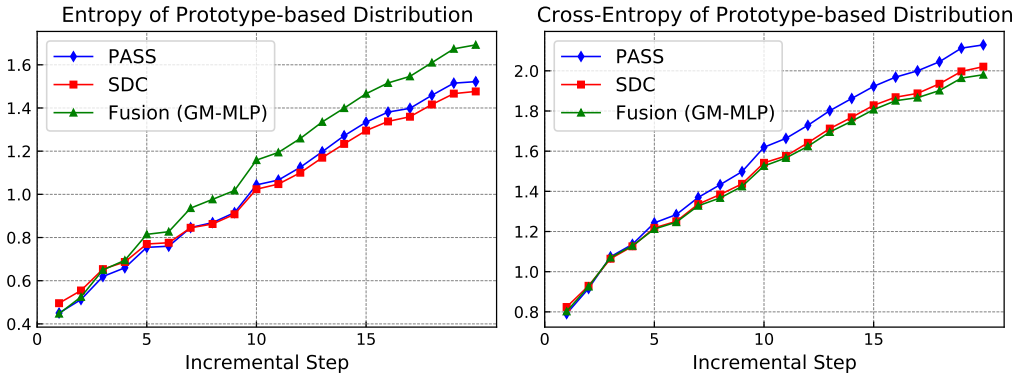


Figure 7.11: Average entropy (H) and cross-entropy (CE) of p_F (CIFAR100, 20 steps).

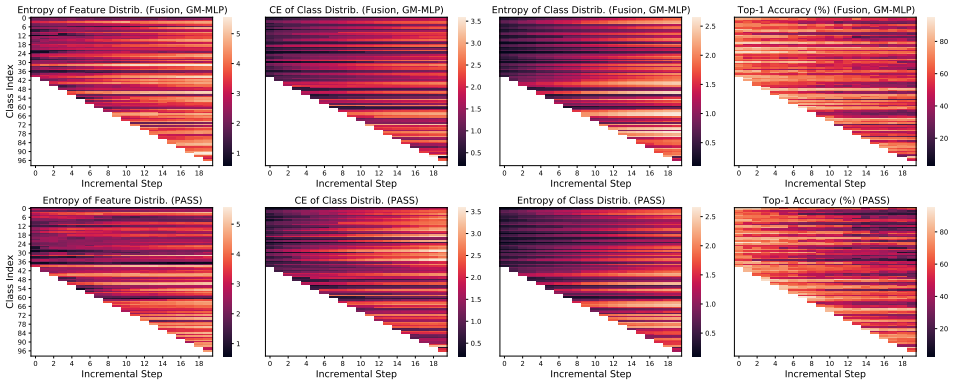
Statistical analyses of representations

We explore how well the estimated revived prototypes of old classes exemplify feature representations of samples of such categories (which are unavailable during training at incremental steps). To this end, we first compute probability distributions over the set of old classes based on the Euclidean distance between feature representation and class prototypes by:

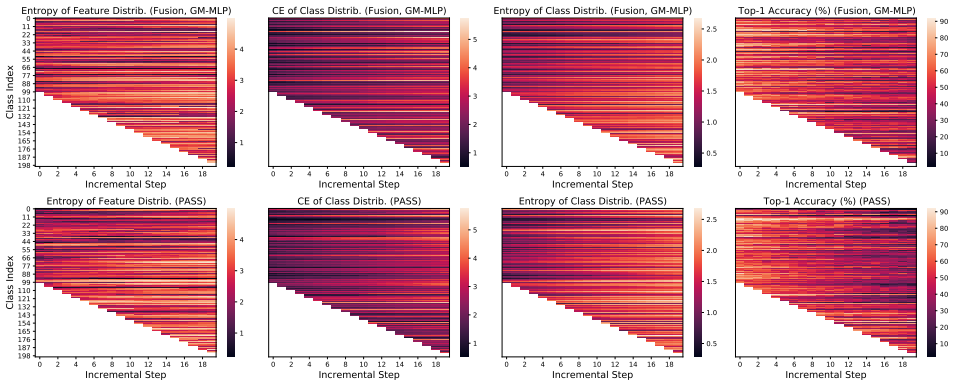
$$p_F(c) = \frac{\exp(-||F - \pi_c)||_2/\zeta)}{\sum_j \exp(-||F - \pi_j)||_2/\zeta)}, \tag{7.10}$$

where $F \in \mathcal{F}_{old}$ is the feature representation of a test sample of $\mathcal{C}_{old}$ as produced by the current feature extractor, $\{\pi_j\}_j$ are revived prototypes of $\mathcal{C}_{old}$ and ζ is set to 0.1.

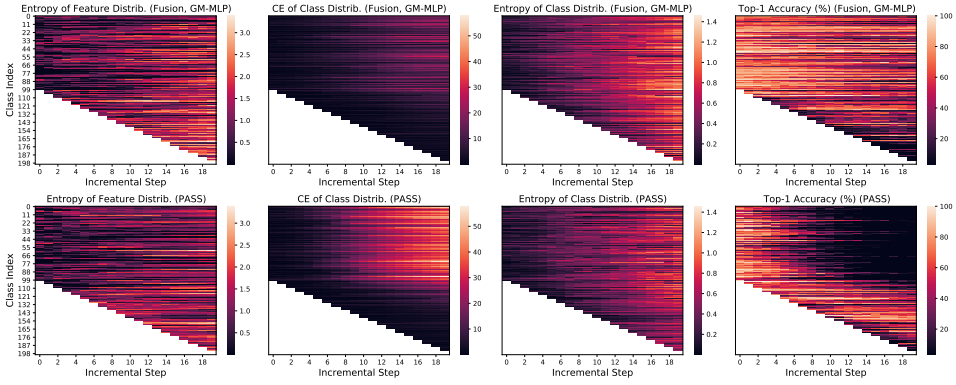
We analyze the change of entropy (H) and cross-entropy (CE) of p_F across incremental steps in in Figure 7.11 and Figure 7.12. The cross-entropy is computed w.r.t. to the one-hotted ground-truth distribution of the labeled input sample corresponding to $F \in \mathcal{F}_{old}$.



(a) CIFAR100



(b) TinyImageNet



(c) CUB200

Figure 7.12: Left: average entropy of $p_c(F_i)$ (Eq. (7.11)). Mid-left and mid-right: entropy and cross-entropy with respect to ground-truth of $p_F(c)$ (Eq. (7.10)). Each feature F_j is extracted from a test image belonging to an old class, and π_j are the estimated prototypes of old classes. Right: Top-1 accuracy for the CIFAR100 (top), TinyImagenet (middle) and CUB200 (bottom) (all models were evaluated for 20 incremental steps).

In addition, for each old class $c \in \mathcal{C}_{old}$, we compute the mean H and CE of p_F, *i.e.*, we average over all F corresponding to the same c.
We observe that our method provides higher H and smaller CE compared to PASS, and this trend is shared across CIFAR100, TinyImageNet and CUB200. This result suggests that information capacity of representations learned by our methods increases along with classification accuracy more rapidly compared to the SotA as models are incrementally trained.

To further validate this claim, for each old class c, we compute the probability distribution over feature representations of input samples of *all* past categories, defined by:

$$p_c(F_i) = \frac{\exp(-||F_i - \pi_c)||_2/\tau)}{\sum_j \exp(-||F_j - \pi_c)||_2/\tau)}, \tag{7.11}$$

where $F_j \in \mathcal{F}_{old}$ are feature representations of test samples and $\{\pi_j\}_j$ are revived old-class prototypes. In addition, τ is set to 0.1. In Figure 7.12, we report entropy values of p_c across different incremental steps. Once more, we observe that our method causes entropy of p_c to reach higher values compared to PASS, indicating that prototypical representations revived by our method are more informative than their fixed counterparts, while still being representative of the corresponding class, as suggested by the lower CE of p_F.

8

Continual Learning by Image-level Replay

8.1 Introduction

As discussed in Chapter 6, class incremental learning on dense tasks (*e.g.*, semantic segmentation), where pixel-wise predictions are performed, has only recently been explored and the first experimental studies show that catastrophic forgetting is even more severe than on the classification task [118, 119]. Current approaches for class-incremental semantic segmentation re-frame knowledge distillation strategies inspired by previous works on image classification [3, 4, 118, 119]. Although they partially alleviate forgetting, they often fail when multiple incremental steps are performed or when background shift [3] (*i.e.*, change of statistics of the background across learning steps, as it incorporates old or future classes) occurs.

In this section, we follow a completely different strategy and, instead of distilling knowledge from a teacher model (*i.e.*, the old one) to avoid forgetting, we propose to generate samples of old classes by using replay strategies. We propose RECALL (**RE**play in **C**ontinu**AL** **L**earning) [241], a method that re-creates representations of old classes and mixes them with the available training data, *i.e.*, containing novel classes being learned (see Figure 8.1). To reduce background shift, we introduce a self-inpainting strategy that re-assigns the background region according to predictions of the previous model. Similarly to the CIL approach developed in Chapter 7, data representations from former tasks are replayed to mitigate forgetting. Yet, here we focus on the more challenging semantic segmentation problem to be tackled in an incremental fashion, and we aim at retrieving old-class training samples directly within the input space, alongside their dense labeling.

To generate representations of past classes we pursue two possible directions. The first is

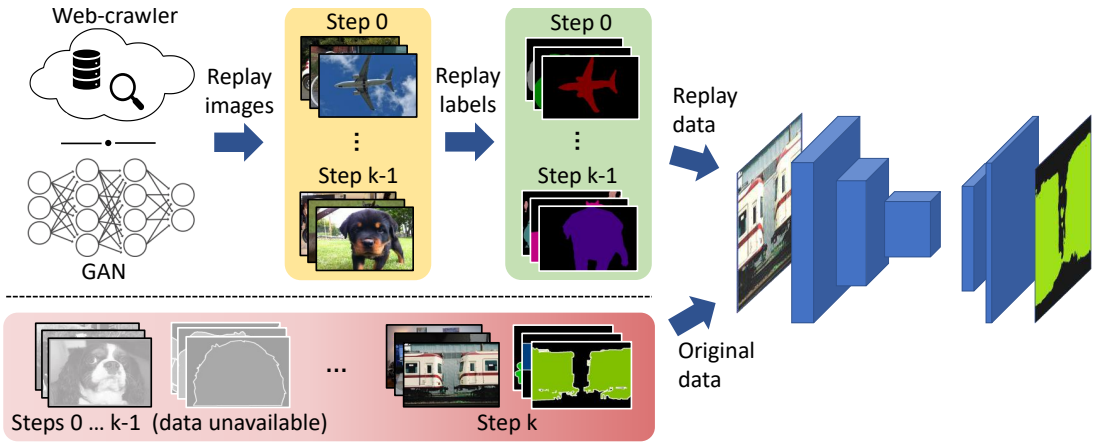


Figure 8.1: Replay images of previously seen classes are retrieved by a web crawler or a generative network and further labeled. Then, the network is incrementally trained with a mixture of new and replay data.

based on a pre-trained generative model, *i.e.*, a Generative Adversarial Network (GAN) [31] conditioned to produce samples of an input class. The GAN has been trained beforehand on a dataset different than the target one (we chose ImageNet as it comprehends a wide variety of classes and domains), thus requiring a Class Mapping Module to perform the translation between the two label spaces. The second strategy, instead, is based on crawling images from the web, querying the class names to drive the search. Both approaches allow to retrieve a large amount of weakly labeled data. Finally, we generate pseudo-labels for semantic segmentation using a side labeling module, which requires only minimal extra storage.

Our main contributions in this chapter are:

(i) We propose RECALL, the first approach to use replay data from external sources (*i.e.*, still following an exemplar-free setup) in continual semantic segmentation.

(ii) To the best of our knowledge, we are the first to propose a webly-supervised paradigm in continual learning, showing how it is possible to effectively extract useful clues from extremely weakly supervised and noisy samples.

(iii) We devise a background inpainting mechanism to enhance ground-truth information by pseudo-labeling image regions associated to old classes, and overcome the semantic shift undergone by the background.

(iv) We achieve state-of-the-art results on a wide set of incremental learning benchmarks, strongly outperforming competitors when the training process comprises a large number of incremental steps.

The reminder of the chapter develops as follows: first the CIL problem will be formalized for semantic segmentation, detailing the experimental setups in terms of ground truth availability that will be considered (Section 8.2); then, our RECALL framework will be introduced in all its components, from the source and labeling evaluation blocks for replay data retrieval to the label inpainting mechanism for ground truth enhancement, along with the training procedure to optimize each component (Section 8.3); we will further delve into the details of the image retrieval strategy, describing the generative and web-crawling solutions we devised (Section 8.4; finally, the chapter will conclude with the experimental analysis and ablation studies to prove the efficacy of the proposed CIL method (Sections 8.5 and 8.6).

8.2 Problem Formulation

The semantic segmentation task consists in labeling each pixel in an image by assigning it to a class from a collection of possible semantic classes $\mathcal{C}$, which typically also comprises a special background category that we denote as b. More formally, given an image $\mathbf{X} \in \mathcal{X} \subset \mathbb{R}^{H \times W \times 3}$, we aim at producing a map $\hat{\mathbf{Y}} \in \mathcal{Y} \subset \mathcal{C}^{H \times W}$ that is a prediction of the ground-truth map $\mathbf{Y}$. This is nowadays usually achieved by using a suitable deep learning model $M : \mathcal{X} \mapsto \mathbb{R}^{H \times W \times |\mathcal{C}|}$, commonly made by a feature extractor M^e followed by a decoding module M^d, *i.e.*, $M = M^d \circ M^e$.

In standard supervised learning, the model is learned in a single shot over a training set $\mathcal{T} \subset \mathcal{X} \times \mathcal{Y}$, available in its complete form to the training algorithm. In class-incremental learning, instead, we assume that the training is performed in multiple steps and only a subset of training data is made available to the algorithm at each step $t = 0, ..., T$. More in detail, we start from an initial step $t = 0$ where only training data concerning a subset of all the classes $\mathcal{C}_0 \subset \mathcal{C}$ is available (we assume that $b \in \mathcal{C}_0$). We denote with $M_0 : \mathcal{X} \mapsto \mathbb{R}^{H \times W \times |\mathcal{C}_0|}$, $M_0 = M_0^d \circ M_0^e$ the model trained after this initial step. Moving to a generic step t, a new set of classes $\mathcal{C}_t$ is added to the class collection $\mathcal{C}_{0:t-1}$ learned up to that point, resulting in an expanded set of learnable classes $\mathcal{C}_{0:t} = \mathcal{C}_{0:t-1} \cup \mathcal{C}_t$ (we assume $\mathcal{C}_{0:t-1} \cap \mathcal{C}_t = \varnothing$). The model after the t-th step of training is $M_t : \mathcal{X} \mapsto \mathbb{R}^{H \times W \times |\mathcal{C}_{0:t}|}$, where $M_t = M_t^d \circ M_0^e$, since in our approach the encoder M_0^e is not trained during the incremental steps and only the decoder is updated [118].

Two main continual scenarios have been proposed (see Section 6.4.1 for a more detailed description) and we tackle both in a unified framework.

Disjoint setup: in the initial step, all the images in the training set with at least one pixel belonging to a class of $\mathcal{C}_0$ (except for b) are assumed to be available. We denote with $\mathcal{Y}_{\mathcal{C}_0 \cup \{b\}} \subset \mathcal{C}_0^{H \times W}$ the corresponding output space where labels can only belong to $\mathcal{C}_0$, while all the pix-

Table 8.1: Formal definition of key notation used throughout Chapter 8.

Symbol	Definition
$t \in \{0, 1, 2, ..., T\}$	Step index
b	Background class
S	Source Block
M_0^e	Encoder trained at step $t = 0$ and kept frozen
$M_{\mathcal{C}_t}^{d,H}$	Helper decoder of step t
$L_{\mathcal{C}_t} = M_{\mathcal{C}_t}^{d,H} \circ M_0^e$	Instance of Label Evaluation Block of step t
$\{L_{\mathcal{C}_t}\}_{\mathcal{C}_t \subset \mathcal{C}}$	Label Evaluation Block
$\mathcal{X}_{\mathcal{C}_t}^{rp}$	Replay image data of step t
$\mathcal{Y}_{\mathcal{C}_t}^{rp}$	Pseudo-labels of replay image data of step t
$\mathcal{R}_{\mathcal{C}_t} \subset \mathcal{X}_{\mathcal{C}_t}^{rp} \times \mathcal{Y}_{\mathcal{C}_t}^{rp}$	Replay training data of step t
$\mathcal{T}_t^{bi} \subset \mathcal{X} \times \mathcal{Y}_{\mathcal{C}_{0:t}}$	Original training data of step $t > 0$ after background inpainting
$\mathcal{T}_t^{rp} = \mathcal{T}_t^{bi} \cup \mathcal{R}_{\mathcal{C}_{0:t-1}}$	Enhanced training data at step t by replay and background inpainting
G	Data generation network (*e.g.*, GAN)
$\mathcal{C}^G$	GAN's class set
I	Classifier on GAN's pre-training dataset

els not pertaining to these classes are assigned to b. Incremental partitions are built as *disjoint* subsets of the whole training set. The training data associated to the t-th step, $\mathcal{T}_t \subset \mathcal{X} \times \mathcal{Y}_{\mathcal{C}_t \cup \{b\}}$, has only images corresponding to classes in $\mathcal{C}_t$ with just classes of step t annotated (old classes are labeled as b), and is disjoint with respect to previous and past partitions. **Overlapped setup**: in the first phase we select the subset of training images having only $\mathcal{C}_0$-labeled pixels. Then, the training set at each incremental step contains *all* the images with labeled pixels from $\mathcal{C}_t$, *i.e.*, $\mathcal{T}_t \subset \mathcal{X} \times \mathcal{Y}_{\mathcal{C}_t \cup \{b\}}$. Similarly to the initial step, labels are limited to semantic classes in $\mathcal{C}_t$, while remaining pixels are assigned to b.

In both setups, b undergoes a semantic shift at each step, as pixels of ever changing class sets are assigned to it.

8.3 RECALL: Replay in Continual Learning

In the standard setup, the segmentation model M is trained with annotated samples from a training set $\mathcal{T}$. Data should be representative of the task we would like to solve, meaning that multiple instances of all the considered semantic classes $\mathcal{C}$ should be available in the provided dataset for the segmentation network to properly learn them. Once $\mathcal{T}$ has been assembled,

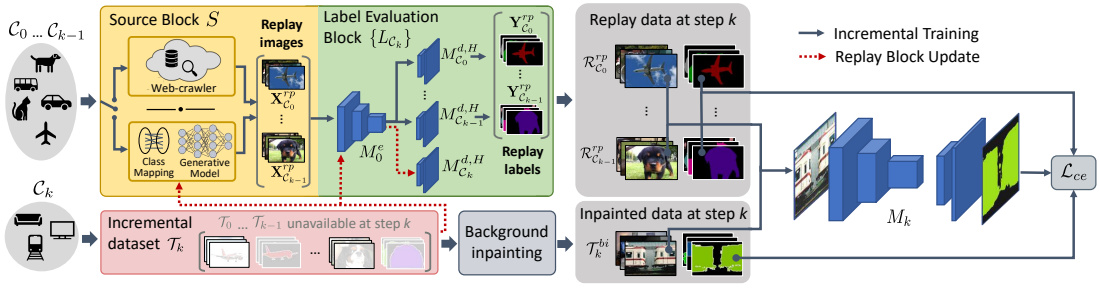


Figure 8.2: Overview of the proposed RECALL: class labels from past incremental steps are provided to a Source Block, either a web crawler or a pre-trained conditional GAN, which retrieves a set of unlabeled replay images for the past semantic classes. Then, a Label Evaluation Block produces the missing annotations. Finally, the segmentation network is incrementally trained with a replay-augmented dataset, composed of both new classes data and replay data.

the cross-entropy objective is commonly employed to optimize the weights of M:

$$\mathcal{L}_{ce}(M;\mathcal{C},\mathcal{T}) = -\frac{1}{|\mathcal{T}|}\sum_{\mathbf{X},\mathbf{Y}\in\mathcal{T}}\sum_{c\in\mathcal{C}}\mathbf{Y}[c]\cdot\log\big(M(\mathbf{X})[c]\big). \tag{8.1}$$

In the incremental learning setting, when performing an incremental training step t only samples related to new classes $\mathcal{C}_t$ are assumed to be at our disposal. Following the simplest approach, we could initialize our model's weights from the previous step ($M_{t\text{-}1}$, $t \geq 1$) and learn the segmentation task over classes $\mathcal{C}_{0:t}$ by optimizing the standard $\mathcal{L}_{ce}(M_t;\mathcal{C}_{0:t},\mathcal{T}_t)$ objective with data from the current training partition $\mathcal{T}_t$. However, simple fine-tuning leads to catastrophic forgetting, being unable to preserve previous knowledge. In the following we will detail the proposed framework to mitigate forgetting by image-level replay. Table 8.1 reports and defines key notation used in this chapter.

8.3.1 Architecture of Replay Block

To cope with this issue, we opt for a replay strategy. Our goal is to retrieve task-related knowledge of past classes to be blended into the ongoing incremental step, all without accessing training data of previous iterations. To this end, we introduce a Replay Block, whose target is twofold:

- First, it has to provide images resembling instances of classes from previous steps, whether generating them from scratch or retrieving them from an available alternative source (*e.g.*, a web database).
- Second, it has to obtain reliable semantic labels of those images, by resorting to learned knowledge from past steps.

The Replay Block's image retrieval task is executed by what we call Source Block:

$$S : \mathcal{C}_t \mapsto \mathcal{X}^{rp}_{\mathcal{C}_t}. \tag{8.2}$$

This module takes in input a set of classes $\mathcal{C}_t$ (background excluded) and provides images whose semantic content can be ascribed to those categories (*e.g.*, $\mathbf{X}^{rp} \in \mathcal{X}^{rp}_{\mathcal{C}_t}$). We adopt two different solutions for the Source Block, namely GAN and web-based techniques, both detailed in Section 8.4.

The Source Block provides unlabeled image data (if we exclude the weak image-level classification labels), and for this reason we introduce an additional Label Evaluation Block $\{L_{\mathcal{C}_t}\}_{\mathcal{C}_t \subset \mathcal{C}}$, which aims at annotating examples provided by the replay module. This block is made of separate instances $L_{\mathcal{C}_t} = M^{d,H}_{\mathcal{C}_t} \circ M^e_0$, each denoting a segmentation model to classify a specific set of semantic categories $\mathcal{C}_t \cup \{b\}$ (*i.e.*, the classes in $\mathcal{C}_t$ plus the background):

$$L_{\mathcal{C}_t} : \mathcal{X}_{\mathcal{C}_t} \mapsto \mathbb{R}^{H \times W \times (|\mathcal{C}_t \cup \{b\}|)}. \tag{8.3}$$

All $L_{\mathcal{C}_t}$ modules share the encoder section M^e_0 from the initial training step, so that only a minimal portion of the segmentation network (*i.e.*, $M^{d,H}_{\mathcal{C}_t}$, which accounts for only few parameters, see Section 8.6.5) is stored for each block's instance. Notice that a single instance recognizing all classes could be used, leading to an even more compact representation, but it experimentally led to worse performance.

Provided that S and $L_{\mathcal{C}_t}$ are available, replay training data can be collected for classes in $\mathcal{C}_t$. A query to S outputs a generic image example $\mathbf{X}^{rp}_{\mathcal{C}_t} = S(\mathcal{C}_t)$, which is then associated to its prediction:

$$\mathbf{Y}^{rp}_{\mathcal{C}_t} = \underset{c \in \mathcal{C}_t \cup \{b\}}{\arg\max}\, L_{\mathcal{C}_t}(\mathbf{X}^{rp}_{\mathcal{C}_t})[c]. \tag{8.4}$$

By retrieving multiple replay examples, we build a replay dataset $\mathcal{R}_{\mathcal{C}_t} = \{(\mathbf{X}^{rp}_{\mathcal{C}_t}, \mathbf{Y}^{rp}_{\mathcal{C}_t})_n\}_{n=1}^{N_r}$, where N_r is a fixed hyper-parameter empirically set (see section 8.5).

8.3.2 Background Self-Inpainting

To deal with the background shift phenomenon, we propose a simple yet effective inpainting mechanism to transfer knowledge from the previous model into the current one. While the replay block re-creates samples of previously seen classes, background inpainting acts on background regions of current samples reducing the background shift and at the same time bringing a regularization effect similar to knowledge distillation [3, 118], although its implementation is quite different. At every step t with training set $\mathcal{T}_t$, we take the background region of each ground-truth map and we label it with the associated prediction from the pre-

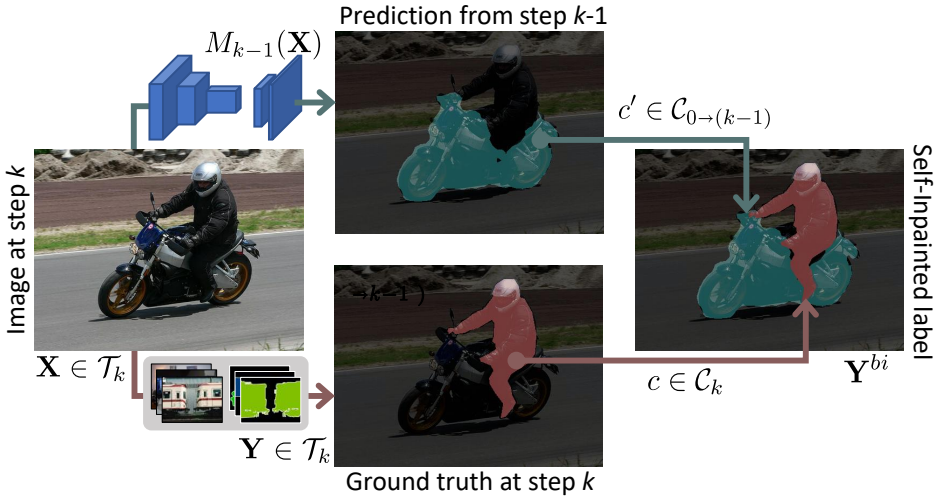


Figure 8.3: Background self-inpainting process.

vious model M_{t-1} (see Figure 8.3). We call it *background inpainting* since the background regions in label maps are changed according to a self-teaching scheme based on the prediction of the old model. More formally, we replace each original label map $\mathbf{Y}$ available at step $t > 0$ with its inpainted version $\mathbf{Y}^{bi}$:

$$\mathbf{Y}^{bi}[h,w] = \begin{cases} \mathbf{Y}[h,w] & \text{if } \mathbf{Y}[h,w] \in \mathcal{C}_t \\ \arg\max_{c \in \mathcal{C}_{0:t-1}} M_{t-1}(\mathbf{X})[h,w][c] & \text{otherwise} \end{cases} \tag{8.5}$$

where $(\mathbf{X}, \mathbf{Y}) \in \mathcal{T}_t$, while $[h, w]$ denotes the pixel coordinates. Labels at step $t = 0$ are not inpainted, as at that stage we lack any prior knowledge of past classes. When background inpainting is performed, each set $\mathcal{T}_t^{bi} \subset \mathcal{X} \times \mathcal{Y}_{\mathcal{C}_{0:t}}$ $(t > 0)$ contains all samples of $\mathcal{T}_t$ after being inpainted.

8.3.3 Incremental Training with Replay Block

The training procedure of RECALL is detailed and summarized in Algorithm 8.1 and the process is described in Figure 8.2. Suppose we are at the incremental step t, with only training data of classes in $\mathcal{C}_t$ from partition $\mathcal{T}_t$ available. In a first stage, the Replay Block is fixed and it is used to retrieve annotated data for steps from 0 to $t-1$ uniformly distributed among all the past classes. Following the described pipeline, the generative and labeling models are applied independently over each incremental class set $\mathcal{C}_i$, $i = 0, \ldots, t-1$. The replay training

Algorithm 8.1 RECALL: incremental training procedure.

Input: $\{\mathcal{T}_t\}_{t=0}^{T}$ and $\{\mathcal{C}_t\}_{t=0}^{T}$
Output: M_T
train $M_0 = M_0^d \circ M_0^e$ with $\mathcal{L}_{ce}(M_0; \mathcal{C}_0, \mathcal{T}_0)$
train S on $(\mathcal{C}_0, \mathcal{T}_0)$
train $M_{\mathcal{C}_0}^{d,H}$ with $\mathcal{L}_{ce}(L_{\mathcal{C}_0}; \mathcal{C}_0, \mathcal{T}_0)$
for $t \leftarrow 1$ to T **do**
  background inpainting on $\mathcal{T}_t$ to obtain $\mathcal{T}_t^{bi}$
  train S on $(\mathcal{C}_t, \mathcal{T}_t)$
  train $M_{\mathcal{C}_t}^{d,H}$ with $\mathcal{L}_{ce}(L_{\mathcal{C}_t}; \mathcal{C}_t \cup \{b\}, \mathcal{T}_t)$
  generate $\mathcal{T}_t^{rp} = \mathcal{T}_t^{bi} \cup \mathcal{R}_{\mathcal{C}_{0:t-1}}$
  train M_t^d with $\mathcal{L}_{ce}(M_t; \mathcal{C}_{0:t}, \mathcal{T}_t^{rp})$
end for

dataset for step t is the union of the single replay sets for each previous step

$$\mathcal{R}_{\mathcal{C}_{0:t-1}} = \bigcup_{i=0}^{t-1} \mathcal{R}_{\mathcal{C}_i}. \tag{8.6}$$

Once we have assembled $\mathcal{R}_{\mathcal{C}_{0:t-1}}$, by merging it with $\mathcal{T}_t^{bi}$ we get an augmented step-t training partition $\mathcal{T}_t^{rp} = \mathcal{T}_t^{bi} \cup \mathcal{R}_{\mathcal{C}_{0:t-1}}$. This new set, in principle, is complete of annotated samples containing both old and new classes, thanks to replay data. Therefore, we effectively learn the segmentation model M_t through the cross-entropy objective $\mathcal{L}_{ce}(M_t; \mathcal{C}_{0:t}, \mathcal{T}_t^{rp})$ on replay-augmented training data. This mitigates the bias toward new classes, thus preventing forgetting.

In a second stage, we exploit $\mathcal{T}_t$ to train the Class Mapping Module if needed (see Section 8.4). In particular, we teach the Source Block S to produce samples of $\mathcal{C}_t$, and we optimize the decoder $M_{\mathcal{C}_t}^{d,H}$ to correctly segment, in conjunction with M_0^e, images from $\mathcal{T}_t$ by minimizing $\mathcal{L}_{ce}(L_{\mathcal{C}_t}; \mathcal{C}_t \cup \{b\}, \mathcal{T}_t)$. This stage is not exploited in the current step, but will be necessary in future ones.

During a standard incremental training stage, we follow a mini-batch gradient descent scheme, where batches of annotated training data are sampled from $\mathcal{T}_t^{rp}$. However, to guarantee a proper stream of information, we opt for an interleaving sampling policy, rather than a random one. In particular, at a generic iteration of training, a batch of data $\mathcal{B}^{rp}$ supplied to the network is made of r_{new} samples from the current training partition $\mathcal{T}_t^{bi}$ and r_{old} replay samples from $\mathcal{R}_{\mathcal{C}_{0:t-1}}$. The ratio between r_{new} and r_{old} controls the proportion of replay and new data (see also Section 8.6.5). We need, in fact, to carefully balance how new data is

dosed with respect to replay one, so that enough information about new classes is provided within the learning process, while concurrently we assist the network in recalling knowledge acquired in past steps to prevent catastrophic forgetting.

8.4 Replay Strategies

In this section we describe more in detail the replay strategies employed for the image generation task of the Source Block S. As mentioned previously, we opt for a generative approach based on a GAN framework and for an online retrieval solution, where images are collected by a web crawler.

8.4.1 Replay by GAN

The GAN-based strategy exploits a deep generative adversarial framework to re-create the no longer available samples for previously seen classes. We use a conditional GAN, G, pre-trained on a generic large-scale visual dataset with data from a wide set of semantic classes $\mathcal{C}^G$ and different domains. For the experiments, we choose an ImageNet [26] based pre-training. On this regard, we remark that classes and domains are not required to be completely coherent: for instance *person* does not exist in ImageNet, but related classes (*e.g.*, *hat*) still allow to preserve its knowledge (further considerations on this are reported in Section 8.6.5). When performing the t-th incremental step, we retrieve images containing previously seen classes by sampling the GAN's generator output, *i.e.*, $\mathbf{X}^{rp} = G(\mathbf{n}, c^G)$ conditioned on GAN's classes $c^G \in \mathcal{C}^G$ corresponding to the target ones from the original training data ($\mathbf{n}$ is a generic noise input).

Since the GAN is pre-trained on a separate dataset, typically it inherits a different label set. For this reason, the Source Block with GAN is composed of two main modules, namely the actual GAN for image generation and a Class Mapping Module to translate each class of the semantic segmentation incremental dataset to the most similar class of the GAN's training dataset. Provided that we have trained both the GAN and class mapping modules, first we use the latter to translate the class set $\mathcal{C}_t$ to the matching set $\mathcal{C}_t^G$. Then, a set of queries to the conditioned GAN's generator:

$$\mathbf{X}_{\mathcal{C}_t}^{rp} = G(\mathbf{n}, c^G), \quad c^G \in \mathcal{C}_t^G \tag{8.7}$$

provides samples resembling the ones in $\mathcal{C}_t$, as long as the mapping is able to properly associate each original class to a statistically similar counterpart in the GAN's label space.

At each incremental step t, the Source Block with GAN goes through two separate training and inference stages. In a first training phase, samples from $\mathcal{T}_t$ are fed to an Image Classifier I, which is pre-trained to solve an image classification task on the GAN's dataset. In particular, for each class $c \in \mathcal{C}_t$ we select the corresponding training subset $\mathcal{T}_t^c \subset \mathcal{T}_t$, *i.e.*, all the samples of set $\mathcal{T}_t$ associated to class c, and we sum the resulting class probability vectors from the classification output. Then, the GAN's class c^G with the highest probability score is identified by:

$$c^G = \arg\max_{j \in \mathcal{C}^G} \sum_{\mathbf{X} \leftarrow \mathcal{T}_t^c} I(\mathbf{X})\,[j]\,, \tag{8.8}$$

where $\mathbf{X}$ is extracted from $\mathcal{T}_t^c$ (labels are not used) and $I(\mathbf{X})$ denotes the vector output of the last softmax layer of I, whose j-th entry corresponds to the j-th GAN's class. By repeating this procedure for every class in $\mathcal{C}_t$, we build the mapped set $\mathcal{C}_t^G$. Class correspondence is stored, so that at each step we have access to class mappings of past iterations.

In a second evaluation phase, classes in $\mathcal{C}_{0:t\text{-}1}$ are provided as input to the Source Block. Thanks to the class correspondences saved in previous steps, $\mathcal{C}_{0:t\text{-}1}$ are mapped to $\mathcal{C}_{0:t\text{-}1}^G$. Next, image generation conditioned on each class of $\mathcal{C}_{0:t\text{-}1}^{GAN}$ is performed, and the resulting replay images are fed to the Label Evaluation Block to be associated to their corresponding semantic labels. By following this procedure, we end up with self-annotated data of past classes suitable to support the supervised training at the current step, which otherwise would be limited to new classes.

8.4.2 Replay by Web Crawler

As an alternative we propose to retrieve training examples from an online source. For the evaluation, we searched images from the $Flickr$ website, but any other online database or search engine can be used.

Assuming we are at the incremental step t and we have access to the names of every class in the past iterations (*e.g.*, $\forall c \in \mathcal{C}_{0:t\text{-}1}$), we download images whose tag and description happen to both contain the class name through the $Flickr$'s web crawler. Then, the web-crawled images are fed to the Label Evaluation Block for their annotation.

Compared to the GAN-based approach, the online retrieval solution is simpler as no learnable modules are introduced. In addition, we completely avoid to assume that a larger dataset is available, whose class range should be sufficiently ample and diverse to cope with the continuous stream of novel classes incrementally introduced. On the other side, this approach requires the availability of an internet connection and in some way exploits additional training data even if almost unsupervised. Plus, we lack control over the weak labeling performed by the web source.

8.5 Experimental Setup

We use the DeepLab-V2 [11] as segmentation architecture with ResNet-101 [100] as backbone. Nonetheless, RECALL is independent of the specific network architecture. Encoder's weights are pre-trained on ImageNet [26] and all network's weights are trained in the initial step 0. In the following steps, only the main decoder is trained, together with the additional $\{M_{\mathcal{C}_t}^{d,H}\}_t$ helper decoders, which are needed to annotate replay samples (as discussed in Section 8.3). For fair comparison, all competing approaches are trained with the same backbone. SGD with momentum is used for weights optimization, with initial learning rate set to $5\text{e}{-4}$ and decreased to $5\text{e}{-6}$ according to polynomial decay of power 0.9. Following previous works [118, 119], we train the model for $|\mathcal{C}_t| \times 1000$ learning steps in the disjoint setup and for $|\mathcal{C}_t| \times 1500$ steps in the overlapped setup. Each helper decoder $M_{C_t}^{d,H}$ is trained with a polynomially decaying learning rate starting from $2\text{e}{-4}$ and ending at $2\text{e}{-6}$ for $|\mathcal{C}_t| \times 1000$ steps. As Source Block, we use BigGAN-deep [209] pre-trained [210] on ImageNet. At each incremental step t, we generate 500 replay samples per old class, *i.e.*$N_r = 500$. To map classes from the segmentation dataset to the GAN's one, we use the EfficientNet-B2 [211] classifier implemented at [212] and pre-trained on ImageNet. The interleaving ratio r_{old}/r_{new} is set to 1. As input pre-processing, random scaling and mirroring are followed by random padding and cropping to 321×321 px. The entire framework is developed in TensorFlow [213] and trained on a single GPU.

8.6 Experimental Results

In this section, we detail the experimental evaluation on the Pascal VOC 2012 dataset [197]. Following previous works on this topic [3, 115, 118, 119], we start by analyzing the performance on three widely used incremental scenarios: *i.e.*, the addition of the last class (19-1), the addition of the last 5 classes at once (15-5) and the addition of the last 5 classes sequentially (15-1). Moreover, we report the performance on three more challenging scenarios in which 10 classes are added sequentially one by one (10-1), in 2 batches of 5 elements each (10-5) and all at once (10-10). Classes for the incremental steps are selected according to the alphabetical order. We compare with the naïve fine-tuning approach (FT), which defines the lower limit to the accuracy of an incremental model, and with the joint training on the complete dataset in one step, which serves as upper bound. We also report the results of a simple Store and Replay (S&R) method, where at each incremental step we store a certain number of true samples for newly added classes, such that the respective size in average matches the size of the helper decoders needed by RECALL (see Figure 8.8).

Table 8.2: mIoU on Pascal VOC2012 for different incremental setups. Results of competitors in the upper part come from [2, 3], while we run their implementations for the new scenarios in the lower part. The highest values have been highlighted in bold.

	19-1						10-10					
	Disjoint			Overlapped			Disjoint			Overlapped		
Method	1-19	20	all	1-19	20	all	1-10	11-20	all	1-10	11-20	all
FT	35.2	13.2	34.2	34.7	14.9	33.8	7.7	60.8	33.0	7.8	58.9	32.1
S&R	55.3	43.2	56.2	54.0	48.0	55.1	25.1	53.9	41.7	18.4	53.3	38.2
LwF [180]	65.8	28.3	64.0	62.6	23.4	60.8	63.1	61.1	62.2	**70.7**	63.4	**67.2**
LwF-MC [112]	38.5	1.0	36.7	37.1	2.3	35.4	52.4	42.5	47.7	53.9	43.0	48.7
ILT [118]	66.9	23.4	64.8	50.2	29.2	49.2	**67.7**	**61.3**	**64.7**	70.3	61.9	66.3
CIL [4]	62.6	18.1	60.5	35.1	13.8	34.0	37.4	60.6	48.4	38.4	60.0	48.7
MiB [3]	69.6	25.6	67.4	**70.2**	22.1	67.8	66.9	57.5	62.4	70.4	63.7	67.2
SDR [2]	**69.9**	37.3	**68.4**	69.1	32.6	67.4	67.5	57.9	62.9	70.5	**63.9**	**67.4**
RECALL (GAN)	65.2	**50.1**	65.8	67.9	53.5	68.4	62.6	56.1	60.8	65.0	58.4	63.1
RECALL (Web)	65.0	47.1	65.4	68.1	**55.3**	**68.6**	64.1	56.9	61.9	66.0	58.8	63.7
Joint	75.5	73.5	75.4	75.5	73.5	75.4	76.6	74.0	75.4	76.6	74.0	75.4

	15-5						15-1					
	Disjoint			Overlapped			Disjoint			Overlapped		
Method	1-15	16-20	all	1-15	16-20	all	1-15	16-20	all	1-15	16-20	all
FT	8.4	33.5	14.4	12.5	36.9	18.3	5.8	4.9	5.6	4.9	3.2	4.5
S&R	38.5	43.1	41.6	36.3	44.2	40.3	41.0	31.8	40.7	38.6	31.2	38.9
LwF [180]	39.7	33.3	38.2	67.0	41.8	61.0	26.2	15.1	23.6	24.0	15.0	21.9
LwF-MC [112]	41.5	25.4	37.6	59.8	22.6	51.0	6.9	2.1	5.7	6.9	2.3	5.8
ILT [118]	31.5	25.1	30.0	69.0	46.4	63.6	6.7	1.2	5.4	5.7	1.0	4.6
CIL [4]	42.6	35.0	40.8	14.9	37.3	20.2	33,3	15.9	29.1	6.3	4.5	5.9
MiB [3]	71.8	43.3	64.7	**75.5**	49.4	69.0	46.2	12.9	37.9	35.1	13.5	29.7
SDR [2]	**73.5**	47.3	**67.2**	75.4	52.6	**69.9**	59.2	12.9	48.1	44.7	21.8	39.2
RECALL (GAN)	66.3	49.8	63.5	66.6	50.9	64.0	66.0	44.9	62.1	65.7	47.8	62.7
RECALL (Web)	69.2	**52.9**	66.3	67.7	**54.3**	65.6	**67.6**	**49.2**	**64.3**	**67.8**	**50.9**	**64.8**
Joint	77.5	68.5	75.4	77.5	68.5	75.4	77.5	68.5	75.4	77.5	68.5	75.4

	10-5						10-1					
	Disjoint			Overlapped			Disjoint			Overlapped		
Method	1-10	11-20	all	1-10	11-20	all	1-10	11-20	all	1-10	11-20	all
FT	7.2	41.9	23.7	7.4	37.5	21.7	6.3	2.0	4.3	6.3	2.8	4.7
S&R	26.0	28.5	29.7	22.2	28.5	27.9	30.2	19.3	27.3	28.3	20.8	27.1
LwF [180]	52.7	47.9	50.4	55.5	47.6	51.7	6.7	6.5	6.6	16.6	14.9	15.8
LwF-MC [112]	44.6	43.0	43.8	44.3	42.0	43.2	6.9	1.7	4.4	11.2	2.5	7.1
ILT [118]	53.4	48.1	50.9	55.0	44.8	51.7	14.1	0.6	7.5	16.5	1.0	9.1
CIL [4]	27.5	41.4	34.1	28.8	41.7	34.9	7.1	2.4	4.9	6.3	0.8	3.6
MiB [3]	54.3	47.6	51.1	55.2	49.9	52.7	14.9	9.5	12.3	15.1	14.8	15.0
SDR [2]	55.5	48.2	52.0	56.9	51.3	54.2	25.5	15.7	20.8	26.3	19.7	23.2
RECALL (GAN)	60.0	52.5	57.8	60.8	52.9	58.4	58.3	46.0	53.9	59.5	46.7	54.8
RECALL (Web)	**63.2**	**55.1**	**60.6**	**64.8**	**57.0**	**62.3**	**62.3**	**50.0**	**57.8**	**65.0**	**53.7**	**60.7**
Joint	76.6	74.0	75.4	76.6	74.0	75.4	76.6	74.0	75.4	76.6	74.0	75.4

As comparison, we include 2 methods extended from classification (*i.e.*, LwF [180] and its single-headed version LwF-MC [112]) and the most relevant methods designed for continual segmentation (*i.e.*, ILT [118], CIL [4], MiB [3] and SDR [2]). Exhaustive quantitative

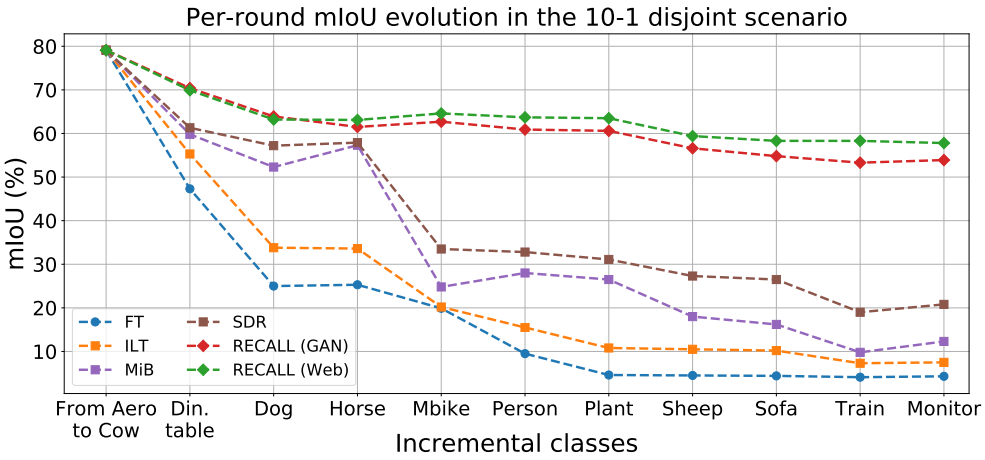


Figure 8.4: Evolution of performance in terms of mIoU on the 10 tasks of the 10-1 disjoint setup.

results in terms of mIoU are shown in Table 8.2. For each setup we report the mean accuracy for the initial set of classes, for the classes in the incremental steps and for all classes, computed after the overall training.

8.6.1 Addition of the last class

First, we train over the first 19 classes during step 0. Then, we perform a single incremental step to learn *tv/monitor*. Looking at Table 8.2 (upper-left section), we notice that FT results in a drastic performance degradation with respect to joint training, due to catastrophic forgetting. RECALL, instead, shows higher overall mIoU than competitors and it is especially effective on the last class, whilst still retaining high accuracy on the past ones thanks to the regularization brought in by background inpainting and replay strategies. S&R, instead, heavily forgets previous classes, thus confirming the usefulness of replay data.

8.6.2 Addition of last 5 classes

In this setup, 15 classes are learned in the initial step, while the remaining 5 are added in one shot (15-5) or sequentially one at a time (15-1). Compared to the 19-1 setup, the addition of multiple classes in the incremental iterations makes catastrophic forgetting even more severe. The accuracy gap between FT and joint training, in fact, raises from about 41% of the 19-1 case to more than 70% of mIoU in the 15-1 scenario. Taking a closer look at the results in Table 8.2 (upper mid and right sections), our replay approaches strongly limit the degradation caused by catastrophic forgetting. This trend can be observed in the 15-5 setup and more evidently in the 15-1 one, both in the disjoint and overlapped settings: exploiting gen-

erated or web-derived replay samples proves to effectively restore knowledge of past classes, leading to a final mIoU approaching that of the joint training. Storing and replaying original samples, instead, improves the performance with respect to FT, but ultimately leads to a mIoU lower of more than 20% if compared to our approaches. This is due to the limited number of samples to be stored in order to match the helper decoder size: their sole addition is, in fact, insufficient to adequately preserve learned knowledge. Finally, we observe that RECALL can scale much better than competitors when multiple incremental steps are performed (scenario 15-1), as typically encountered in real-world applications.

8.6.3 Addition of last 10 classes

To analyze the previous claim, we introduce some new challenging experiments, not evaluated in previous works. In these tests only 10 classes are observed in the initial step, while the remaining ones are added in a single batch (10-10), in 2 steps of 5 classes each (10-5), or individually (10-1). Again, FT is heavily affected by the information loss that occurs when performing incremental training without regularization, leading to performance drops up to about 71% of mIoU w.r.t. the joint training in the most challenging 10-1 setting. Thanks to the introduction of replay data, RECALL brings a remarkable performance boost to the segmentation accuracy and becomes more and more valuable as the difficulty of the settings increases. In the 10-10 case, our method achieves slightly lower mIoU results than competitors (although comparable). As we increase complexity, our approach is able to outperform competitors by about 8% of mIoU in 10-5 and by 37% of mIoU in 10-1. We remark how RECALL shows a convincing capability of providing a rather steady accuracy in different setups, regardless of the number of incremental steps used to introduce new classes. For example, in the disjoint scenario, when moving from simpler to more challenging setups (*i.e.*, from 10-10 to 10-1, passing through 10-5), the mIoU of FT drops as $33.0\% \rightarrow 23.7\% \rightarrow 4.3\%$ and the one of SDR (*i.e.*, the best compared approach) as $62.9\% \rightarrow 52.0\% \rightarrow 20.8\%$, while our approach maintains a stable mIoU trend of $61.9\% \rightarrow 60.6\% \rightarrow 57.8\%$. Finally, we report the mIoU after each incremental step on the 10-1 disjoint scenario in Figure 8.4, where our approaches show much higher mIoU at every learning step, indicating improved resilience to forgetting and background shift, than competitors. In the qualitative results in Figure 4.2 we observe that RECALL effectively alleviates forgetting and reduces the bias towards novel classes. In the first row, the *bus* is correctly preserved while FT, S&R and inpainting wrongly classify it as *train* (*i.e.*, one of the novel classes); in the second row, FT places *sheep* and *tv* (newly added classes) in place of *cow*; in the third row, some *horse*'s features are either mixed with those of *person* and *cat* or completely destroyed, while they are preserved by our methods (the web scheme shows higher accuracy than the GAN here).

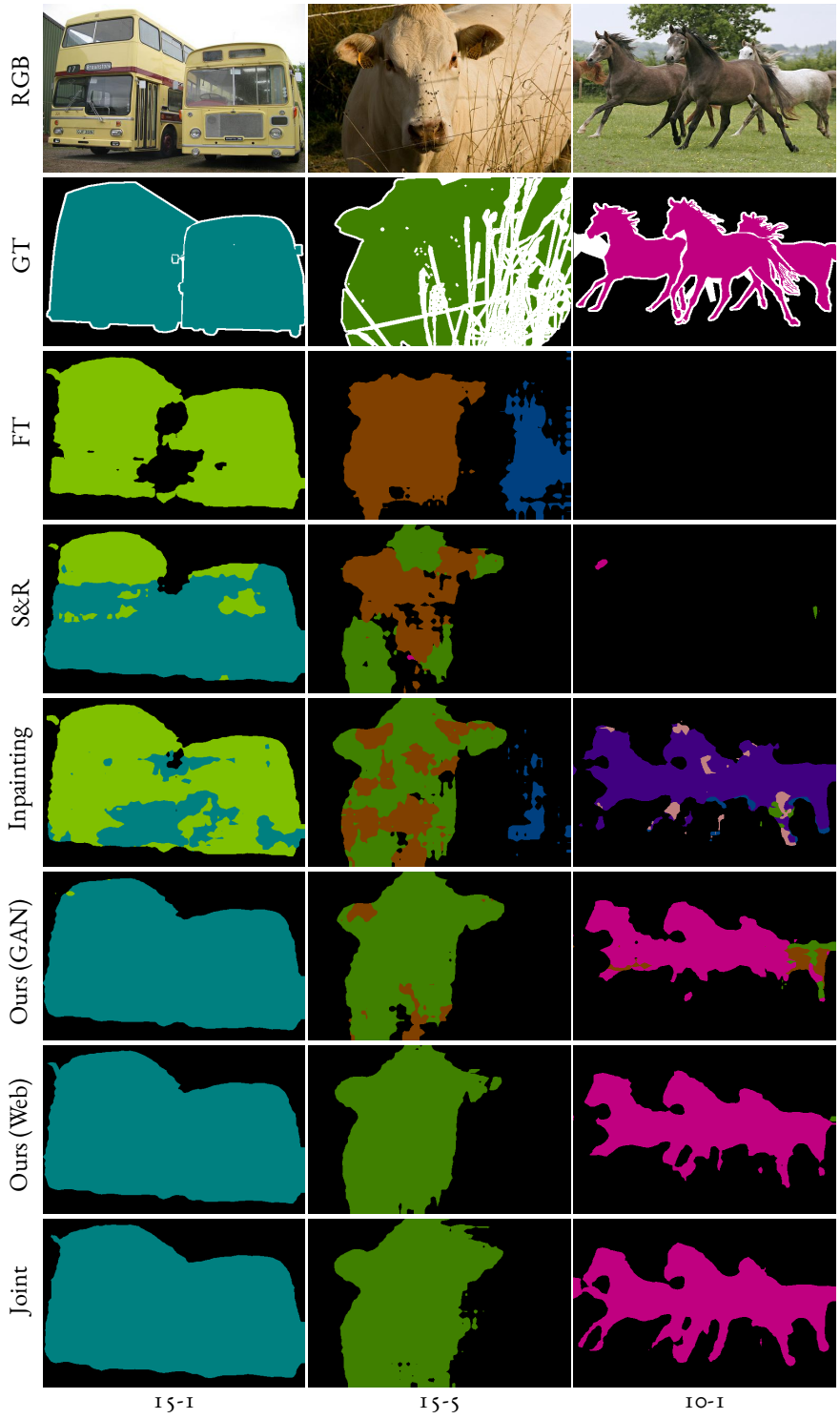


Figure 8.5: Qualitative results on disjoint incremental setups: from top to bottom 15-1, 15-5 and 10-1.

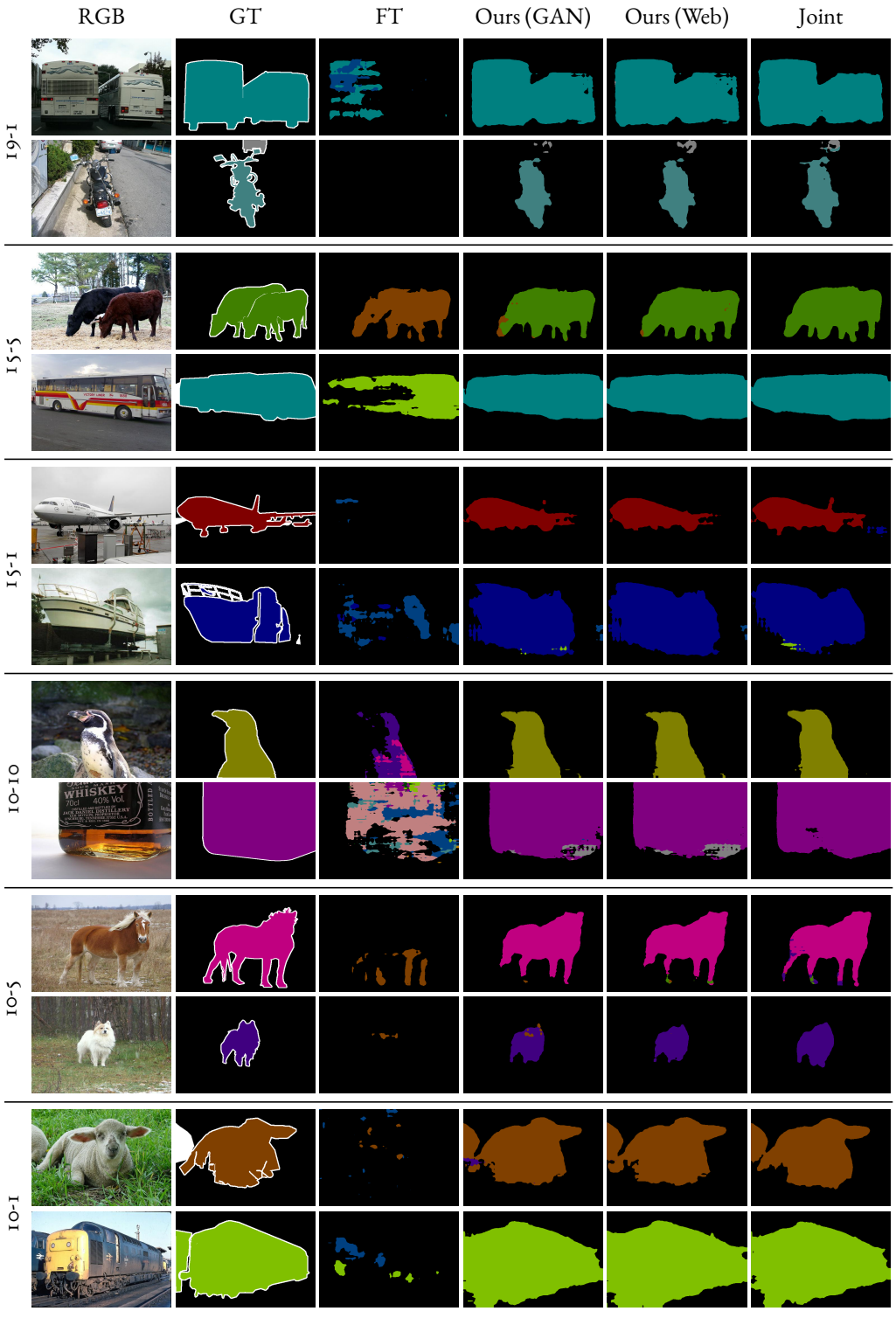


Figure 8.6: Qualitative results on disjoint incremental setups.

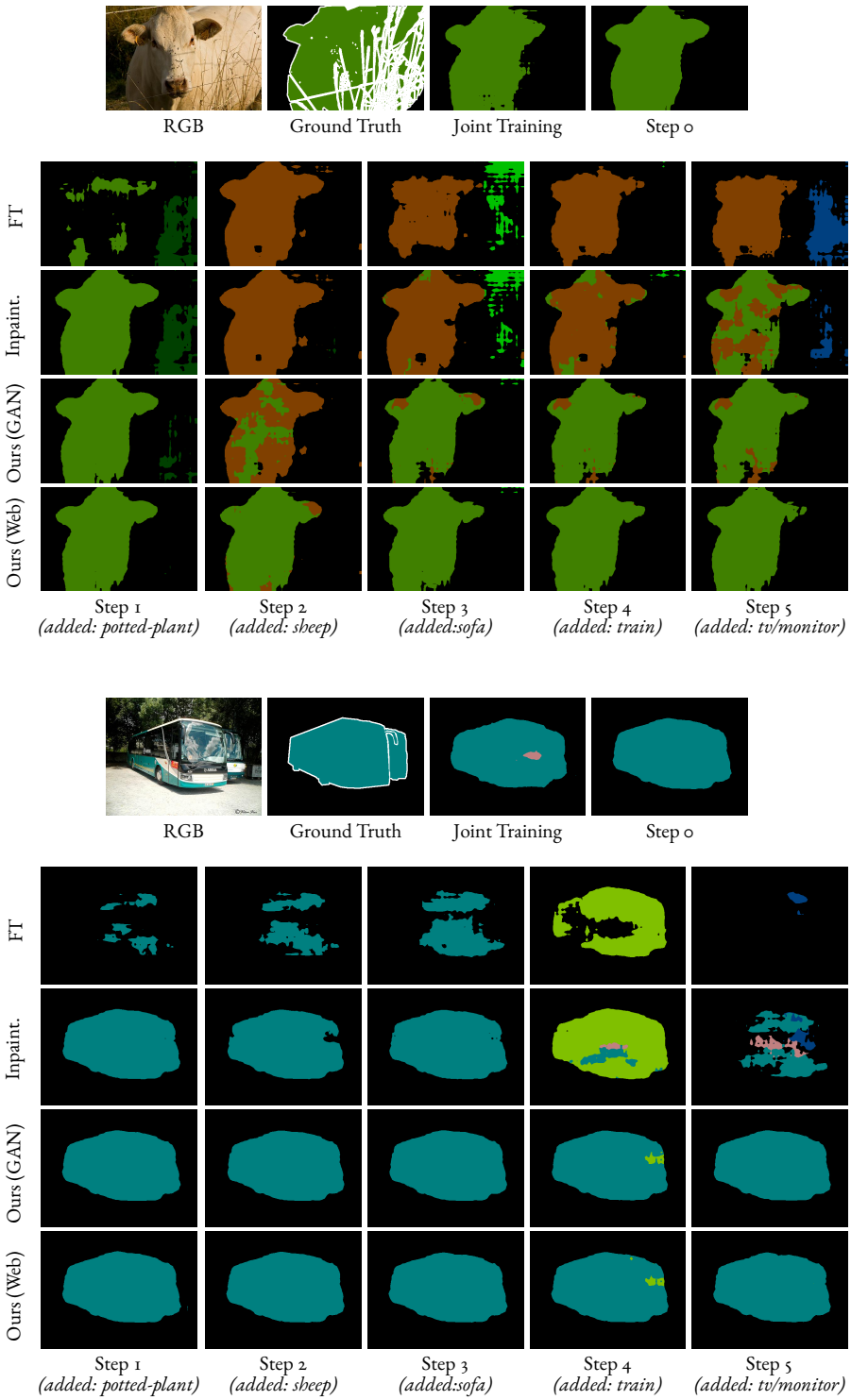


Figure 8.7: Per-step prediction maps on the 15-1 disjoint incremental setup for different training strategies.

Table 8.3: Per-class IoU of compared methods in disjoint experimental protocol on multiple scenarios of Pascal VOC 2012.

	Method	backgr.	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	din. table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	old	new	all
19-1	FT	72.4	62.4	6.7	45.0	47.1	39.5	33.7	40.9	25.7	4.3	54.0	8.0	25.0	50.4	50.6	0.0	35.3	43.0	0.8	59.5	13.2	35.2	13.2	34.2
	Inp.	91.0	83.9	35.1	77.3	62.3	70.7	77.9	73.4	85.7	31.5	73.1	48.0	81.3	74.4	64.6	81.0	44.1	75.7	41.3	74.5	30.4	66.1	30.4	65.6
	GAN	91.7	82.8	32.3	82.6	62.8	74.1	86.2	79.6	86.0	30.0	58.9	45.9	80.5	67.9	73.4	80.6	35.3	62.9	39.6	77.9	50.1	65.2	50.1	65.8
	Web	91.4	82.8	35.9	83.4	59.9	73.5	85.3	73.7	85.7	31.3	59.4	40.9	81.1	67.1	73.4	80.5	43.1	61.5	42.6	74.4	47.1	65.0	47.1	65.4
	Joint	92.5	89.9	39.2	87.6	65.2	77.3	91.1	88.5	92.9	34.8	84.0	53.7	88.9	85.0	85.1	84.9	60.0	79.7	47.0	82.2	73.5	75.5	73.5	75.4
15-5	FT	72.4	62.4	6.7	45.0	47.1	39.5	33.7	40.9	25.7	4.3	54.0	8.0	25.0	50.4	50.6	0.0	35.3	43.0	0.8	59.5	13.2	35.2	13.2	34.2
	Inp.	89.0	68.7	36.0	68.2	48.4	71.4	12.8	77.3	85.6	26.7	8.1	48.8	80.3	61.6	68.8	78.7	20.1	29.0	26.3	38.4	51.8	56.1	33.1	52.2
	GAN	90.4	78.8	35.0	79.5	60.3	75.7	79.3	78.7	85.9	22.8	55.0	46.6	80.0	67.4	72.1	77.8	37.3	60.2	32.2	64.4	55.1	66.3	49.8	63.5
	Web	90.8	82.2	35.5	81.7	63.9	75.3	85.0	77.8	86.3	28.0	67.5	48.7	81.0	72.7	73.8	78.0	40.4	65.7	31.9	69.1	57.6	69.2	52.9	66.3
	Joint	92.5	89.9	39.2	87.6	65.2	77.3	91.1	88.5	92.9	34.8	84.0	53.7	88.9	85.0	85.1	84.9	60.0	79.7	47.0	82.2	73.5	77.5	68.5	75.4
15-1	FT	74.2	27.2	0.0	1.6	15.1	11.3	0.0	4.1	0.5	0.0	0.0	0.0	0.0	0.2	0.2	0.0	27.0	25.6	28.9	33.5	52.2	8.4	33.5	14.4
	Inp.	85.9	38.9	31.4	79.4	41.5	71.3	28.9	62.6	85.6	32.2	29.6	50.2	76.6	69.2	55.3	80.2	18.5	37.4	36.3	19.8	17.9	55.5	26.0	49.9
	GAN	90.5	80.7	34.5	79.5	59.1	75.5	72.7	78.2	85.3	25.3	59.0	39.9	79.9	68.8	72.5	78.6	23.2	58.0	39.2	60.1	43.8	66.0	44.9	62.1
	Web	90.5	82.1	34.4	81.5	62.6	76.0	82.3	77.0	85.1	27.4	63.6	39.4	80.3	71.9	72.2	78.4	35.4	64.4	35.7	61.9	48.7	67.6	49.2	64.3
	Joint	92.5	89.9	39.2	87.6	65.2	77.3	91.1	88.5	92.9	34.8	84.0	53.7	88.9	85.0	85.1	84.9	60.0	79.7	47.0	82.2	73.5	77.5	68.5	75.4
10-10	FT	82.1	0.2	0.0	1.2	0.0	1.4	0.0	0.0	0.0	0.0	0.0	52.3	73.2	49.8	73.1	81.8	41.4	49.7	49.1	76.1	62.2	7.7	60.9	33.0
	Inp.	90.9	81.8	34.1	73.1	58.6	73.3	85.6	78.8	78.2	29.0	29.1	43.7	66.6	47.7	73.0	74.2	29.6	57.3	38.8	70.9	61.4	62.2	56.3	60.7
	GAN	90.8	83.3	30.4	75.8	61.4	73.5	80.8	77.2	72.8	23.6	46.8	48.0	65.4	55.3	66.1	72.5	36.8	58.3	36.1	67.1	55.6	62.6	56.1	60.8
	Web	90.9	82.3	32.7	75.4	63.2	72.8	81.7	73.5	76.2	24.2	58.5	46.5	68.8	60.2	64.7	73.3	38.3	58.3	34.2	68.5	56.2	64.1	56.9	61.9
	Joint	92.5	89.9	39.2	87.6	65.2	77.3	91.1	88.5	92.9	34.8	84.0	53.7	88.9	85.0	85.1	84.9	60.0	79.7	47.0	82.2	73.5	76.6	74.0	75.4
10-5	FT	78.2	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	16.4	13.7	23.4	55.7	46.7	37.4	39.8	47.9	75.6	62.3	7.2	41.9	23.7
	Inp.	88.5	58.8	31.9	55.4	58.2	69.2	0.2	78.3	83.2	28.2	5.0	36.4	71.6	34.7	61.4	74.1	20.4	26.2	25.5	34.0	47.9	46.8	43.2	47.1
	GAN	89.3	77.9	28.9	72.1	59.3	73.6	75.1	75.8	79.5	20.5	37.2	44.2	67.8	50.9	59.5	71.3	31.4	51.9	32.3	63.1	53.0	60.0	52.5	57.8
	Web	89.5	80.8	31.2	74.6	61.6	72.0	81.6	74.3	80.6	19.8	55.1	44.3	69.2	56.9	56.5	71.7	39.8	59.0	30.2	69.7	54.2	63.2	55.1	60.6
	Joint	92.5	89.9	39.2	87.6	65.2	77.3	91.1	88.5	92.9	34.8	84.0	53.7	88.9	85.0	85.1	84.9	60.0	79.7	47.0	82.2	73.5	76.6	74.0	75.4
10-1	FT	69.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	19.9	6.3	2.0	4.3
	Inp.	85.0	35.3	28.6	72.6	37.2	67.9	16.9	65.3	83.0	32.1	13.3	35.1	41.4	5.0	22.5	71.7	21.5	16.9	34.5	19.2	13.0	45.2	28.1	39.0
	GAN	88.8	77.9	26.6	71.8	58.6	73.2	63.5	74.0	75.7	20.5	41.3	34.2	60.5	43.6	58.6	66.2	15.8	51.7	37.4	53.0	38.8	58.3	46.0	53.9
	Web	89.1	79.1	31.0	74.4	62.2	66.5	81.7	74.1	78.7	19.4	56.2	41.8	62.7	58.5	62.1	66.6	8.5	59.3	36.8	59.2	45.0	62.3	50.0	57.8
	Joint	92.5	89.9	39.2	87.6	65.2	77.3	91.1	88.5	92.9	34.8	84.0	53.7	88.9	85.0	85.1	84.9	60.0	79.7	47.0	82.2	73.5	76.6	74.0	75.4

8.6.4 Per-Task and Per-Class Results

For a more detailed evaluation, we present the per-class IoU values for some of the proposed approaches and scenarios. We considered the following methods in the disjoint scenario on all the experimental protocols: fine-tuning (FT), background inpainting, RECALL (GAN), RECALL (Web) and joint training. The results are summarized in Table 8.3. From here, we can appreciate how fine-tuning always catastrophically forgets previous classes when learning new ones. The simple background inpainting strategy allows to largely alleviate such phenomenon bringing a similar effect to recent knowledge distillation approaches [3, 119]. On top of this, we apply GAN or Web-based replay strategies to regularize training and background content inpainting scheme to reduce bias toward the background. While these strategies are specifically designed to preserve old knowledge, they also allow to achieve large mIoU gains on new classes reducing the false positive rate (*i.e.*, the detection of new classes in locations containing the old ones).

In order to better understand the effect of our proposed modules, we report in Table 8.4 the Pixel Accuracy (PA) and the IoU for the class being added at each step of the disjoint 10-1 scenario. The results demonstrate that, on the newly introduced class, FT generally achieves a very high PA (top-left) and a per-class IoU (top-right) comparable to the other approaches. Yet, FT concurrently shows very low mIoU over all classes learned up to the current step (bottom-right), as well as over only previously seen categories (bottom-left). All combined, this is indicative of an overestimation of the new class. In other words, FT progressively forgets foregoing semantic information, while predicting more often the newly seen class (which experiences high PA but low IoU, due to many false positive predictions). Our approach, instead, can effectively improve knowledge preservation thanks to replay data and background inpainting, providing steady mIoU results throughout the incremental steps.

8.6.5 Ablation Study

To further validate the robustness of our approach, we perform some ablation studies.

Memory Requirements

First of all, we analyze the memory requirements. The plot in Figure 8.8 shows in semi-log scale the memory occupation (expressed in MB) of the data to be stored at the end of each incremental step, as a function of the number of classes learned up to that point. We denote with *standard* an incremental approach which do not store any sample (*e.g.*, FT, LwF, ILT, MiB, SDR). The saved model generally corresponds to a fixed size encoder and a decoder, whose dimension slightly increases at each step to account for additional output channels

Table 8.4: Per-round accuracy measures in the 10-1 disjoint scenario. In the top part we report the PA (left) and IoU (right) of the last class currently introduced. The bottom part, instead, shows the mean IoU over the old classes up to the ongoing step (left), as well as the overall mean IoU including the new classes (right). The classes added at each incremental step are: 1:*dining table*, 2:*dog*, 3:*horse*, 4:*motorbike*, 5:*person*, 6:*potted plant*, 7:*sheep*, 8:*sofa*, 9:*train* and 10:*tv/monitor*. The highest values have been highlighted in bold.

PA (new)	step 1	step 2	step 3	step 4	step 5	step 6	step 7	step 8	step 9	step 10
FT	69.2	**90.9**	**87.1**	**79.2**	88.3	3.8	26.9	**49.9**	55.0	57.5
ILT [118]	70.2	88.3	44.3	52.8	86.9	58.7	22.5	43.6	48.8	56.3
MiB [3]	**75.5**	87.2	38.5	51.2	**89.6**	**61.0**	6.1	38.2	47.4	**60.2**
SDR [2]	68.7	73.2	34.0	31.0	84.5	55.6	15.7	39.0	45.1	55.8
RECALL (GAN)	24.0	56.7	38.1	58.5	76.2	27.1	36.8	25.4	53.0	45.0
RECALL (Web)	23.8	57.3	46.1	62.3	79.3	36.9	**42.8**	26.9	**64.2**	57.4

IoU (new)	step 1	step 2	step 3	step 4	step 5	step 6	step 7	step 8	step 9	step 10
FT	16.2	37.0	15.8	12.6	**78.4**	3.2	9.2	16.3	9.7	19.9
ILT [118]	14.6	37.2	14.5	11.5	68.6	5.5	4.9	10.3	11.1	13.8
MiB [3]	14.6	38.0	12.2	9.5	66.0	9.6	1.7	5.5	10.9	7.0
SDR [2]	**25.0**	54.0	13.3	10.9	69.2	10.9	5.4	8.9	15.4	17.8
RECALL (GAN)	22.4	53.9	36.1	54.6	64.4	23.6	34.4	23.6	49.3	38.8
RECALL (Web)	22.1	**54.5**	**43.7**	**58.4**	67.9	**30.5**	**39.6**	**24.7**	**58.3**	**45.0**

mIoU (old)	step 1	step 2	step 3	step 4	step 5	step 6	step 7	step 8	step 9	step 10
FT	50.4	23.9	26.1	20.5	4.6	4.7	4.2	3.7	3.8	3.5
ILT [118]	59.4	33.5	35.2	20.9	11.7	11.2	10.9	10.2	7.1	7.2
MiB [3]	64.3	53.6	61.1	26.0	25.3	27.6	19.0	16.8	9.7	12.6
SDR [2]	64.9	57.5	61.6	35.2	30.2	32.4	28.7	27.5	19.2	21.0
RECALL (GAN)	**74.7**	**64.7**	63.4	63.2	60.7	62.9	57.9	56.5	53.5	54.6
RECALL (Web)	74.3	63.9	**64.6**	**65.0**	**63.5**	**65.5**	**60.6**	**60.1**	**58.3**	**58.4**

mIoU (all)	step 1	step 2	step 3	step 4	step 5	step 6	step 7	step 8	step 9	step 10
FT	47.3	25.0	25.3	19.9	9.5	4.6	4.5	4.4	4.1	4.3
ILT [118]	55.3	33.8	33.6	20.2	15.5	10.8	10.5	10.2	7.3	7.5
MiB [3]	59.8	52.3	57.3	24.8	28.0	26.5	18.0	16.2	9.8	12.3
SDR [2]	61.3	57.2	57.9	33.5	32.8	31.1	27.3	26.5	19.0	20.8
RECALL (GAN)	**70.4**	**63.9**	61.5	62.7	60.9	60.6	56.6	54.8	53.3	53.9
RECALL (Web)	69.9	63.2	**63.1**	**64.6**	**63.7**	**63.5**	**59.4**	**58.3**	**58.3**	**57.8**

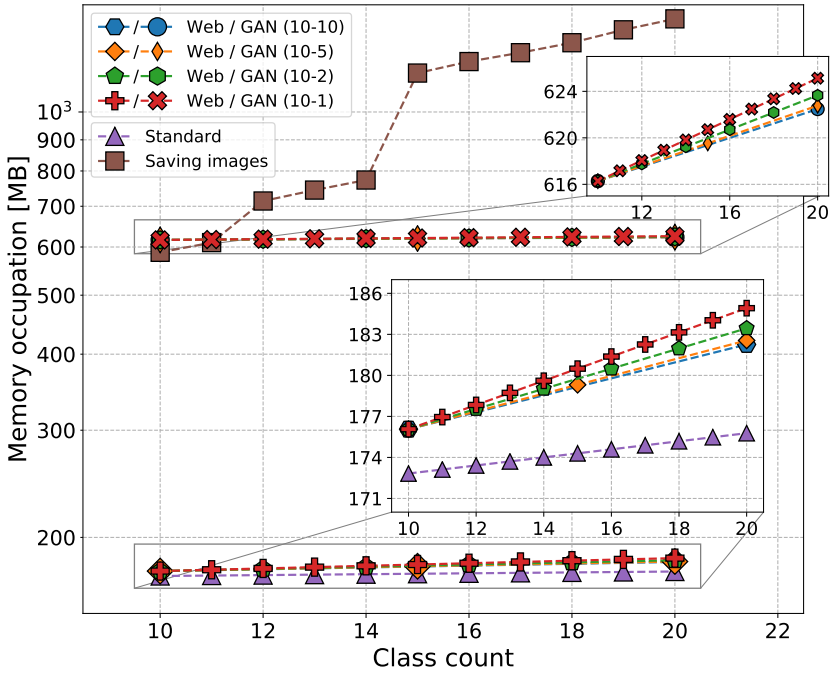


Figure 8.8: Memory occupation in the disjoint scenario.

for the new learnable classes. *Saving images*, instead, refers to the extreme scenario where training images of past steps are stored, thus being available throughout the entire incremental process. As concerns our approach, to annotate originally weakly-labeled replay images, we devise a specific module (Section 8.3), which requires to save a set of helper decoders $\{M_{\mathcal{C}_i}^{d,H}\}_{i=0}^{k}$, one for each past step. Finally, for the GAN-based approach we add the storage required for the generative model. Figure 8.8 shows that our web-based solution is very close to the *standard* ones in terms of memory occupation. The space required to store the GAN is comparable to that needed to save images in the very initial steps, but then remains constant while the space for saving all training data quickly grows.

Background Inpainting

We further analyze the contribution of the background inpainting and replay techniques in Table 8.5. While inpainting alone provides a solid contribution in terms of knowledge preservation acting similarly to knowledge distillation, we observe that its effect tends to attenuate with multiple incremental steps. For example, moving from 10-10 to 10-1 overlapped setups, the mIoU drops more than 20%. On the other hand, the proposed replay techniques prove to be beneficial when multiple training stages are involved. On the same setting, replay techniques alone limit the degradation to only 8%. Yet, jointly employing replay and inpainting

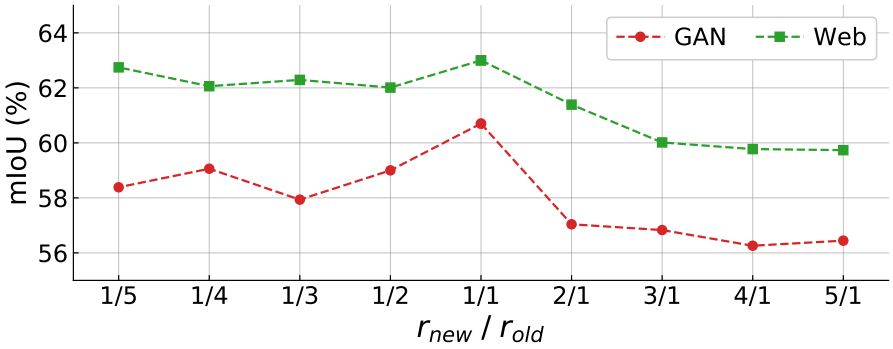


Figure 8.9: Distinct interleaving policies in the 15-1 disjoint setup.

Table 8.5: mIoU results showing the contribution of each module of our framework, D: Disjoint, O: Overlapped.

	19-1		15-5		15-1		10-10		10-5		10-1	
Method	D	O	D	O	D	O	D	O	D	O	D	O
Bgr inp.	65.6	66.7	52.2	52.5	49.7	49.9	58.8	60.7	47.5	47.1	34.0	39.0
GAN	54.5	56.2	49.8	49.1	47.9	48.2	45.8	48.8	38.1	43.7	36.6	40.8
Web	57.3	57.4	55.2	54.7	55.0	53.7	55.2	58.2	47.9	52.1	45.4	50.1
GAN+inp.	65.8	68.4	63.5	64.0	62.1	62.7	60.8	63.1	57.8	58.4	53.9	54.8
Web+inp.	65.4	68.6	66.3	65.6	64.3	64.8	61.9	63.7	60.6	62.3	57.8	60.7

further boosts the final results in all setups (up to 15%), proving that they can be effectively combined.

New vs Replay Data Balance

Finally, we analyze how results vary w.r.t. the proportion of new (r_{new}) and replay (r_{old}) samples seen during training (Figure 8.9): the mIoU is quite stable w.r.t. this ratio, however the maximum value is reached when the same number of old and replay samples is used, *i.e.*, $r_{new}/r_{old} = 1$.

RECALL as Additional Module

To the best of our knowledge, no works on continual semantic segmentation using GAN-generated or web-crawled data exist. The aim of our work is to provide a general framework to retrieve and employ unlabeled replay data. In this section, we demonstrate that our framework can be applied on top of competing approaches to improve their performance: some experimental results are shown in Table 8.6. Adding replay data with naïve pseudo-labeling (*i.e.*, using the decoder of the previous step) already leads to a performance improvement,

Table 8.6: mIoU on VOC2012 disjoint 15-1 with replay data. G: GAN, F: Flickr. Naïve: only decoder of last step is used for pseudo-labeling, Ours: our complete approach (RECALL) is used.

	none	+ Naïve (G)	+ Naïve (F)	+ Ours (G)	+ Ours (F)
ILT	5.4	37.8 (+32.4)	39.9 (+34.5)	49.6 (+44.2)	51.5 (+46.1)
MiB	37.9	49.3 (+11.4)	50.5 (+12.6)	63.5 (+25.6)	65.7 (+27.8)
SDR	48.1	53.1 (+05.0)	55.8 (+07.7)	65.5 (+17.4)	66.5 (+18.4)

but combining our method with previous approaches leads to much higher results with improvements ranging from 17% to 46%, proving the effectiveness and general applicability of the modules introduced in RECALL.

Class Mapping Module

Here we provide some further analysis and insights on the Class Mapping Module (introduced in Section 8.4.1), which is used to translate each class of the semantic segmentation incremental dataset (*e.g.*, Pascal VOC2012 [197]) to the most similar class of the GAN's training dataset (*e.g.*, ImageNet [26]). Notice that properly mapping the labels between the different domains is an important step, since incorrect pairings may easily harm the accuracy of the final model.

To solve the task, we took an Image Classifier I pre-trained to address an image classification task on the GAN's dataset. Then, for each class c in the current label set we select the corresponding training subset (*i.e.*, all the samples of the current training set associated to class c), and we sum the resulting class probability vectors from the classification output (according to I). An $\arg\max$ operation is then performed, to identify the GAN's class c_G with the highest probability score. To show the effectiveness of the proposed classification, we report in Table 8.7 the 3 classes from the GAN's dataset with the highest score for each class of the Pascal VOC2012 dataset. We can see that for all the classes the top selected pairings appear reasonable at first (notice that only the best matching class is selected in the proposed approach). At a closer look, we find that the classifier selects an unexpected label only in a single case, that is the *person* class being translated into *cowboy hat*; however, we remark that the ImageNet dataset does not contain the *person* class, thus inherently lacking a close match for that category. In light of this, we believe that the chosen class (*i.e.*, *cowboy hat*) is a reasonable choice and may still help in retaining high accuracy on the *person* class, being the *cowboy hat* always shown on top of people's heads. This situation is interesting as it shows the robustness of our approach not only to different domains with different statistical distributions (ImageNet domain versus Pascal VOC2012 one), but also to different labeling domains (the label set of VOC2012 is not a subset of the ImageNet one).

Pascal		ImageNet		
index	class	1st class	2nd class	3rd class
1	airplane	airliner	warplane	wing
2	bicycle	mountain bike	tandem	tricycle
3	bird	kite (bird)	dipper	quail
4	boat	catamaran	lakeside	fireboat
5	bottle	beer bottle	soda bottle	water bottle
6	bus	trolleybus	carriage	minibus
7	car	racing car	station wagon	minivan
8	cat	tabby cat	Egyptian cat	tiger cat
9	chair	rocking chair	dining table	folding chair
10	cow	ox	oxcart	water ox
11	dining table	dining table	china closet	restaurant
12	dog	Labrador retriever	pit bull terrier	beagle
13	horse	sorrel	ox	fox squirrel
14	motorbike	moped	scooter	disc brake
15	person	cowboy hat	crash helmet	crutch
16	potted plant	pot	pencil case	greenhouse
17	sheep	ram	llama	bighorn sheep
18	sofa	studio couch	quilt	rocking chair
19	train	carriage	electric locomotive	freight car
20	tv/monitor*	-	-	-

Table 8.7: Class mapping between Pascal VOC and ImageNet datasets. The table shows the 3 best matching ImageNet classes for each Pascal VOC 2012 class. (*): matching classes for *tv/monitor* are not computed since replay data is not needed.

The effectiveness of the mapping can also be visually appreciated from the sample generated images shown in Figure 8.10. In particular, notice how even for the *person* class mapped to the *cowboy hat* the images look reasonable, even if the variability in this case is much smaller if compared to the original VOC class data.

Figure 8.10: Original images from the incremental Pascal VOC dataset, together with replay data generated by GAN or retrieved by Flickr's web crawler.

Part III

Learning Under Task and Domain Shift

9 Transfer Learning Across Tasks and Domains

9.1 Introduction

Although capable of remarkable performance in narrow and confined tasks, deep models tend to struggle when confronted with continual learning of dynamic tasks in ever-changing environments. For instance, variable input distribution between supervised training data and target data has been shown to cause performance degradation, giving rise to the need for *domain adaptation*, which targets knowledge transferability across domains (Part I). Additionally, a major issue stands in the tendency to *catastrophically forget* previously acquired knowledge [166], with new information erasing that experienced so far (Part II). Both constitute critical problems when it comes to deploying deep models in practical applications, as in the real world it is very likely to encounter distribution variability both in terms of input data experienced and of target tasks. Nonetheless, the fundamental problem of continuously learning *and* adapting to novel environments remains open and is actively investigated, with a long way before its definitive solution.

A thriving research endeavour has been devoted to continual learning in vision problems, such as image classification [112, 113, 166], object detection [115, 214, 215] and, more recently, semantic segmentation [3, 118, 120]. As detailed in Section 6.3, the majority of those works, however, are limited to a *class incremental* perspective of the continual learning problem, where the focus is strictly posed on the variable task (*e.g.*, class) supervision and label-space shift experienced throughout the learning process. On the other side, a significant re-

search effort has been directed toward the domain adaptation problem, ranging from a static learning setting [30, 55, 75] to, quite recently, a dynamic perspective [132–134], taking into account incremental changes in the data distribution. Yet, all these works are generally focused only on the input distribution shift, neglecting the task-related side of the incremental learning.

In Chapter 10 we will propose a framework to address a general continual learning setup for semantic segmentation. The core idea is to build a robust distillation framework to retain and adapt former-task knowledge, while leveraging a stylization mechanism to effectively translate information across incremental domains.

The reminder of this chapter includes a formalization of the Continual Learning under both task and domain shift (Section 9.2); next, a survey of the limited literature tackling domain shift in more flexible settings w.r.t. the standard DA will be presented (Section 9.3); to conclude, we will provide a description of the proposed experimental setups and benchmarks to evaluate methods exploring the general CL problem (Section 9.4), which will be used in Chapter 10 to conduct the experimental campaign.

9.2 Problem Formulation

Similarly to Unsupervised Domain Adaptation (Part I) and Continual Learning under task shift (Part II), the more general continual learning setting in the presence of distribution variability within both input and label spaces can be considered as falling under the umbrella of transfer learning. Nonetheless, this under-explored comprehensive learning setting comprises stricter and more realistic constraints. On one hand, the general CL finds the same obstacle as task-focused CL in the tendency of deep prediction models to catastrophically forget former tasks, thus requiring learning mechanisms to retain previously acquired knowledge (as in the case of the CIL problem). On the other hand, the general CL setting calls for domain adaptation, as the input distribution continually shifting entails that learned clues must be robust and adaptable to mutable input appearance, while supervision comes only for a single (currently) experienced domain distribution (as with the UDA problem).

Here the overall goal is to transfer incrementally gained experience across learning steps; this requires to address the catastrophic forgetting phenomenon in order to retain task-related information, while spreading and adapting said knowledge to all the domains encountered. In other words, we ultimately seek to achieve satisfactory prediction performance on all tasks considered and in all domains experienced.

More formally, the learning process is divided into multiple steps $t = 0, 1, ..., T$, each characterized by its own training data $(\mathcal{X}_t, \mathcal{Y}_t)$ distributed according to $\mathbb{P}(\mathcal{X}_t, \mathcal{Y}_t)$ over $\mathcal{X}_t \times$

$\mathcal{Y}_t$. As in case of CIL, the challenge stands in the t-th training set being no longer available after the end of the t-th step. Moreover, single-step data represent a subset of a comprehensive domain $(\mathcal{X}, \mathcal{Y})$ associated to $\mathbb{P}(\mathcal{X}, \mathcal{Y})$ over $\mathcal{X} \times \mathcal{Y}$, with $\mathcal{X} = \bigcup_{t=0}^{T} \mathcal{X}_t$ and $\mathcal{Y} = \bigcup_{t=0}^{T} \mathcal{Y}_t$. We also assume that $\mathcal{Y}_{t+1} \neq \mathcal{Y}_t$, to represent the incrementally changing task supervision, similarly to what happens task-focused continual learning. Yet, we additionally enforce that $\mathcal{D}_{t+1} \neq \mathcal{D}_t$, to signify that input distribution is also mutating alongside tasks. The employed notation is the same characterizing Part II (see Table 6.1), and will be used throughout Part III as well.

Given a prediction model M_T incrementally trained, our goal is to solve the empirical risk minimization:

$$\text{argmin}_\theta \mathbb{E}_{(\mathcal{X},\mathcal{Y})} \left[\mathcal{L} \left(M_T \left(\mathcal{X}; \theta \right), \mathcal{Y} \right) \right], \tag{9.1}$$

with θ and $\mathcal{L}$ being respectively the model's parameters and the loss function. Direct optimization is not an option, due to partial task supervision and input distribution availability experienced incrementally. Thus, it is necessary to develop some techniques to reach satisfactory performance across all the tasks and domains encountered.

9.3 Adaptation with Variable Tasks or Domains

As thoroughly discussed in the first part of this dissertation, deep models are known to suffer performance degradation when presented with mutable input distribution between training and testing phases [216]. Domain adaptation has been extensively investigated to alleviate the aforementioned problem, by safely transferring learned knowledge from label-abundant source domains to label-scarce, or even unsupervised, target ones. We showed that particularly flourishing has been Unsupervised Domain Adaptation (UDA) for the semantic segmentation task [234], as supervision in terms of dense segmentation maps is usually very costly and time expensive to be collected for real-world data, and thus it is of prime interest to adapt knowledge from large-scale datasets already available. We refer to Chapter 2, where we presented an in-depth description of numerous UDA techniques. Yet, in its standard form, UDA entails no continual learning, being the task at hand the same on both source and target static domains, which are concurrently available. In the last part of the dissertation, instead, we are interested in a more realistic setup with dynamic task and domain evolution.

In the rest of thus section, we will report some recent works that have departed from the standard closed-set DA to study more realistic setups involving variable tasks or incremental domain shift. In particular, different variations of the static DA have been proposed, relaxing some of the original strict assumptions.

Domain-dependent Tasks

One research direction involves distinct tasks between source and target domains, *i.e.*, it allows source and target classes to be different. Depending on the relationship between source and target class sets, partial [217], open-set [218] and universal [28, 219–221] domain adaptation setups have been proposed, where the *universal* setup implies the largest flexibility concerning the target class set, which is not supposed to be known a priori and can contain both source and unknown categories. Nonetheless, most research has been confined to the image classification problem [218, 220, 221]. Moreover, these works do not involve class incremental learning, as adaptation is performed with simultaneous access to source and target domains in a single learning phase.

Continual Domain Shift

Another line of works has explored diverse setups in terms of domain availability. Some propose to handle multiple source [222, 223] or target [132–134, 224–227] domains. This can involve a single adaptation phase [222, 223], or multiple phases where different domains are experienced in different learning steps in a incremental fashion [132–134, 226, 227], in fact, undertaking continual learning under the domain adaptation perspective. Yet, all these works assume homogeneity of tasks across all the domains encountered, whereas the class and domain incremental setup we are investigating deals with variable learning conditions both along task and domain progressions.

Task and Domain Incremental Learning

It is possible to find in literature a few works that address both task incremental and domain adaptation problems.

Garg *et al.* [228] develop a multi-domain incremental learning (MDIL) framework that involves classification tasks shifting across multiple domains experienced in an incremental fashion. They rely on a dynamic architecture built upon a set of domain-invariant parameters to capture domain-invariant semantic clues shared by all domains, along with a set of domain-specific parameters apt to perceive the statistical properties peculiar to each domain. However, *total* supervision is assumed to be available on all the domains encountered, leading to overlapping incremental class sets. We, instead, adhere to a stricter and more realistic CIL setup, with disjoint groups of semantic categories incrementally introduced.

Furthermore, a few works focus on setups involving task and domain form of variability [229–231], whereas posing some constraints that differentiate their setup from a general task and domain continual learning. Kalb *et al.* [229] discuss class and domain incremental learn-

ing, but each problem is tackled individually by evaluating standard CIL and DA methods. Moreover, in [230], coarse-to-fine continual learning is explored, but the proposed setup does not involve domain shift across learning steps, as source and target domains are kept fixed.

Closer to the comprehensive CL, Simon *et al.* [231] address continual learning with tasks and domains dynamically evolving. They develop an additional learnable module to model class similarities, while introducing a knowledge distillation mechanism based on exponential moving averages of model parameters. Still, they assume to have task supervision on all the considered domains at each task incremental step, which may not be a realistic assumption in real-world applications. In addition, rehearsal of training exemplars is performed, and the method specifically targets image classification.

9.4 Experimental Setups

In this section, we detail the benchmarks we propose to evaluate the comprehensive task and domain continual learning, which will be used for validation purposes in Chapter 10. Most of these benchmarks were exploited for experimental analyses carried out for the UDA methods, as shown in Part I. However, in this last part of the dissertation we are focusing on how they can be leveraged to reproduce domain incremental shift.

To simulate the distribution shift at the input (image) level, we make use of multiple driving data sets, each limited to a specific geographic region or environmental factors, and thus characterized by its distinctive low-level appearance (*e.g.*, road pavement material, type of vehicles, light conditions). On the contrary, the high-level semantic content is mostly consistent across image sets, that is, the road-related or other categories, moving and static obstacles can be found everywhere, and follow similar inter-class structural relations (*e.g.*, the sky will always appear above the road).

Urban Datasets

Cityscapes. The Cityscapes [14] dataset (CS) is a popular benchmark for autonomous driving applications. Images are collected across 50 cities, all located in Central Europe, with almost the totality being within German borders. For this reason, scenes comprise for the most part street-like urban scenarios, with gray tones typical of weather conditions of this geographic area.

BDD100K. The Berkeley DeepDrive dataset (BDD) [88] is a more diverse collection of road scenes, captured with variable weather conditions at different times of the day. Com-

Figure 9.1: Domain distribution shift due to change of geographic location (*urban* setting).

Figure 9.2: Domain distribution shift due to change of geographic location (*worldwide* setting).

pared to Cityscapes, an urban environment is joined by suburban scenery with highway-centered landscape. Still, all samples are from 4 restricted localities in the United States.

IDD. The Indian Driving Dataset (IDD) [16] includes driving scenes from Indian cities and their outskirts. It offers a diversified set of moving and static road obstacles, as well as a wilder and more natural environment, which breaks away from the typical European or American urban scenarios of the two previous benchmarks.

By experiencing in succession the 3 aforementioned benchmarks in 3 different learning steps, it is possible to generate incremental distribution shift, as each single set provides peculiar visual features that differentiate it from the others, while the high-level semantic content (*i.e.*, driving scenery) is the same for all of them (as shown in Figure 9.1).

Worldwide Scene Collection

Mapillary Vistas. The Mapillary Vistas dataset [17] contains images collected worldwide, with highly diverse acquisition settings and locations. Unlike previously introduced benchmarks, samples are not limited to a few cities located within quite uniform geographic regions. It is possible leverage the Mapillary to generate continent-wise data splits (see Figure 9.2), as well as to test the domain generalization potential of the proposed class and domain incremental approach.

Figure 9.3: Domain distribution shift due to *environmental* factors.

Environmental Conditions

Shift. The Shift benchmark [89] is a synthetic dataset for autonomous driving, designed to provide a plethora of distribution shifts, simulating the highly variable environmental conditions faced in real-world applications. It can be exploited to mimic domain shift due to environmental diversity, such as due to variable illumination or weather conditions (as Figure 9.3 shows).

10 Generalized Continual Semantic Segmentation

10.1 Introduction

As discussed in the previous chapter, the general continual learning problem across both tasks and domains is yet unexplored for the semantic segmentation task. Where class incremental methods usually struggle to cope with domain knowledge transferability, domain incremental methods lack predisposition to address incremental task supervision. We instead propose to tackle continual semantic segmentation with joint incremental shift along class and domain directions. The training process involves multiple steps, each of which carries a new set of classes to learn, along with a training set comprising image samples with a step-distinctive distribution, differing from those experienced in previous steps, and supervision available only on the newly introduced class set. The overall objective is for the incremental segmentation model to deliver satisfactory performance across all the tasks (*i.e.*, class sets) and domains encountered so far, with the class- and domain- wise joint training as the target upper bound.

In this novel problem setup (see Figure 10.1), both domain adaptation and recollection of past classes must be performed to achieve satisfactory performance. Under the domain incremental angle, it is required to simultaneously learn new classes over past domains and adapt old-class knowledge to the new domain. From the class incremental perspective, recollection of past knowledge must take into account the variable input distribution characterizing the addressed incremental learning.

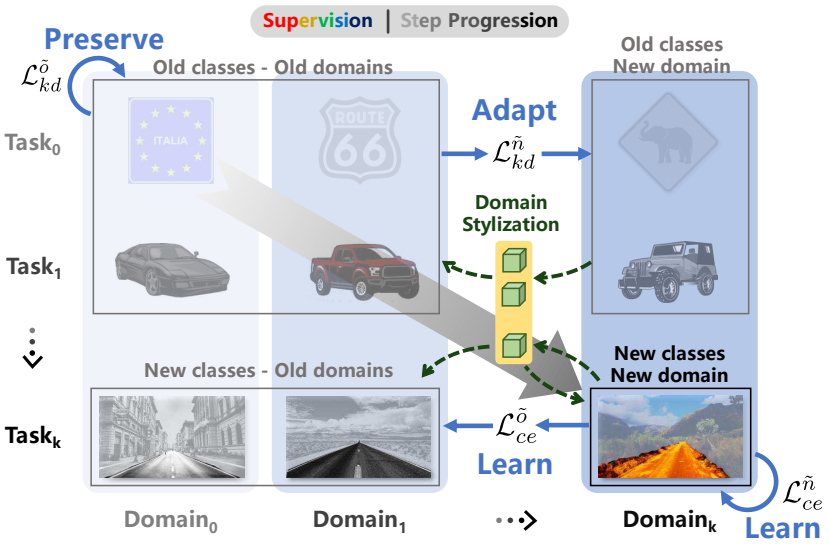


Figure 10.1: High-level view of our approach. Transparency decrease (top→down and left→right) indicates progression through learning steps. Colored task icons denote presence of supervision within training data, grayscale ones signal lack of supervision. At each step, we leverage training data to *learn* new classes on the new domain. Domain stylization allows to reiterate old-domain distribution, crucial to *learn* new tasks and *preserve* old ones on former domains, and to *adapt* old-domain old-task knowledge to new domains.

We therefore devise multiple training objectives to face underlying sub-problems. While to rehearse knowledge of old classes we resort to the old-step segmentation model, which is a common practice among class incremental learning methods [118], to replay information of past-domain input distribution we propose a stylization mechanism. The average style (*i.e.*, a very compact representation) of each encountered domain is computed and stored in a memory bank, to be transferred to novel domains in future steps and reproduce some domain-level information.

The overall optimization framework is made of (i) a standard task loss (*i.e.*, cross-entropy objective) to learn new classes over available training data, (ii) an additional task loss instance to learn new classes in old domains by leveraging stylization, (iii) a knowledge distillation-like objective to infuse adapted information of past classes in the form of hard pseudo-labels to the new domain and finally (iv) an output-level knowledge distillation objective applied on stylized images to retain old-domain old-class performance.

To summarize, our contributions in this chapter are as follows:

(i) We investigate a novel comprehensive incremental learning setting that accounts for variable distribution within both input and label spaces.

(ii) We develop a framework to tackle all facets of the class and domain incremental learning problem, based on a stylization mechanism to recall domain knowledge under

incremental task supervision and a robust distillation framework to retain task knowledge under incremental domain shift.

(iii) We devise novel experimental setups to simulate the proposed learning setting and conduct an extensive evaluation campaign.

(iv) We show that the proposed method outperforms existing state-of-the-art methods that address the IL problem only from a class or a domain incremental perspective.

In the rest of this chapter, we will start by providing a detailed formulation of the considered class and domain incremental learning targeting the semantic segmentation task (Section 10.2); we will then move to an overview of the proposed continual learning method (Section 10.3), analyzing each module of the overall framework (Section 10.4); we will finally conclude the chapter with a in depth description of the class and domain incremental setups we devise to simulate the general CL (Section 10.5), followed by an extensive experimental campaign to validate the proposed method (Section 10.6).

10.2 Problem Formulation

In semantic segmentation we aim at labeling every individual spatial location of an image by associating it with a semantic class taken from a predefined collection of candidates $\mathcal{C}$. That is, given an RGB image $\mathbf{X} \in \mathcal{X} \subset \mathbb{R}^{H \times W \times 3}$, a segmentation network $M : \mathcal{X} \mapsto \mathcal{Y}$ is exploited to provide its segmentation map $\hat{\mathbf{Y}} \in \mathcal{Y} \subset \mathcal{C}^{H \times W}$. $\hat{\mathbf{Y}}$ should be an accurate prediction of the ground-truth map $\mathbf{Y}$, which is available only at training time.

We follow an incremental learning protocol to optimize the segmentation network, as depicted in Figure 10.2. Specifically, the predictor is trained in multiple steps $t = 0, ..., T$ to recognize a progressively increasing set of semantic classes. At step t, a new class set $\mathcal{C}_t$ is introduced, along with training data $\mathcal{T}_t = \{(\mathbf{X}_t, \mathbf{Y}_t)\} \subset \mathcal{X}_t \times \mathcal{Y}_t$ associated to that set, which is available on the current image domain $\mathcal{X}_t$. The supervision provided by $\mathcal{T}_t$ is restricted to $\mathcal{C}_t$, meaning that any pixel within $\mathcal{T}_t$ is tagged in $\mathcal{Y}_t$ with $c \in \mathcal{C}_t$. At the end of the step, all the currently accessible data is discarded and is not reused again. The procedure is reiterated for multiple learning steps, with a new domain $\mathcal{X}_t$ and class set $\mathcal{C}_t$ being introduced and used for training at each step.
More formally, the objective is to train $M_t : \mathcal{X}_{0:t} \mapsto \mathcal{Y}_{0:t}$

- to recognize all the semantic classes observed up to the present step t:

$$\mathcal{Y}_{0:t} \in \mathcal{C}_{0:t}^{H \times W}, \quad \mathcal{C}_{0:t} = \bigcup_{k=0}^{t} \mathcal{C}_k, \tag{10.1}$$

Table 10.1: Formal definition of key notation used throughout Chapter 10.

Symbol	Definition
$t \in \{0, 1, 2, ..., T\}$	Step index
u	*Unknown* class
M_t, M_{t-1}	Current and past segmentation models at step t
$\hat{\mathbf{Y}}_t$	Segmentation prediction by M_t
$\hat{\mathbf{Y}}^{\mathcal{K}}_{t-1}$	Pseudo-label by M_{t-1} over stylized images from steps $k \in \mathcal{K}$
$\hat{\mathbf{Y}}^{<t}_{t-1}$	Pseudo-label by M_{t-1} over oldly-stylized images
$\hat{\mathbf{Y}}^{<t}_{t-1}$	Refined pseudo-label by M_{t-1} over oldly-stylized images
$\bar{F}^A_t$	Style of domain of step t
W_β	Style window, with size controlled by β
$\mathcal{M}^F_{0:t}$	Style memory bank up to step t
$\tilde{\mathcal{X}}_{t \to k}$	Domain of step t stylized as domain of step k
P^k_t	Prob. map by M_t over image with style of step k
$\mathring{P}_t$	Prob. map by M_t with *past* and *unknown* class probability channels grouped
$\overset{*}{P}_t$	Prob. map by M_t with *new* and *unknown* class probability channels grouped

- on all the image domains experienced so far:

$$\mathcal{X}_{0:t} = \bigcup_{k=0}^{t} \mathcal{X}_k. \tag{10.2}$$

We remark that $\{\mathcal{X}_t\}_{t=0}^{T}$ are characterized by diverse statistical properties, *i.e.*, domain shift occurs between them, typically manifested through cross-domain variable visual appearance of scene elements that yet share semantic significance. All $\mathcal{C}_t$ are disjoint sets, except for the *unknown* (u) class, which belongs to each of them. Class u at step t contains all the past and future classes. In other words, u undergoes a semantic shift across subsequent steps and, for this reason, demands special care when being handled [3].

10.3 Overview of the Proposed Method

We concurrently face challenges peculiar to both domain adaptation and class incremental learning settings.

Domain Adaptation. The segmentation network is trained on data from multiple domains, each holding only a subset of the whole set of the semantic classes. Even so, the model

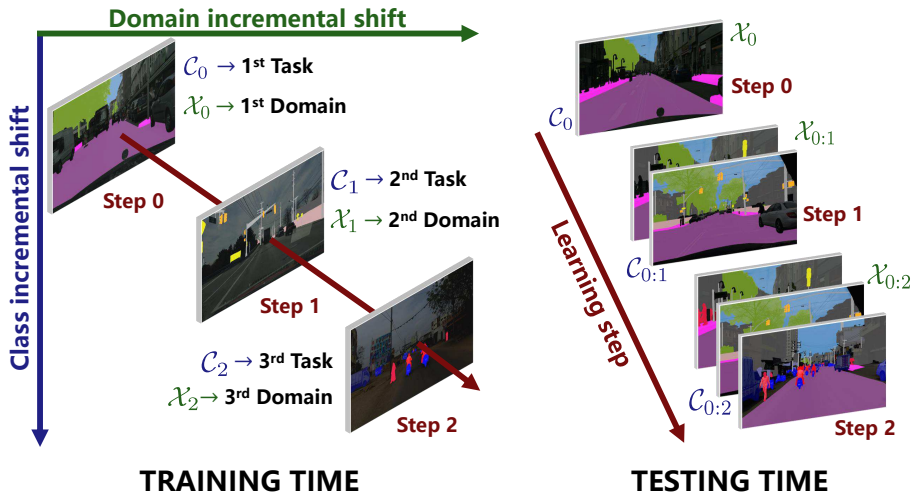


Figure 10.2: Overview of the class and domain incremental setup. When training, each step sees training data from a new domain, labeled on a new class set. When testing, performance is measured on all domains and classes experienced so far.

is expected to provide satisfactory prediction performance on all the observed domains and semantic classes. Hence, it is necessary to transfer knowledge across domains to

(i) learn *new-class* clues shared across the current (supervised) domain and the past ones (where new-class supervision was not available during past steps),

(ii) adapt *old-class* knowledge learned in former domains to the novel domain.

Class Incremental Learning. The different class supervision available on different domains leads us to a class incremental problem, where semantic categories come across in a continual fashion. Therefore, we are required to address the widely known catastrophic forgetting phenomenon [166], aiming at preserving knowledge from past classes when learning new ones. However, unlike standard CIL, knowledge preservation has to be performed differently depending on the domain in which it is applied:

(i) in *past domains* straightforward recollection of previously observed classes can be imposed, as those classes were learned over past domain distributions,

(ii) whereas, in the *novel domain* recalled memory of past classes should be adapted to account for the semantic shift happening within the input space.

We break down the domain shift and class continual learning problems into simpler underlying sub-problems, as indicated above. Our overall learning framework builds upon multiple individual objectives, each focusing on a specific challenge enclosed in the general setup.

Table 10.2: Training objectives: the *n*/*o* superscripts denote the use of new/old domain data, with $\tilde{\cdot}$ implying stylization.

	New Domain	Old Domains
New Classes	$\mathcal{L}_{ce}^{\tilde{n}}$	$\mathcal{L}_{ce}^{\tilde{o}}$
Old Classes	$\mathcal{L}_{kd}^{\tilde{n}}$	$\mathcal{L}_{kd}^{\tilde{o}}$

We simultaneously progress along class and domain incremental directions; at each learning step, after the first one, both classes and domains experienced so far can be arranged into *new* or *old* types, according to whether they are currently available or not. More in detail, we propose a specific learning objective for each of the different combinations of domain and class types (see Table 10.2 and Figure 10.4), *i.e.*, to:

(i) learn new classes on new domains (Section 10.4.1);

(ii) learn new classes on old domains (Section 10.4.2);

(iii) adapt old-class information to new domains (Section 10.4.3);

(iv) preserve old-class information in old domains (Section 10.4.4).

10.3.1 Domain Stylization

We resort to a style transfer mechanism to recreate image data with statistical properties resembling those of past domains (Figure 10.3). More specifically, starting from the available image data originating from the input domain accessible at the current step, we transfer the styles extracted from all the previously encountered domains. By doing so, a stylized version of each of the former domains is produced, with image content derived from the novel dataset.

The benefits that originate from domain stylization are manifold. (i) We force the prediction model to experience past input distributions under supervision or pseudo-supervision, tackling domain-level catastrophic forgetting. (ii) We aim at learning new classes on old domains, where supervision was not available when they were directly observed. At the same time, we propose to preserve old-class knowledge on old domains, counteracting class-level catastrophic forgetting. (iii) By encountering a variegate input distribution, the predictor is encouraged to develop the ability to generalize to unseen domains, which is crucial in a continual learning paradigm that involves domain shift.

The style transfer mechanism we adopt is inspired by [30] and involves low computational cost and memory requirements. The original algorithm works in the Fourier transform domain: the low frequency portion of the amplitude of the spectral representation from a target

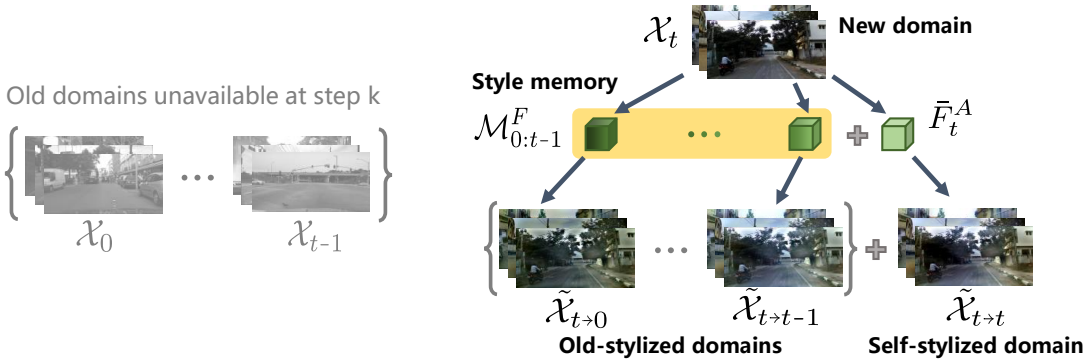


Figure 10.3: Domain stylization: to access no longer available old domain data, we stylize new domain data according to former-domain styles stored in a memory bank. In addition, we perform self-stylization to improve generalization performance.

image (*i.e.*, the style) is extracted and applied to replace that of a source image (*i.e.*, the content), whose phase component is kept unchanged. The outcome is image data with source semantic information, and target-like low-level appearance.

We enhance the original method to accommodate for the further complexity brought in by the class and domain incremental setting. From each image of the currently available dataset, we extract its style tensor (*i.e.*, the amplitude central window), and we average it over all the samples:

$$\bar{F}^A_t = \frac{1}{|\mathcal{T}_t|} \sum_{\mathbf{X}\in\mathcal{T}_t} \mathcal{F}^A(\mathbf{X})[W_\beta], \tag{10.3}$$

where $\mathcal{F}^A(\mathbf{X})$ is the amplitude obtained by the FFT applied to image $\mathbf{X}$, and W_β is the style window. By doing so, we are extracting significant knowledge of *domain-dependent* statistical properties, condensed in a compact representation. The domain-specific style $\bar{F}^A_t$ of step t is stored in an incrementally-filled memory bank $\mathcal{M}^F_{0:t-1} = \{\bar{F}^A_k \,|\, k < t\}$ and preserved across steps. By leveraging the proposed storage mechanism, at each incremental step we can access crucial information of past domain low-level properties (yet minimal if compared to that contained in whole training sets), without requiring direct access to raw image data, which would violate the exemplar-free assumption. We stress that domain shift affects low-level details, while high-level semantic content is mostly shared across domains (*e.g.*, the road serves the same purpose regardless of the dataset, while its appearance in terms of texture or pavement material might vary considerably). To create an oldly-stylized dataset at step t looking back at step $k<t$ (*i.e.*, $\tilde{\mathcal{X}}^t_k$), for each image of the current domain we replace its amplitude window with that of the selected former domain as follows:

$$\tilde{\mathcal{X}}_{t\to k} = \{\mathcal{F}^{-1}([\bar{F}^A_k + \mathcal{F}^A(\mathbf{X})[W^c_\beta], \mathcal{F}^P(\mathbf{X})]) \,|\, \mathbf{X}\in\mathcal{X}_t\}, \tag{10.4}$$

where $\mathcal{F}^{-1}$ is the inverse FFT operator and $\mathcal{F}^P(\mathbf{X})$ is the Fourier phase component of $\mathbf{X}$. In addition, we devise a self-stylization mechanism by self-applying domain style to improve generalization toward future steps, promoting forward transfer. As for the dimension of the style window, we experimentally found that the β parameter as defined in [30] (*i.e.*, the parameter controlling the window size) provides satisfactory and robust results when set to $1e{-}2$. Finally, we stress that our approach is independent of the style transfer technique used, provided that style information and content can be extracted in two distinct steps.

10.4 Learning Across Tasks and Domains

10.4.1 Learning New Classes over New Domains

In the proposed class and domain continual learning framework, direct supervision comes uniquely for the newly introduced class set $\mathcal{C}_t$ and image domain $\mathcal{X}_t$ in the form of the training dataset $\mathcal{T}_t \subset \mathcal{X}_t \times \mathcal{Y}_t$. As mentioned before, image pixels not belonging to $\mathcal{C}_t$, *i.e.*, of past or never seen classes, are assigned to a special class *unknown*, whose semantic statistical properties are highly dynamic.

To account for the semantic shift suffered by the *unknown* class at the current step $t > 0$ w.r.t. previous steps, we group the past and unknown class probability channels as follows:

$$\mathring{P}_t(\mathbf{X})[x, y, c] = \begin{cases} P_t(\mathbf{X})[x, y, c], & \text{if } c \neq u \\ \sum_{c' \in \mathcal{C}_{0:t-1}} P_t(\mathbf{X})[x, y, c'], & \text{if } c = u \end{cases} \tag{10.5}$$

where $P_t(\mathbf{X}) \in \mathbb{R}^{H \times W \times |\mathcal{C}_{0:t}|}$ is the output of M_t prior to the $\arg\max$ when a generic image $\mathbf{X} \in \mathcal{X}$ is given as input. We additionally define $\tilde{\mathcal{T}}_{t \to t} \subset \tilde{\mathcal{X}}_{t \to t} \times \mathcal{Y}_t$ as the *self-stylized* training dataset at step t, where the average style (defined above in Section 10.3.1) of the current image domain has been applied on top of the $\mathcal{X}_t$ domain itself.

To learn the newly introduced classes over the new domain we optimize:

$$\mathcal{L}_{ce}^{\tilde{n}}(\mathcal{C}_t, \mathcal{X}_t) = -\frac{1}{|\tilde{\mathcal{T}}_{t \to t}|} \sum_{\tilde{\mathbf{X}}, \mathbf{Y} \in \tilde{\mathcal{T}}_{t \to t}} \mathbf{Y} \cdot \log \mathring{P}_t(\tilde{\mathbf{X}}), \tag{10.6}$$

where we leverage input data with current style and supervision over the new class set. The $\tilde{n}$ superscript indicates the use of self-stylized data on the *new* domain. The purpose of self-stylization is twofold: first, it provides additional robustness and generalization capability to the prediction model, since input data is supplied with more homogeneous low-level statistic across individual samples. Second, it forces the prediction model to experience domain

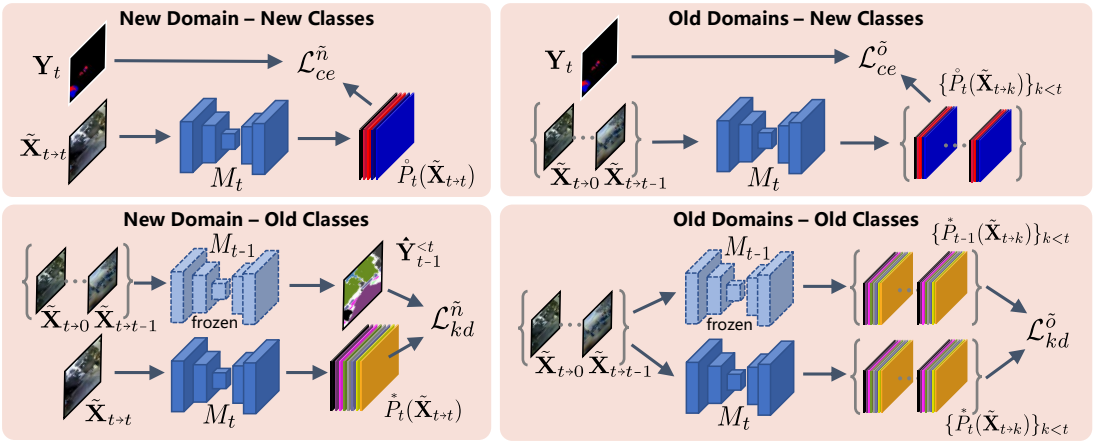


Figure 10.4: Model architecture: we decompose class and domain IL into simpler sub-problems, each addressed by a suitable objective; we learn new classes on new domains (top-left panel), we learn new classes on old domains (top-right panel), we adapt old classes to new domains (bottom-left panel) and preserve old classes on old domains (bottom-right panel).

statistics that will be stored and replayed in the future, acting as proxies for the no longer available previous domain statistics.

10.4.2 Learning New Classes over Past Domains

To compensate for the lack of available input data for former domains, we generate proxy datasets retaining low-level statistics resembling those of past domains. More precisely, for each style $\bar{F}_k^A \in \mathcal{M}_{0:t}$ of step $k < t$ we build $\tilde{\mathcal{T}}_{t\to k} \subset \tilde{\mathcal{X}}_{t\to k} \times \mathcal{Y}_t$ (as detailed in Section 10.3.1), *i.e.*, an *oldly-stylized* training dataset at step t, for which domain-specific visual attributes of step $k < t$ has been applied on domain $\mathcal{X}_t$. Supervision on the newly introduced classes over the old domains is exploited by optimizing:

$$\mathcal{L}_{ce}^{\tilde{o}}(\mathcal{C}_t, \mathcal{X}_{0:t\text{-}1}) = -\frac{1}{t}\sum_{k=0}^{t\text{-}1} \frac{1}{|\tilde{\mathcal{T}}_{t\to k}|} \sum_{\tilde{\mathbf{X}},\mathbf{Y}\in\tilde{\mathcal{T}}_{t\to k}} \mathbf{Y} \cdot \log \mathring{P}_t(\tilde{\mathbf{X}}), \tag{10.7}$$

where we leverage input data with past styles (*i.e.*, with distributions supposedly close* to those of no longer available former domains) and the supervision over the new class set. The superscript $\tilde{o}$ indicates the use of *oldly*-stylized data.

*The *closeness* depends on what the style transfer mechanism is able to transfer in terms of statistical properties. The distribution gap is reduced in terms of low-level properties, while semantic high-level distribution should already be similar across domains.

By concurrently learning the segmentation task at the present step over an augmented pool of input data distributions from the past, the prediction model should learn more general and shareable clues, overcoming the domain shift inherent in the domain continual learning paradigm.

10.4.3 Adapting Old Classes to New Domains

In the addressed class incremental learning scenario, at each new learning step all past class sets are assumed to lack any direct supervision. To recall previously acquired knowledge, we resort to the well-known knowledge distillation objective [165]. Yet, differently from the standard class incremental learning problem as traditionally formalized in the literature [112], we expect to encounter additional challenges:
(i) the input data of past domains (*i.e.*, experienced by the segmentation model when previous class sets were learned) are no longer available;
(ii) a distribution shift separates the current image data to that available at former steps. Thus, we no longer have access to data distributed as that experienced by the segmentation model saved from the past step, which, in principle, should be leveraged to distill knowledge of old classes.

To replicate the image distribution of data of past steps, we resort to the stylization mechanism (Section 10.3.1). Specifically, for each old domain $\mathcal{X}_k$, $k<t$, we build an oldly-stylized dataset $\mathcal{T}_{t\to k}$ starting from that of the current step t.

To access a form of supervision over the past classes we make use of pseudo-labeling via the prediction model from the previous step, which should retain profitable knowledge on the semantic categories learned so far. However, said model might not distill knowledge effectively when fed with input data of an unseen distribution, *i.e.*, originating from the newly introduced domain. Therefore, we exploit oldly-stylized data to enhance pseudo-labeling by mitigating domain shift. We denote with $P^k_{t-1}(\tilde{\mathbf{X}}) \subset \mathbb{R}^{H\times W\times|\mathcal{C}_{t-1}|}$, $\tilde{\mathbf{X}} \in \mathcal{X}_{t\to k}$, the classification probability map from model M_{t-1} over new domain images with the style of step k. We then compute pseudo-labels following:

$$\hat{\mathbf{Y}}^{\mathcal{K}}_{t-1}[x,y] = \arg\max_{c\in\mathcal{C}_{0:t-1}} \max_{k\in\mathcal{K}} P^k_{t-1}(\tilde{\mathbf{X}})[x,y], \tag{10.8}$$

where we leverage old model predictions over past styles, *i.e.*, we set $\mathcal{K}=\{0,\dots,t-1\}$, while $\max_{k\in\mathcal{K}} P^k_{t-1}(\tilde{\mathbf{X}})[x,y]$ indicates that for each spatial location (x,y) we take the probability vector associated to the style with maximum peak value. We then refine the generated pseudo-labels at each spatial location (we will shorten $\hat{\mathbf{Y}}^{\mathcal{K}=\{0,\dots,t-1\}}_{t-1}$ as $\hat{\mathbf{Y}}^{<t}_{t-1}$ and drop the term $[x,y]$

for ease of notation) as:

$$\hat{\mathbf{Y}}^{<t}_{t\text{-}1} = \begin{cases} \hat{\mathbf{Y}}^{<t}_{t\text{-}1}, & \text{if } \hat{\mathbf{Y}}^{<t}_{t\text{-}1} \text{ confident } \wedge\, \mathbf{Y}_t = u \\ u, & \text{if } \mathbf{Y}_t \neq u \\ \text{ignore}, & \text{elsewhere} \end{cases} \tag{10.9}$$

where $\mathbf{Y}_t \in \mathcal{Y}_t$. The hard pseudo-label $\hat{\mathbf{Y}}^{<t}_{t\text{-}1}[x, y]$ (*i.e.*, after the $\arg\max$ operation in Eq. (10.8)) is considered to provide a confident prediction if the peak probability value (of the probability map prior to the $\arg\max$) is bigger than a threshold τ, or if that value is among the top-K fraction of highest peaks for class $c = \hat{\mathbf{Y}}_{t\text{-}1}[x, y]$. We set $\tau = 0.9$ and $K = 0.66$ as advised in [30]. In addition, we leverage the ground-truth supervision on new classes to correct noisy estimations in pseudo-labels, by marking as *unknown* (*i.e.*, *u*) all the pixels of newly introduced categories. We remark that the employed knowledge distillation is designed to provide insight on previous tasks (where current new classes were assigned to the *u* class), whereas we entrust Eq. (10.6) to instill understanding of the novel task. We experimentally verify that using separate objectives to train on new and old classes leads to improved results, as it forces the model to learn to better discriminate between different incremental class sets, part of which might coexist under the same *unknown* group for one or more learning steps. This is especially true for autonomous driving datasets, where each image can contain several semantically diverse elements, for all of which we may not have supervision from the start of the training.

To infuse adapted information about past classes at the current step without direct access to ground-truth information, we resort to the following objective:

$$\mathcal{L}^{\tilde{n}}_{kd}(\mathcal{C}_{0:t\text{-}1}, \mathcal{X}_t) = -\frac{1}{|\mathcal{T}_t|} \sum_{\tilde{\mathbf{X}} \in \mathcal{T}_t} \hat{\mathbf{Y}}^{<t}_{t\text{-}1} \cdot \log \overset{*}{P}_t(\tilde{\mathbf{X}}), \tag{10.10}$$

by which we distill knowledge of past tasks (*i.e.*, recognition of classes in $\mathcal{C}_{0:t\text{-}1}$) over the new domain $\mathcal{X}_t$ via the pseudo-labels derived from the old model S_t.

To account for the semantic shift suffered by the *unknown* class of step $t-1$ when moving to a new step $t > 0$, we group *new* and *unknown* class probability channels as follows:

$$\overset{*}{P}_t(\mathbf{X})[x, y, c] = \begin{cases} P_t(\mathbf{X})[x, y, c], & \text{if } c \neq u \\ \sum_{c' \in \mathcal{C}_t} P_t(\mathbf{X})[x, y, c'], & \text{if } c = u \end{cases} \tag{10.11}$$

where $\overset{*}{P}_t(\mathbf{X}) \in \mathbb{R}^{H \times W \times |\mathcal{C}_{0:t\text{-}1}|}$. We opt for the use of hard-labels in place of the more common soft-labels in the distillation-like loss in order to prevent enforcing an uncertain behav-

ior to S_t. This behaviour could be originated by the mismatch between training and inference input distribution undergone by the old model S_{t-1}, which has been trained over past domains and now is fed with new domain data (the oldly-stylizing operation reduces domain shift but has no guarantees on its complete removal). In Section 10.6.4 we will provide an experimental study on pseudo-labeling.

10.4.4 Preserving Old Classes on Old Domains

In Section 10.4.3 we focused on distilling old-task knowledge on the current novel domain. Nonetheless, our ultimate target is to end up with a segmentation network capable to recognize all the observed classes over all the experienced domains, that is a prediction model robust to both domain and label distribution shifts. For this reason, at every novel incremental step it is required to preserve the task knowledge acquired in the past, that is, on past classes over past domains. To do so, we leverage the output-level knowledge distillation objective in its standard formulation [165], where we force a student model (*i.e.*, the current model) to mimic the predicted classification probability distribution of a teacher model (*i.e.*, the model saved and kept frozen since the end of the previous step). We opted for the objective in its standard fashion [165], as both image and label distributions ideally originate from previous steps, so no domain shift should, in principle, affect the distillation process. In practice, we can not access former incremental datasets. Therefore, to retrieve the missing old-domain data, we resort once more to stylization (Section 10.3.1), so that we can leverage oldly-stylized data as proxy for the missing original images. The final objective is of the following form:

$$\mathcal{L}_{kd}^{\tilde{o}}(\mathcal{C}_{0:t-1}, \mathcal{X}_{0:t-1}) = -\frac{1}{t}\sum_{k=0}^{t-1}\frac{1}{|\tilde{\mathcal{T}}_{t\to k}|}\sum_{\tilde{\mathbf{X}}\in\tilde{\mathcal{T}}_{t\to k}} P_{t-1}(\tilde{\mathbf{X}})\cdot\log \overset{*}{P}_t(\tilde{\mathbf{X}}), \tag{10.12}$$

where $\overset{*}{P}_t(\tilde{\mathbf{X}}) \in \mathbb{R}^{H\times W\times|\mathcal{C}_{0:t-1}|}$ refers to the modified probability distribution of Eq. (10.11), for which *new* and *unknown* categories are incorporated into a single output channel to address the label shift within the u class.

The overall objective is given by:

$$\mathcal{L}_{tot} = \mathcal{L}_{ce}^{\tilde{n}} + \lambda_{ce}^{\tilde{o}}\cdot\mathcal{L}_{ce}^{\tilde{o}} + \lambda_{kd}^{\tilde{n}}\cdot\mathcal{L}_{kd}^{\tilde{n}} + \lambda_{kd}^{\tilde{o}}\cdot\mathcal{L}_{kd}^{\tilde{o}}. \tag{10.13}$$

10.5 Experimental Setup

In this section we provide a detailed description of the experimental setup utilized to validate the proposed framework against multiple competing methods. In Section 10.6 and 10.6.4 we will report the results of the evaluation campaign and extensive ablation studies as additional support.

10.5.1 Incremental Learning Setup

As introduced in Section 9.4, we make use of multiple driving datasets to reproduce the domain and class incremental shift in terms of input distribution, which should characterize the considered general continual learning setting. In particular, we resort to individual datasets, such as the Cityscapes (CS) [14], BDD100K (BDD) [88] and IDD [16], with distinctive visual traits to introduce incremental domain shift by experiencing them in a continual fashion. In addition, we exploit the richer and more diverse Mapillary [17] and Shift [89] benchmarks to simulate input distribution variability by splitting them into uniform sections, according to some criterion (more detail will be provided in the rest of the section). Finally, for the BDD, IDD and Mapillary datasets, only the 19 classes available on the Cityscapes were used. For the Shift dataset, instead, we considered the available 22 semantic categories it offers. In the following, we will detail more specifically how we constructed the domain and class incremental setups used for the experimental evaluation of our method.

Domain Incremental Setup The first domain incremental setup is created by experiencing in succession the CS, BDD and IDD datasets (in different orders) during 3 separate learning steps. Additionally, we propose a further setup, where domain shift across learning steps is achieved by splitting the entire Mapillary dataset into incremental sets based on geographic proximity of samples, *i.e.*, 6 separate data subsets are generated, grouping together pictures taken on the same continent. Finally, we leverage Shift to simulate incrementally variable environmental conditions, by partitioning the whole dataset into 3 groups of samples according to light conditions (*i.e.*, *daytime*, *twilight* and *night*).

Class Incremental Setup We start by following [4] to identify 3 separate groups within the 19 Cityscapes's classes, *i.e.*, (i) *background regions*, (ii) *moving elements*, (iii) *static elements*, which are observed incrementally under various arrangements. Then, we extend the aforementioned 3-way class splitting to Shift in a similar fashion to [4], this time on the 22 classes offered by the synthetic benchmark. All the class incremental sets are detailed in Table 10.3.

Table 10.3: Split of Cityscapes's (CS) and Shift's class sets following the criterion proposed by [4].

		$\mathcal{C}_{bgr}$	$\mathcal{C}_{stat}$	$\mathcal{C}_{mov}$
CS	$\mathcal{C}^0$	{road, sidewalk}	{building, wall, fence}	{person, rider, motorcycle, bicycle}
	$\mathcal{C}^1$	{vegetation, terrain, sky}	{pole, t. light, t. sign}	{car, truck, bus, train}
Shift	$\mathcal{C}^s$	{road line, road, sidewalk, vegetation, sky, ground, water, terrain}	{building, fence, pole, wall t. light, bridge, rail track, guard rail, t. sign, static}	{pedestrian, vehicles, dynamic}

Table 10.4: Class and domain incremental sets.

	Class sets	Domains
Urban	$\{\mathcal{C}_{bgr}, \mathcal{C}_{stat}, \mathcal{C}_{mov}\}$	{CS, BDD, IDD}
Worldwide	$\{\mathcal{C}^0_{bgr}, \mathcal{C}^1_{bgr}, \mathcal{C}^0_{stat}, \mathcal{C}^1_{stat}, \mathcal{C}^0_{mov}, \mathcal{C}^1_{mov}\}$	{EU, NA, AS, OC, AF, SA}
Environmental	$\{\mathcal{C}^s_{bgr}, \mathcal{C}^s_{stat}, \mathcal{C}^s_{mov}\}$	{Daytime, Twilight, Night}

$\mathcal{C}^s$ indicates that the class subset is derived from Shift's original set.

Class and Domain Incremental Setup By merging class and domain individual settings, we devise each class and domain incremental setup reported in Table 10.4. The first (*i.e.*, *urban*) is generated using CS, BDD and IDD datasets, together with the 3-way class split from [4]. Formally, we set the total number of learning steps $T = 3$, and at each step $0 \leq t < T$:

$$\mathcal{D}_t \subset (\mathcal{X}_t, \mathcal{C}_t) \in \{\text{CS}, \text{BDD}, \text{IDD}\} \times \{\mathcal{C}_{bgr}, \mathcal{C}_{stat}, \mathcal{C}_{mov}\}, \tag{10.14}$$

where each dataset and class split is observed once.

We further propose an incremental setup (*i.e.*, *worldwide*) based on continent-wise splitting of the Mapillary dataset. To match the increase in domain set size to 6 elements, we divide each class group [4] in half, for a total of 6 class splits (Table 10.3). We set $T = 6$, and at each step $0 \leq t < T$:

$$\begin{aligned} \mathcal{D}_t \subset (\mathcal{X}_t, \mathcal{C}_t) \in &\{\text{EU}, \text{NA}, \text{AS}, \text{OC}, \text{AF}, \text{SA}\} \times \\ &\{\mathcal{C}^0_{bgr}, \mathcal{C}^1_{bgr}, \mathcal{C}^0_{stat}, \mathcal{C}^1_{stat}, \mathcal{C}^0_{mov}, \mathcal{C}^1_{mov}\}, \end{aligned} \tag{10.15}$$

where each class set and each domain appears only in a single step. Among the large number of possible incremental sequences, we perform the experimental evaluation in the EU $\rightarrow$ NA $\rightarrow$ AS $\rightarrow$ OC $\rightarrow$ AF $\rightarrow$ SA and $\mathcal{C}^0_{bgr} \rightarrow \mathcal{C}^1_{bgr} \rightarrow \mathcal{C}^0_{stat} \rightarrow \mathcal{C}^1_{stat} \rightarrow \mathcal{C}^0_{mov} \rightarrow \mathcal{C}^1_{mov}$ setup.

Finally, the last setup (*i.e.*, *environmental*) combines the environmental partitioning chosen

for Shift with the 3-way class splitting from [4], leading to

$$\mathcal{D}_t \subset (\mathcal{X}_t, \mathcal{C}_t) \in \{\text{Daytime, Twilight, Night}\} \times \{\mathcal{C}^s_{bgr}, \mathcal{C}^s_{stat}, \mathcal{C}^s_{mov}\}. \tag{10.16}$$

for $t < T, T = 3$. Here we opted for the Daytime $\rightarrow$ Twilight $\rightarrow$ Night and $\mathcal{C}^s_{bgr} \rightarrow \mathcal{C}^s_{stat} \rightarrow \mathcal{C}^s_{mov}$ incremental combination for the experimental evaluation.

10.5.2 Implementation Details

We built our framework in PyTorch. Due to the complexity of the investigated problem, in most experiments we use a lightweight segmentation model, *i.e.*, the ErfNet [232]. We argue that a smaller network complies more realistically to deployment-related constraints in real-word applications, *e.g.*, in terms of memory occupation and inference speed. Yet, for comparison purposes we report additional results with the heavier and better performing DeeplabV3 architecture [13] with ResNet101 backbone [100]. In all experiments, the segmentation model is pre-trained on ImageNet [26].

With the ErfNet architecture, we use the Adam optimizer [91] and learning rate set to 5e-4. With the DeeplabV3 architecture, we use the SGD optimizer and learning rate set to 1e-3. Weight decay is fixed to 1e-4, and we employ a polynomial decay of power 0.9 for learning rate scheduling. We train for 100 and 50 epochs at each learning step, with ErfNet and DeeplabV3 respectively (except in Shift, where we set the number of epochs to 10). With ErfNet we use a batch size of 6, with DeeplabV3 we reduce its value to 2 due to GPU memory constraints. When experimentally evaluating on Cityscapes-BDD-IDD and Shift setups, images are resized to 512×1024 resolution. When using Mapillary for training, inputs are first resized to 1024 width (fixed aspect ratio), and then cropped to 512×1024. This pre-processing is done to accommodate for the highly variable aspect ratios of Mapillary's samples. The β parameter controlling the size of the style window is empirically set to 1e-2 and fixed in all experiments. Plus, we experimentally fix $\lambda^{\tilde{o}}_{ce} = \lambda^{\tilde{n}}_{kd} = \lambda^{\tilde{o}}_{kd} = 10$, and keep them unchanged in every incremental setup. This shows that our approach is robust to change of experimental setting, and requires minimal hyper-parameter tuning. We provide ablation studies on performance variation w.r.t. β and loss weights in Section 10.6.4.

10.5.3 Competitors

To the best of our knowledge, this is the first work explicitly modeling and addressing class and domain incremental learning in semantic segmentation. For this reason, we compare with other methods targeting class (CIL) or domain (DIL) incremental learning individually.

Among class-incremental methods, we consider ILT [118] and MiB [3], along with state-of-the-art PLOP [120] and UCD [123]. When using PLOP with ErfNet, we apply the *LocalPOD* loss [120] on embeddings extracted at the end of the first and second blocks, as well as at the output of the encoder. For UCD, we modify the contrastive distillation loss so that the maximum number of positives and negatives is set to 3000 each (which are randomly selected among the whole sets as defined in the original work). We perform this adjustment to meet GPU memory limitations. All experiments were performed on a RTX Titan GPU with 24GB of memory. We believe that a fair comparison should involve comparable GPU resources for all the competitors.

On the domain-incremental side, we compare with [228]. Differently from our setup, they assume to have full task supervision on all the domains incrementally encountered. We adapt their framework to a class-incremental setup by replacing the standard cross-entropy loss with the unbiased version from [3], to prevent the background shift from erasing the task-knowledge learned in past steps.

10.5.4 Evaluation metrics

Inspired by [228], to provide a valuable measure of prediction performance across multiple tasks and domains, we resort to a domain average relative performance w.r.t. a fully-supervised *oracle* reference (the smaller the better) defined at any step t as:

$$\bar{\Delta}_t = \underbrace{\frac{1}{t+1}\sum_{k=0}^{t}}_{\text{domain avg}} \underbrace{\frac{A^{\mathcal{C}_{0:t}}_{\mathcal{X}_k|S_t} - A^{\mathcal{C}_{0:t}}_{\mathcal{X}_k|S^*}}{A^{\mathcal{C}_{0:t}}_{\mathcal{X}_k|S^*}}}_{\substack{\Delta^k_t\text{: relative acc. gap w.r.t.} \\ \text{oracle on step-}k\text{ domain}}}, \tag{10.17}$$

where $A^{\mathcal{C}}_{\mathcal{X}|S}$ is the class-average accuracy (we make use of the commonly employed mIoU metric [6]) attained by segmentation network S on domain $\mathcal{X}$ and class set $\mathcal{C}$. S^* is the oracle segmentation model, *i.e.*, trained with full supervision on the entire pool of classes and domains (even classes and domains that will be observed after step t).

We further provide a measure of generalization aptitude (the higher the better), expressed as the accuracy (*i.e.*, in terms of mIoU) achieved over the entire class set observed so far on a novel dataset never experienced before. At step t, the metric follows:

$$\Gamma^{gen}_t = A^{\mathcal{C}_{0:t}}_{\mathcal{X}_{ext}|S_t} = \frac{1}{|\mathcal{C}_{0:t}|}\sum_{c\in\mathcal{C}_{0:t}} A^{c}_{\mathcal{X}_{ext}|S_t}, \tag{10.18}$$

where $\mathcal{X}_{ext}$ is the unseen domain.

Table 10.5: Experimental results on CS $\rightarrow$ BDD $\rightarrow$ IDD domain setup and $\mathcal{C}_{bgr} \rightarrow \mathcal{C}_{stat} \rightarrow \mathcal{C}_{mov}$ class setup. The best values have been highlighted in bold.

	Step 0			Step 1					Step 2						
	CS ($\mathcal{X}_0$)			**BDD** ($\mathcal{X}_1$)		CS ($\mathcal{X}_0$)			**IDD** ($\mathcal{X}_2$)		BDD ($\mathcal{X}_1$)		CS ($\mathcal{X}_0$)		
	$\text{mIoU}_0^0\uparrow$	$\Delta_0^0\downarrow$	$\bar{\Delta}_0\downarrow$	$\text{mIoU}_1^1\uparrow$	$\Delta_1^1\downarrow$	$\text{mIoU}_1^0\uparrow$	$\Delta_1^0\downarrow$	$\bar{\Delta}_1\downarrow$	$\text{mIoU}_2^2\uparrow$	$\Delta_2^2\downarrow$	$\text{mIoU}_2^1\uparrow$	$\Delta_2^1\downarrow$	$\text{mIoU}_2^0\uparrow$	$\Delta_2^0\downarrow$	$\bar{\Delta}_2\downarrow$
FT ($\mathcal{L}_{ce}^n$)	79.67	5.32	5.32	24.38	61.35	18.11	74.06	67.71	26.27	61.48	10.47	81.72	12.10	81.18	74.79
FT† ($\mathcal{L}_{ce}^{\tilde{n}}$)	79.19	5.89	5.89	20.41	67.65	19.08	72.67	70.16	27.12	60.24	11.51	79.91	13.68	78.72	72.95
MDIL [228]	80.35	4.51	**4.51**	26.12	58.59	23.65	66.13	62.36	28.10	58.80	12.46	78.25	13.22	79.44	72.16
ILT [118]	79.67	5.32	5.32	22.21	64.80	44.70	35.99	50.39	26.69	60.87	16.70	70.85	29.76	53.71	61.81
MiB [3]	79.67	5.32	5.32	34.35	45.55	49.24	29.48	37.51	42.58	37.57	26.36	53.98	36.58	43.10	44.88
PLOP [120]	79.67	5.32	5.32	36.78	41.70	50.05	28.32	35.01	43.15	36.73	27.24	52.44	36.84	42.70	43.96
UCD [123]	79.67	5.32	5.32	35.45	43.80	50.38	27.85	35.83	43.19	36.67	27.38	52.19	37.34	41.91	43.59
LwS w/o $\mathcal{L}_{kd}^{\tilde{o}}$	79.19	5.89	5.89	44.41	29.60	50.77	27.29	28.44	50.70	25.66	34.86	39.14	43.04	33.05	32.62
LwS [237]	79.19	5.89	5.89	44.47	29.51	53.31	23.65	**26.58**	51.20	24.93	35.73	37.62	44.17	31.29	**31.28**
Oracle	84.15	-	-	63.08	-	69.82	-	-	68.20	-	57.28	-	64.29	-	-

† indicates the presence of self-stylization.

10.6 Experimental Results

10.6.1 Evaluation on Urban Scenes

The first experimental setup we explore entails incrementally transitioning between urban and suburban areas of different regions around the world. High- and low- level image contents undergo distribution shifts of different extent: although it might be reasonable to assume that the basic semantic structure of road images is invariant to geographic location, scene elements are likely to change appearance significantly when travelling around the world.

Study on Domain Ordering

To reproduce class and domain distribution shifts, we train on the Cityscapes, BDD and IDD datasets in an incremental fashion. The class incremental protocol is instead the one proposed in [4] (*i.e.*, $\mathcal{C}_{bgr} \rightarrow \mathcal{C}_{stat} \rightarrow \mathcal{C}_{mov}$). As detailed in Section 10.5.1, we define a total of 3 learning steps. In Tables 10.5, 10.6 and 10.7 we report experimental results following 3 different dataset orders, so that each dataset is viewed at all the 3 possible learning steps, considering all experiments performed.

We report results in terms of mIoU computed over all classes excluding the *unknown* one, as typically done in the literature. The mIoU is computed for each domain $\mathcal{X}_k$ (*i.e.*, dataset) experienced up to a current step t (*i.e.*, mIoU_t^k, $k \leq t$), $\forall t < T$. In addition, we provide a measure of relative performance w.r.t. a supervised reference, both for individual domains

Table 10.6: Experimental results on BDD $\rightarrow$ IDD $\rightarrow$ CS domain setup and $\mathcal{C}_{bgr} \rightarrow \mathcal{C}_{stat} \rightarrow \mathcal{C}_{mov}$ class setup. The best values have been highlighted in bold.

	Step 0			Step 1					Step 2						
	BDD ($\mathcal{X}_0$)			**IDD** ($\mathcal{X}_1$)		BDD ($\mathcal{X}_0$)			**CS** ($\mathcal{X}_2$)		IDD ($\mathcal{X}_1$)		BDD ($\mathcal{X}_0$)		
	$\text{mIoU}_0^0\uparrow$	$\Delta_0^0\downarrow$	$\bar{\Delta}_0\downarrow$	$\text{mIoU}_1^1\uparrow$	$\Delta_1^1\downarrow$	$\text{mIoU}_1^0\uparrow$	$\Delta_1^0\downarrow$	$\bar{\Delta}_1\downarrow$	$\text{mIoU}_2^2\uparrow$	$\Delta_2^2\downarrow$	$\text{mIoU}_2^1\uparrow$	$\Delta_2^1\downarrow$	$\text{mIoU}_2^0\uparrow$	$\Delta_2^0\downarrow$	$\bar{\Delta}_2\downarrow$
FT ($\mathcal{L}_{ce}^{n}$)	72.22	6.61	6.61	33.37	52.43	20.49	67.52	59.98	23.36	65.60	7.09	89.60	5.45	89.02	81.41
FT† ($\mathcal{L}_{ce}^{\tilde{n}}$)	72.12	6.74	6.74	33.27	52.58	21.12	66.52	59.55	28.52	55.64	15.44	77.36	14.24	75.14	69.38
MDIL [228]	72.44	6.33	**6.33**	26.78	61.83	15.44	75.52	68.68	25.52	60.30	11.61	82.98	10.77	81.20	74.83
ILT [118]	72.22	6.61	6.61	42.10	39.98	43.00	31.84	35.91	33.33	48.15	26.93	60.52	29.68	48.19	52.29
MiB [3]	72.22	6.61	6.61	52.18	25.62	45.28	28.22	26.92	48.22	25.00	33.57	50.77	30.94	45.98	40.58
PLOP [120]	72.22	6.61	6.61	53.15	24.24	44.25	29.85	27.05	47.21	26.56	35.36	48.15	32.02	44.10	39.60
UCD [123]	72.22	6.61	6.61	52.42	25.28	45.20	28.35	26.81	48.40	24.72	32.60	52.19	28.95	49.47	42.13
LwS w/o $\mathcal{L}_{kd}^{\tilde{o}}$	72.12	6.74	6.74	54.34	21.07	41.36	34.44	27.75	52.56	18.24	36.70	46.19	32.33	43.56	36.00
LwS [237]	72.12	6.74	6.74	54.53	20.80	43.98	30.28	**25.54**	52.63	18.14	38.14	44.08	34.03	40.59	**34.27**
Oracle	77.33	-	-	70.16	-	63.08	-	-	68.20	-	57.28	-	64.29	-	-

† indicates the presence of self-stylization.

Δ_t^k, and as a global quantity $\bar{\Delta}_t$ (Eq. (10.17)). The supervised reference, denoted as *Oracle*, corresponds to the joint training over both class sets and domains.

We compare with methods addressing class incremental learning (ILT [118], MiB [3], PLOP [120] and UCD [123]) and with a recent domain incremental method (MDIL [228]). We also include a simple baseline, activating only the task loss on the new classes and new domain (Eq. (10.6)). This approach is usually referred to as *fine-tuning*, as the focus is just posed on learning the new task. Two variants are reported for this baseline, *i.e.*, with or without self-stylization applied on input images, indicated respectively as $\mathcal{L}_{ce}^{n}$ and $\mathcal{L}_{ce}^{\tilde{n}}$. As for our approach, we evaluate its final form (Eq. (10.13)), complete of all the training objectives detailed in Section 10.4, as well as a simpler configuration without the $\mathcal{L}_{kd}^{\tilde{o}}$ loss (Eq. (10.12)).

By inspecting results in Tables 10.5, 10.6 and 10.7, we notice that the performance attained by different methods at the end of the **initial learning step** are comparable. This is due to the similar objectives employed so far, to learn just the first class set ($\mathcal{C}_{bgr}$) on the first domain, regardless of the domain order. We remark that the proposed self-stylization is not detrimental when learning the current task. We will provide some ablation studies on the impact of stylization in Section 10.6.4

When progressing to the **first incremental step**, catastrophic forgetting has to be addressed to retain good performance. We observe that the $\mathcal{L}_{ce}^{n}$ and $\mathcal{L}_{ce}^{\tilde{n}}$ losses alone are not sufficient to achieve satisfactory results, being focused on the new task and providing no constraints to preserve past knowledge. MDIL [228] performs poorly as well, since the proposed dynamic architecture is not suitable to address partial class incremental supervision, which in our setup is present along with domain incremental shift.

Table 10.7: Experimental results on IDD → CS → BDD domain setup and $\mathcal{C}_{bgr} \rightarrow \mathcal{C}_{stat} \rightarrow \mathcal{C}_{mov}$ class setup. The best values have been highlighted in bold.

	Step 0			Step 1					Step 2						
	IDD ($\mathcal{X}_0$)			**CS** ($\mathcal{X}_1$)		IDD ($\mathcal{X}_0$)			**BDD** ($\mathcal{X}_2$)		CS ($\mathcal{X}_1$)		IDD ($\mathcal{X}_0$)		
	$\text{mIoU}_0^0\uparrow$	$\Delta_0^0\downarrow$	$\bar{\Delta}_0\downarrow$	$\text{mIoU}_1^1\uparrow$	$\Delta_1^1\downarrow$	$\text{mIoU}_1^0\uparrow$	$\Delta_1^0\downarrow$	$\bar{\Delta}_1\downarrow$	$\text{mIoU}_2^2\uparrow$	$\Delta_2^2\downarrow$	$\text{mIoU}_2^1\uparrow$	$\Delta_2^1\downarrow$	$\text{mIoU}_2^0\uparrow$	$\Delta_2^0\downarrow$	$\bar{\Delta}_2\downarrow$
FT ($\mathcal{L}_{ce}^n$)	78.80	8.52	**8.52**	9.66	47.37	8.64	93.07	70.22	9.66	83.14	8.64	86.56	7.09	89.60	86.43
FT† ($\mathcal{L}_{ce}^{\tilde{n}}$)	78.78	8.55	8.55	42.11	39.69	19.81	71.76	55.73	14.05	75.47	12.63	80.35	11.01	83.86	79.89
MDIL [228]	78.72	8.62	8.62	34.87	50.06	11.70	83.32	66.69	8.90	84.46	8.22	87.21	6.70	90.18	87.28
ILT [118]	78.80	8.52	**8.52**	44.44	36.35	43.32	38.26	37.30	24.48	57.26	30.00	53.34	27.88	59.12	56.57
MiB [3]	78.80	8.52	**8.52**	56.23	19.47	23.59	66.37	42.92	23.62	58.76	33.24	48.30	20.57	69.84	58.97
PLOP [120]	78.80	8.52	**8.52**	57.05	18.29	24.74	64.74	41.51	24.18	57.79	34.23	46.76	21.42	68.59	57.71
UCD [123]	78.80	8.52	**8.52**	56.29	19.38	26.45	62.29	40.84	24.88	56.57	34.72	45.99	22.35	67.24	56.60
LwS w/o $\mathcal{L}_{kd}^{\tilde{o}}$	78.78	8.55	8.55	59.61	14.63	43.30	38.28	26.45	34.84	39.18	39.11	39.17	36.13	47.03	41.79
LwS [237]	78.78	8.55	8.55	59.26	15.13	43.95	37.35	**26.24**	37.94	33.76	42.10	34.51	36.60	46.34	**38.21**
Oracle	86.14	-	-	69.82	-	70.16	-	-	68.20	-	57.28	-	64.29	-	-

† indicates the presence of self-stylization.

By analyzing class incremental learning methods, we note that they are able to preserve previously acquired knowledge to some extent, while allowing some plasticity for learning the new task. Still, the domain shift between previous and current datasets has a negative impact on the prediction accuracy of the incrementally trained predictor. All the considered CIL methods, in fact, rely on the ability of a segmentation model frozen from the previous step to preserve knowledge of the past. Yet, because of the domain discrepancy between past and new data, this distillation mechanism could introduce unreliable guidance on former tasks, as the frozen model is subject to a shift in the experienced distribution at the input level when fed with new domain data. At the same time, the distribution gap may hinder the transferability of new-class knowledge to old domains, which are no longer available as training data. These drawbacks are revealed by results of Table 10.7 (IDD → CS → BDD): the significant domain shift between the Cityscapes and IDD datasets prevents CIL methods from effectively preserving and learning task-related clues on IDD, which was experienced at step 0. On the contrary, our approach addresses domain shift by leveraging the stylization scheme and applying carefully designed objectives to suitably tackle the general class and domain incremental learning. In particular, the proposed objectives $\mathcal{L}_{ce}^{\tilde{o}}$ (Eq. (10.7)) and $\mathcal{L}_{kd}^{\tilde{o}}$ (Eq. (10.12)) are specifically designed to address the aforementioned problems affecting CIL methods and retain superior accuracy on former domains. As a result, our LwS improves accuracy by more than 17 mIoU points on IDD at step 1 w.r.t. the best competitor (*i.e.*, UCD [123]).

We also remark that, even with alternative domain orders (Tables 10.5 and 10.6), LwS shows the best stability-plasticity trade-off, retaining the best overall accuracy in terms of $\bar{\Delta}_1$. Fur-

Table 10.8: Generalization performance (Γ_t^{gen}) as mIoU on Mapillary's test set ($\mathcal{C}_{bgr} \rightarrow \mathcal{C}_{stat} \rightarrow \mathcal{C}_{mov}$ setup). The best values have been highlighted in bold.

	CS → BDD → IDD			BDD → IDD → CS			IDD → CS → BDD		
	Step 0	Step 1	Step 2	Step 0	Step 1	Step 2	Step 0	Step 1	Step 2
FT ($\mathcal{L}_{ce}^{n}$)	36.27	22.03	13.71	66.74	25.27	6.60	**59.56**	7.81	8.52
FT† ($\mathcal{L}_{ce}^{\tilde{n}}$)	**58.09**	19.83	14.99	**66.83**	25.77	16.34	59.19	23.40	11.97
MDIL	44.60	24.77	16.05	66.36	18.40	11.01	56.41	14.86	8.55
ILT	36.27	26.80	20.69	66.74	41.32	28.97	**59.56**	39.27	27.96
MiB	36.27	37.68	32.36	66.74	45.99	33.01	**59.56**	23.61	24.23
PLOP	36.27	39.62	33.69	66.74	45.45	34.01	**59.56**	25.29	25.04
UCD	36.27	38.46	34.07	66.74	**46.22**	29.92	**59.56**	27.08	25.89
LwS	**58.09**	**46.36**	**40.43**	**66.83**	44.99	**37.33**	59.19	**43.15**	**39.16**
Oracle	83.96	73.77	65.42	83.96	73.77	65.42	83.96	73.77	65.42

† indicates the presence of self-stylization.

thermore, we can see that, for both CS → BDD → IDD and BDD → IDD → CS orders, the addition of the $\mathcal{L}_{kd}^{\tilde{o}}$ objective in our LwS leads to a boost in performance on the past domain, which coincides with the design purpose of the objective.

In the **final learning step**, the struggle to handle the class and domain incremental training is exacerbated for all the competitors. Baselines and MDIL still provide inferior results, with the latter performing even worse than naïve fine-tuning with self-stylization in some setups. As for CIL methods, PLOP [120] and UCD [123] are the best performing. Both combines output and feature level objectives, which prove to be somewhat robust to domain shift. Even so, the simpler MiB [3] approach shows very competitive results, suggesting that strategies taking into account only a class incremental perspective may not be so effective when incremental domain shift is also occurring. Plus, our method in its complete form greatly outperforms all CIL competitors by a large margin regardless of domain order, going from 5% (BDD → IDD → CS) to 12% (CS → BDD → IDD) and even 16% (IDD → CS → BDD) in terms of $\bar{\Delta}_2$ gap.

Furthermore, in Table 10.8 we investigate the generalization performance (*i.e.*, Γ_t^{gen} from Eq. (10.18)) achieved by the considered methods. To do so, we compute the accuracy at each incremental step on the *unseen* Mapillary dataset for the sets of classes observed so far. We notice that simple fine-tuning and MDIL offer poor generalization results, which is expected due to the low accuracy they already provide on datasets directly observed. On the other hand, CIL methods reach more competitive results, even if none of them proves to be superior in all setups. Still, our approach outperforms all competitors, getting significantly closer to the *Oracle* upper-bound (*i.e.*, the supervised training on the entire Mapillary), specially in the IDD → CS → BDD setup. Also, we remark how we get similar generalization

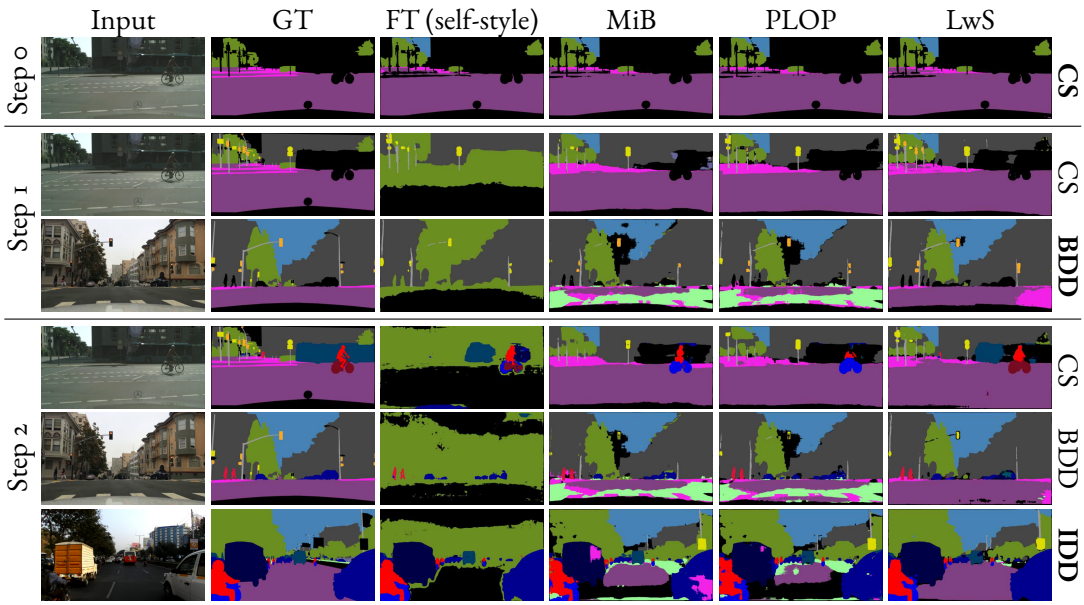


Figure 10.5: Qualitative results on CS → BDD → IDD domain setup and $\mathcal{C}_{bgr} \rightarrow \mathcal{C}_{stat} \rightarrow \mathcal{C}_{mov}$ class setup.

results with different domain incremental orders, demonstrating how our approach is able to learn and preserve generalizable task-related clues regardless of the training environment.

Finally, qualitative results in the form of segmentation maps are provided in Figure 10.5. We stress how the proposed approach yields better **backward** and **forward transfer** throughout the incremental learning. In particular, moving classes like *bicycle* and *bus* appear to be recognized more effectively by our method on the Cityscapes (CS) dataset at the end of the incremental training, even though CS was experienced only along with background-class supervision during the first step. On the other hand, MiB and PLOP fail to provide satisfactory **backward transfer** of those classes to the past CS domain. A similar reasoning can be done regarding the **forward transfer** aptitude. Our approach is able to deliver good segmentation accuracy on the *road* and *sidewalk* background classes even on BDD and IDD datasets, despite them being experienced when $\mathcal{C}_{bgr}$ supervision is no longer available. Contrarily, MiB and PLOP suffer from the domain statistical gap across learning steps, struggling to maintain satisfactory segmentation accuracy of first-step classes by forward transferring knowledge to future steps. Additional analyses will be provided in Section 10.6.4.

Study on Class Ordering

We further investigate the impact of a shift to the class incremental arrangement. Table 10.10 reports experimental results with the CS → BDD → IDD progression, but a modified class order with moving categories $\mathcal{C}_{mov}$ experienced before static ones $\mathcal{C}_{stat}$. We notice a similar

Table 10.10: Experimental results on CS → BDD → IDD domain setup and $\mathcal{C}_{bgr} \rightarrow \mathcal{C}_{mov} \rightarrow \mathcal{C}_{stat}$ class setup. The best values have been highlighted in bold.

CS → BDD → IDD $\mathcal{C}_{bgr} \rightarrow \mathcal{C}_{stat} \rightarrow \mathcal{C}_{mov}$			Method $\mathcal{L}_{ce}^{n}$	$\mathcal{L}_{ce}^{\bar{n}}$	MiB	PLOP	UCD	LwS	Oracle
Step 0	$mIoU_0 \uparrow$	**CS**	79.83	79.39	79.83	79.83	79.83	79.39	84.15
	$\bar{\Delta}_0 \downarrow$		**5.13**	5.65	**5.13**	**5.13**	**5.13**	5.65	-
Step 1	$mIoU_1 \uparrow$	**BDD**	15.79	19.43	26.15	27.69	23.73	40.92	63.08
		CS	14.26	17.22	40.38	42.44	39.76	49.70	69.82
	$\bar{\Delta}_1 \downarrow$		76.26	71.03	48.21	45.40	50.68	**28.99**	-
Step 2	$mIoU_2 \uparrow$	**IDD**	13.47	14.82	31.01	31.89	30.72	43.54	68.20
		BDD	6.94	8.21	23.40	25.21	22.83	32.34	57.28
		CS	7.45	10.49	33.60	33.81	32.85	39.76	64.29
	$\bar{\Delta}_2 \downarrow$		85.52	82.54	53.81	52.21	54.67	**39.29**	-

trend to that observed in Table 10.5 (*i.e.*, same domain order, but different class order), with baselines and MDIL [228] performing poorly, and the improved accuracy achieved by CIL methods still being largely outperformed by the proposed approach.
In addition, we observe that the absolute results are decreased by applying the new class order. The performance of our approach, in fact, drops from 31.28% to 39.29% of $\bar{\Delta}_2$. This discrepancy might be due to class sets observed on domains where it is harder to learn them, and, at the same time, to generalize to the other domains. For instance, we note that IDD provides a lower overall percentage of pixels of $\mathcal{C}_{stat}$ w.r.t. the BDD (11% vs 17%), while for $\mathcal{C}_{mov}$ numbers are similar between them (both around 10% of total pixels). Still, the performance loss is similar for CIL methods, with the gap w.r.t. the best competitor rising from 12 to 13 points of $\bar{\Delta}_2$ (compared to the previous class order).

Study on Model Architecture

We finally evaluate the considered methods when a more capable segmentation network is used, moving from the lightweight ErfNet to the more complex DeeplabV3 with ResNet101 backbone. For comparison purposes, the setup analyzed is again that with CS → BDD → IDD and $\mathcal{C}_{bgr} \rightarrow \mathcal{C}_{stat} \rightarrow \mathcal{C}_{mov}$ orders (Table 10.11). For what concerns our approach, we observe an improved relative performance, raising from 31.28% to 28.53% in terms of $\bar{\Delta}_2$. We emphasize that the $\bar{\Delta}$ measure already takes into account the better oracle results; the accuracy boost, then, shows that our method is able to capitalize the increased capacity offered by the segmentation model.

On the other hand, the CIL competitors are unable to take advantage of the growth in

Table 10.11: Experimental results with **DeeplabV3-ResNet101**. The best values have been highlighted in bold.

CS → BDD → IDD $\mathcal{C}_{bgr} \rightarrow \mathcal{C}_{stat} \rightarrow \mathcal{C}_{mov}$			$\mathcal{L}_{ce}^{n}$	$\mathcal{L}_{ce}^{\bar{n}}$	Method MiB	PLOP	UCD	LwS	Oracle
Step 0	$\text{mIoU}_0\uparrow$	**CS**	78.1	77.13	78.1	78.1	78.1	77.13	84.30
	$\bar{\Delta}_0\downarrow$		**7.35**	8.50	**7.35**	**7.35**	**7.35**	8.50	-
Step 1	$\text{mIoU}_1\uparrow$	**BDD**	31.97	29.44	28.38	29.07	30.84	50.36	64.60
		CS	31.82	55.13	45.10	45.15	45.51	54.75	71.24
	$\bar{\Delta}_1\downarrow$		52.92	56.19	46.38	45.81	44.19	**22.60**	-
Step 2	$\text{mIoU}_2\uparrow$	**IDD**	30.68	30.29	35.93	33.98	38.24	51.92	70.94
		BDD	17.48	17.27	24.18	23.57	26.14	44.98	61.48
		CS	19.46	17.92	33.22	34.38	34.93	47.08	69.17
	$\bar{\Delta}_2\downarrow$		66.73	67.77	53.99	54.68	51.02	**28.53**	-

network capacity, which could indicate a tendency to overfit on the currently observed domain distribution. The best competitor (*i.e.*, UCD), in fact, is significantly outperformed by more than 20% in terms of $\bar{\Delta}$ at both steps 1 and 2. We remark that no additional parameter tuning is performed in this experimental setup concerning method-specific parameters.

10.6.2 Evaluation with Worldwide Data

The second experimental class and domain incremental setup we explore is derived from the Mapillary dataset. Domain shift is once more induced by the variable geographic origin of image samples collected worldwide, *i.e.*, we identify data partitions associated to 6 different continents, corresponding to 6 incremental steps. However, the Mapillary dataset contains variegate data distribution, even considering intra-continent samples, providing a more robust support for training segmentation models. Data richness in turns promotes generalization across steps, in fact lessening the domain gap between different domains. We report experimental results in Table 10.12. In the first steps, we observe similar performance attained by different approaches, that is, when domain shift is small (*e.g.*, between Europe, EU, and North America, NA). Nonetheless, when progressing to the last steps and experiencing increased statistical gap (*e.g.*, when introducing Africa's images, AF), we note that our approach outperforms CIL competitors by a considerable margin, which is of 5 points of $\bar{\Delta}$ w.r.t. the best competitor (PLOP) at the end of incremental training. Also, superior performance in later steps is attained in both new and old domains, confirming the better plasticity-stability trade-off provided by our method. Overall, the improved results LwS reaches w.r.t. state-of-the-art CIL competitors, even when training data is collected to ensure some statistical diversity (as in the experimental setup just considered), further suggests that CIL methods

Table 10.12: Experimental results on the Mapillary dataset. The best values have been highlighted in bold.

EU→NA→AS→OC→AF→SA $\mathcal{C}_{bgr}^{0\to1} \to \mathcal{C}_{stat}^{0\to1} \to \mathcal{C}_{mov}^{0\to1}$			$\mathcal{L}_{ce}^{n}$	$\mathcal{L}_{ce}^{\tilde{n}}$	Method MiB	PLOP	UCD	LwS	Oracle
Step 0	$\text{mIoU}_0\uparrow$	**EU**	73.12	73.07	73.12	73.12	73.12	73.07	79.53
	$\bar{\Delta}_0\downarrow$		**8.06**	8.13	**8.06**	**8.06**	**8.06**	8.13	-
Step 1	$\text{mIoU}_1\uparrow$	**NA**	51.80	51.63	81.28	80.82	81.70	81.85	87.51
		EU	47.67	47.52	76.05	75.76	75.26	74.80	82.34
	$\bar{\Delta}_1\downarrow$		41.46	41.65	**7.38**	7.82	7.62	7.82	-
Step 2	$\text{mIoU}_2\uparrow$	**AS**	25.18	26.09	65.40	65.98	65.70	65.36	74.70
		NA	23.61	23.82	69.28	69.63	68.66	69.77	79.40
		EU	23.98	24.10	66.66	66.86	65.79	65.87	76.62
	$\bar{\Delta}_2\downarrow$		68.42	67.87	12.73	**12.24**	13.23	12.89	-
Step 3	$\text{mIoU}_3\uparrow$	**OC**	16.53	16.74	61.29	60.58	60.53	63.07	76.46
		AS	14.31	14.22	57.95	57.60	57.61	58.13	70.96
		NA	17.10	17.20	62.41	63.29	61.91	64.04	75.77
		EU	14.94	14.94	59.78	60.01	59.34	61.15	72.97
	$\bar{\Delta}_3\downarrow$		78.79	78.72	18.47	18.46	19.15	**16.82**	-
Step 4	$\text{mIoU}_4\uparrow$	**AF**	8.98	7.77	38.48	39.97	40.54	43.93	66.54
		OC	6.03	5.95	40.17	43.52	42.15	47.43	72.30
		AS	7.23	7.31	39.15	41.09	42.03	46.13	69.87
		NA	7.78	7.07	43.10	45.12	45.00	50.07	74.22
		EU	5.45	5.41	38.99	41.52	41.28	46.37	70.22
	$\bar{\Delta}_4\downarrow$		89.91	90.48	43.40	40.20	40.24	**33.77**	-
Step 5	$\text{mIoU}_5\uparrow$	**SA**	9.61	9.18	39.64	41.79	41.48	45.36	64.45
		AF	11.35	9.63	40.76	41.98	41.44	45.25	63.03
		OC	6.78	7.42	36.52	39.08	37.21	41.76	60.82
		AS	8.97	8.17	37.90	40.15	38.76	43.63	64.74
		NA	8.97	9.54	41.40	43.45	43.05	47.08	66.88
		EU	8.18	7.63	38.37	40.53	38.93	43.51	64.04
	$\bar{\Delta}_5\downarrow$		85.98	86.58	38.90	35.67	37.28	**30.57**	-

are likely to be inadequate to deal with distribution shift in the input space.

10.6.3 Evaluation with Variable Environmental Conditions

We evaluate the proposed method when incremental domain shift is due to changing environmental factors, *i.e.*, variable light conditions experienced at different times during the day. In this setting, we employed the Shift synthetic benchmark. We consider the *Daytime* $\to$ *Twilight* $\to$ *Night* domain sequence. Class incremental scheduling follows the $\mathcal{C}_{bgr}$ $\to$

Table 10.13: Experimental results on the Shift dataset. The best values have been highlighted in bold.

Daytime→ Twilight→ Night $\mathcal{C}_{bgr} \rightarrow \mathcal{C}_{stat} \rightarrow \mathcal{C}_{mov}$	**Night**		Twilight		Daytime		
	$\text{mIoU}_2^2 \uparrow$	$\Delta_2^2 \downarrow$	$\text{mIoU}_2^1 \uparrow$	$\Delta_2^1 \downarrow$	$\text{mIoU}_2^0 \uparrow$	$\Delta_2^0 \downarrow$	$\bar{\Delta}_2 \downarrow$
FT ($\mathcal{L}_{ce}^{n}$)	10.54	85.82	9.62	87.21	4.61	94.06	89.03
FT w/ self-style ($\mathcal{L}_{ce}^{\tilde{n}}$)	10.12	86.39	8.50	88.70	7.56	90.26	88.45
MiB	48.07	35.35	52.71	29.92	48.29	37.77	34.34
PLOP	48.58	34.67	53.66	28.66	51.11	34.13	32.48
LwS	60.27	18.94	62.57	16.81	59.78	22.97	**19.57**
Oracle	74.35	-	75.21	-	77.60	-	-

$\mathcal{C}_{stat} \rightarrow \mathcal{C}_{mov}$ arrangement of [4], with the only difference from [4] being that the starting class pool to be split corresponds to the 22 Shift's categories in place of the 19 Cityscape's ones. Results are reported in Table 10.13, where we compare with MiB and PLOP as CIL competitors, along with fine-tuning baselines. We verify the superiority of our approach in jointly handling class and domain incremental training, as we surpass PLOP by 13 points of $\bar{\Delta}_2$. We once more point out the better stability-plasticity balance reached by our method, which achieves improved performance simultaneously over novel and former domains. Overall, results show that the proposed method is effective under domain shifts of different nature. On the other hand, CIL methods prove to be greatly penalized just from the variable scene illumination in different tasks. We argue that in many real-world applications, such as autonomous driving, it is unrealistic to assume that a continual learner will not experience any sort of alteration in input data distribution, making our CL approach much more applicable.

10.6.4 Ablation Studies

In this section, we provide extensive ablation studies to investigate key features of our approach. We will consider the *urban* experimental setup, with CS → BDD → IDD domain and $\mathcal{C}_{bgr} \rightarrow \mathcal{C}_{stat} \rightarrow \mathcal{C}_{mov}$ class orders, unless otherwise stated.

Contribution of Individual Optimization Objectives

We investigate the impact of each of the proposed learning objectives in the overall optimization framework in Table 10.14. Just leveraging the currently available training data by fine-tuning (first two rows) yields unsatisfactory results (even with self-stylization), leading to catastrophic forgetting of class and domain knowledge. Yet, $\mathcal{L}_{ce}^{n}$ (or $\mathcal{L}_{ce}^{\tilde{n}}$) is essential to learn new tasks, so it will be kept in the following analyses to test multi-term objectives.

Table 10.14: Ablation study on the contribution of loss components. The $\mathcal{L}^n_{kd}$ notation here implies that pseudo-labels are generated leveraging new-domain input samples. The best values block-wise have been underlined.

CS → BDD → IDD	**IDD**		BDD		CS		
$\mathcal{C}_{bgr} \rightarrow \mathcal{C}_{stat} \rightarrow \mathcal{C}_{mov}$	$\text{mIoU}^2_2 \uparrow$	$\Delta^2_2 \downarrow$	$\text{mIoU}^1_2 \uparrow$	$\Delta^1_2 \downarrow$	$\text{mIoU}^0_2 \uparrow$	$\Delta^0_2 \downarrow$	$\bar{\Delta}_2 \downarrow$
$\mathcal{L}^n_{ce}$	26.27	61.48	10.47	81.72	12.10	81.18	74.79
$\mathcal{L}^{\tilde{n}}_{ce}$	27.12	60.24	11.51	79.91	13.68	78.72	72.95
$\mathcal{L}^{\tilde{n}}_{ce} + \mathcal{L}^{\tilde{o}}_{ce}$	28.09	58.81	13.32	76.75	16.32	74.61	70.06
$\mathcal{L}^{\tilde{n}}_{ce} + \mathcal{L}^{\tilde{o}}_{kd}$	40.63	40.43	24.95	56.44	34.14	46.90	47.92
$\mathcal{L}^{n}_{ce} + \mathcal{L}^{n}_{kd}$	43.33	36.47	26.62	53.53	37.36	41.89	43.96
$\mathcal{L}^{\tilde{n}}_{ce} + \mathcal{L}^{\tilde{n}}_{kd}$	48.12	29.45	32.40	43.44	40.57	36.89	36.59
$\mathcal{L}^{\tilde{n}}_{ce} + \mathcal{L}^{\tilde{n}}_{kd} + \mathcal{L}^{\tilde{o}}_{kd}$	19.23	71.80	17.68	69.13	24.15	62.44	67.79
$\mathcal{L}^{\tilde{n}}_{ce} + \mathcal{L}^{\tilde{o}}_{ce} + \mathcal{L}^{\tilde{o}}_{kd}$	50.08	26.57	34.16	40.36	42.86	33.33	33.42
$\mathcal{L}^{\tilde{n}}_{ce} + \mathcal{L}^{\tilde{n}}_{kd} + \mathcal{L}^{\tilde{o}}_{ce}$	50.70	25.66	34.86	39.14	43.04	33.05	32.62
$\mathcal{L}^{n}_{ce} + \mathcal{L}^{n}_{kd} + \mathcal{L}^{\tilde{o}}_{ce} + \mathcal{L}^{\tilde{o}}_{kd}$	46.59	31.69	30.51	46.74	40.44	37.10	38.51
$\mathcal{L}^{\tilde{n}}_{ce} + \mathcal{L}^{\tilde{n}}_{kd} + \mathcal{L}^{\tilde{o}}_{ce} + \mathcal{L}^{\tilde{o}}_{kd}$	51.20	24.93	35.73	37.62	44.17	31.29	**31.28**
Oracle	68.20	-	57.28	-	64.29	-	-

By adding a second term in the overall objective (second block of rows) we improve results, especially if the supplemental objective is focused on retaining old-class knowledge. We reach, in fact, the best performance with a 2-term configuration when $\mathcal{L}^{\tilde{n}}_{kd}$ is introduced. This suggests that old-class knowledge preservation is effective even when applied on the new domain, which is directly experienced by means of the available training data. At the same time, the $\mathcal{L}^{\tilde{n}}_{kd}$ objective allows to retain good accuracy w.r.t. past domains, thanks to the improved generalization aptitude promoted by the stylization mechanism, without which (*i.e.*, third row of the block) multiple accuracy points are lost.

When analyzing 3-term objectives (third block of rows), we see noticeable gain with different combinations, except for $\mathcal{L}^{\tilde{n}}_{kd}$ and $\mathcal{L}^{\tilde{o}}_{kd}$ jointly active, where the excessive focus on past-class knowledge preservation generates training instability. In the last row of the block, we clearly see that, by adding the $\mathcal{L}^{\tilde{o}}_{ce}$ loss on top of the best two-term configuration, the incremental learning becomes more robust, with improved final results on all domains.

Finally, we remark that the full framework (last block) yields the best overall performance, with stylization once more playing a substantial role. The overall performance is, in fact, strongly degraded if stylization is turned off, as showed in the second last row.

Pseudo-label Generation

We further analyze the influence exerted by pseudo-labeling. We remark that the proposed enhanced labeling mechanism (described in Section 10.4.3) exploits oldly-stylized images to

Table 10.15: Ablation study on pseudo-labeling schemes. The best values have been highlighted in bold.

CS → BDD → IDD $\mathcal{C}_{bgr} \to \mathcal{C}_{stat} \to \mathcal{C}_{mov}$	**IDD**		BDD		CS		
	$\text{mIoU}_2^2 \uparrow$	$\Delta_2^2 \downarrow$	$\text{mIoU}_2^1 \uparrow$	$\Delta_2^1 \downarrow$	$\text{mIoU}_2^0 \uparrow$	$\Delta_2^0 \downarrow$	$\bar{\Delta}_2 \downarrow$
$\mathcal{L}_{ce}^{n}+\mathcal{L}_{kd,n}^{n}+\mathcal{L}_{ce}^{\tilde{o}}+\mathcal{L}_{kd}^{\tilde{o}}$	46.59	31.69	30.51	46.74	40.44	37.10	38.51
$\mathcal{L}_{ce}^{n}+\mathcal{L}_{kd,o}^{n}+\mathcal{L}_{ce}^{\tilde{o}}+\mathcal{L}_{kd}^{\tilde{o}}$	40.09	41.22	25.55	55.40	34.90	45.71	47.44
$\mathcal{L}_{ce}^{\tilde{n}}+\mathcal{L}_{kd,n}^{\tilde{n}}+\mathcal{L}_{ce}^{\tilde{o}}+\mathcal{L}_{kd}^{\tilde{o}}$	51.11	25.06	34.01	40.63	43.96	31.62	32.44
$\mathcal{L}_{ce}^{\tilde{n}}+\mathcal{L}_{kd,o}^{\tilde{n}}+\mathcal{L}_{ce}^{\tilde{o}}+\mathcal{L}_{kd}^{\tilde{o}}$	51.20	24.93	35.73	37.62	44.17	31.29	**31.28**
Oracle	68.20	-	57.28	-	64.29	-	-

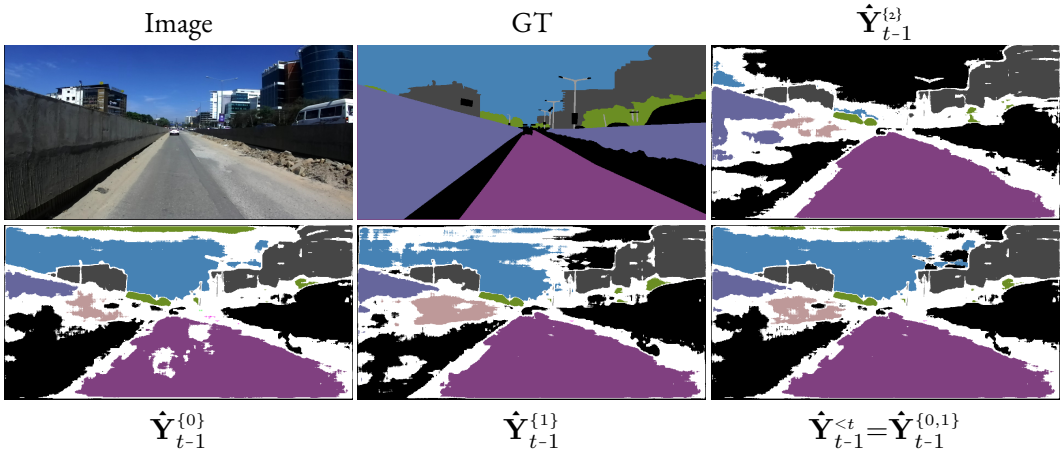


Figure 10.6: Different ways of pseudo-labeling ($t=2$). White regions correspond to the *ignore* label.

mitigate the domain shift endured by the frozen segmentation model distilling knowledge from the past. Results are reported in Table 10.15; the $\mathcal{L}_{kd,\{n,o\}}^{d}$, $d \in \{n, \tilde{n}\}$, notation indicates that pseudo-labels are generated leveraging new-domain ($\mathcal{L}_{kd,n}^{d}$) or oldly-stylized ($\mathcal{L}_{kd,o}^{d}$) input samples.

We notice that when self-stylization is disabled (first two rows) the efficacy of our method is reduced, while the beneficial effect offered by the self-stylizing module can be appreciated in the last two rows. This occurs because self-stylization better prepares the segmentation model for future steps, in which the stylizing mechanism leverages old-domain styles to inject old-domain knowledge into the ongoing learning step. In other words, when self-stylizing images, what will be experienced as an *old* style will have already been experienced as a *new* style before. Therefore, the undesired visual artifacts generated by style transfer are experienced by the network from the very first step in which each domain is introduced. This, in turn, ensures greater robustness over the incremental learning process. Furthermore, in setups with self-stylization, as opposed to what occurs without it, pseudo-labeling performed on top of oldly-stylized images yields the best overall performance, if compared to the same

Table 10.17: Ablation study on stylization ($\beta = 0.01$ corresponds to the default configuration). The best values have been highlighted in bold.

CS → BDD → IDD $\mathcal{C}_{bgr} \rightarrow \mathcal{C}_{stat} \rightarrow \mathcal{C}_{mov}$			No stylization	β 0.001	β 0.01	β 0.1
Step 0	$\text{mIoU}_0\uparrow$	**CS**	79.67	79.8	79.19	78.54
	$\bar{\Delta}_0\downarrow$		5.32	**5.17**	5.89	6.66
Step 1	$\text{mIoU}_1\uparrow$	**BDD**	33.67	35.06	44.47	44.79
		CS	49.20	43.75	53.31	50.45
	$\bar{\Delta}_1\downarrow$		38.08	40.88	**26.58**	28.37
Step 2	$\text{mIoU}_2\uparrow$	**IDD**	43.33	48.60	51.20	50.03
		BDD	26.62	27.77	35.73	34.84
		CS	37.36	37.61	44.17	43.01
	$\bar{\Delta}_2\downarrow$		43.96	40.59	**31.28**	32.97

labeling process executed over image samples with new-domain style. This happens because the network (frozen from the past step) used to generate pseudo-labels is better equipped to face input distributions of previously experienced old domains, while, instead, it may suffer from domain shift when presented with new unseen input distributions.

In Figure 10.6 we report pseudo-labels generated according to different criteria, to provide visual confirmation of the improved pseudo-supervision achieved on top of the oldly stylization. The considered setup involves CS $\rightarrow$ BDD $\rightarrow$ IDD and $\mathcal{C}_{bgr} \rightarrow \mathcal{C}_{stat} \rightarrow \mathcal{C}_{mov}$ progressions, and maps are retrieved at the last step (*i.e.*, $t = 2$). We observe that the segmentation model taken from step $t-1$ (*i.e.*, second last step) is not detecting the sky region of the new-domain image, *i.e.*, $\hat{\mathbf{Y}}^{\{2\}}_{t\text{-}1}$ provides unreliable supervision by labeling the top portion of the picture as *unknown* (when the true *sky* class is among those already seen). On the other hand, when leveraging oldly-stylized images to generate pseudo-supervision ($\hat{\mathbf{Y}}^{<t}_{t\text{-}1}$), more reliable old-domain guidance ($\hat{\mathbf{Y}}^{\{0\}}_{t\text{-}1}$ and $\hat{\mathbf{Y}}^{\{1\}}_{t\text{-}1}$) is exploited, with individual positive contributions successfully merged in the final map (*e.g.*, in *sky* and *road* regions). Thus, we end up with $\hat{\mathbf{Y}}^{<t}_{t\text{-}1}$ being more accurate than each domain-specific alternative $\hat{\mathbf{Y}}^{\{k\}}_{t\text{-}1},\ k \leq t$.

Degree of Stylization

We propose to further study on the stylization mechanism. Table 10.17 shows the results of our method (complete with all objectives) under different degrees of stylization, which are determined by the β parameter (see Section 10.3.1). We notice that disabling stylization or operating it in a more conservative manner (*i.e.*, with $\beta = 0.001$) yields low results, with the latter configuration still outperforming the no stylization approach, as the statistical proper-

Figure 10.7: Domain-knowledge transfer (mIoU↑ (%)).

ties captured and transferred are not sufficient to successfully retain old-domain information. On the other hand, if the stylization is raised to an excessive extent (*i.e.*, with $\beta = 0.1$), we observe performance degradation on the overall $\bar{\Delta}_2$ score. In this scenario, artifacts are more likely to be introduced on oldly-stylized images, thus hindering the segmentation task.

Knowledge Transfer Across Tasks and Domains

We propose ablation studies to evaluate the knowledge transfer aptitude of our method, both under task and domain perspectives. Figure 10.7 presents a comparative of multiple CIL competitors in terms of predisposition towards *domain-knowledge transfer;* we report the mIoU achieved on individual domains only on classes experienced so far across multiple steps in matrix form. We consider multiple incremental setups, with urban datasets and variable domain order. We observe that our approach, right from the first learning step, achieves better *forward transfer* to future domains, as indicated by per-domain mIoU values in the top triangular sections, regardless of the setup considered. At the same time, this translates into superior performance on current domains (represented by diagonal mIoU values), as they benefit from a better forward-adaptability acquired before. Plus, improved *backward transfer* to former domains is testified by higher mIoU values in the bottom triangular part of matrices.

To provide an insight on *task-knowledge transfer* proneness of different incremental meth-

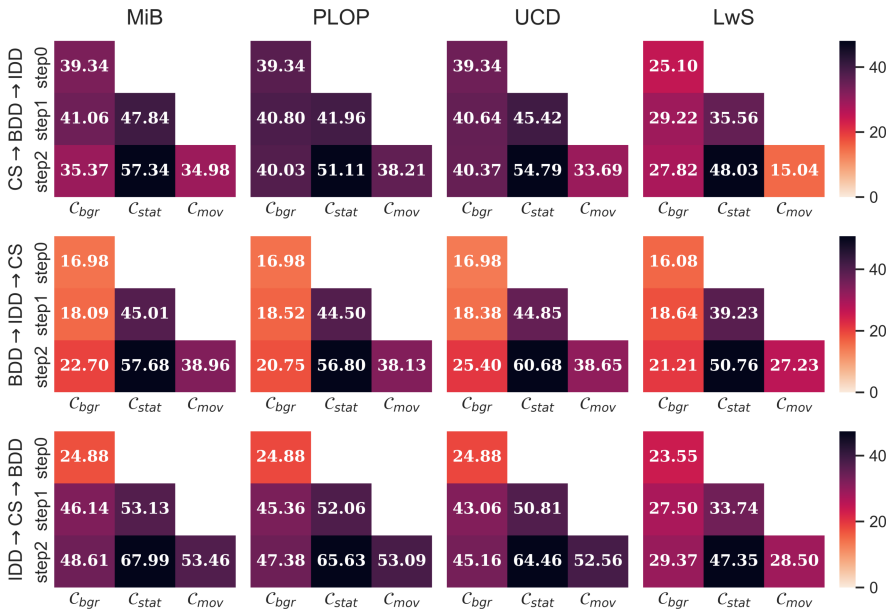


Figure 10.8: Task-knowledge transfer ($\bar{\Delta}\downarrow$).

ods, in Figure 10.8 we report a comparative in terms of $\bar{\Delta}$ results at multiple learning steps; values are computed on *single* incremental sets of classes and represent an average score across all domains (both experienced and future ones).The experimental setups are the same considered when studying domain transfer, and results are arranged in matrix form. We observe that our $\bar{\Delta}$ scores in the bottom triangular part of matrices are lower than competitors, suggesting that our method yields better *backward transfer* in terms of task knowledge. At the same time, the smaller $\bar{\Delta}$ diagonal elements indicate improved performance on current tasks, confirming the better stability-plasticity compromise offered by our approach.

11 Conclusion

Deep learning has recently witnessed a rapid boost in popularity in the computer vision field, allowing to reach unprecedented efficacy in multiple fundamental problems. Yet, while deep neural networks have been shown to shine in domain-specific static learning contexts, real-world applicability of deep learning based algorithms requires them to operate in ever-changing environments, addressing prediction tasks evolving over time. Starting from such premises, in this thesis we investigated the aptitude of deep learning models to generalize acquired knowledge across domains and tasks. Firstly, we explore the *domain* and the *task* knowledge transferability as separate individual problems. Domain adaptability is analyzed in the setting of Unsupervised Domain Adaptation, aiming at extending the information captured for a static task from a label-abundant source domain to a label-deficient target one. Continual task learning is, instead, examined as Class Incremental Learning, with the continual deep learner undergoing a dynamic expansion of the class pool characterizing the prediction task, while still operating on a static input data distribution. Next, we fuse the aforementioned transfer learning problems to undertake a more general continual learning under the incremental distribution shift of both input and label spaces.

In the first part of the dissertation, we focused on the Unsupervised Domain Adaptation for semantic image segmentation. We presented an in-depth overview of several research directions that have been undertaken to tackle this problem. At a high level, we differentiate adaptation methods according to the representation depth at which the adaptation problem is addressed. On the one hand, it is possible to focus directly on the varying visual properties of image data across input domain spaces; on the other hand, adaptation mechanisms aimed at feature representations or output predictions have proven to be very effective. We, therefore, propose to investigate knowledge transferability across diverse representation lev-

els within the segmentation pipeline. We started with input-level adaptation, developing an image-to-image translation framework to bridge low-level statistics of source and target image data. This is achieved by matching the visual appearance of image samples between domains and generating target-like pseudo-supervision. Then, we moved to output-space adaptation, devising a domain-adversarial mechanism, based on a pair of discriminative modules, and a self-training scheme to enforce statistical homogeneity of prediction maps from different domains. Finally, we targeted the adaptability of latent representations via a class-conditional domain alignment, built upon optimization modules that enforce feature clustering and class separability, driven by several regularization objectives.

In the second part of the thesis, we posed our attention on the continual learning problem under a task perspective. We paired the incremental learning with classification tasks, both at coarse (whole-image classification) and fine (semantic segmentation) level.

First off, we identified the availability of replay data from former learning steps as playing a crucial role in maintaining a satisfactory prediction performance throughout long-lasting incremental progressions. Therefore, we built a feature-level generative framework to reiterate former-task knowledge. However, due to the lack of training data from past steps, old-task feature distribution was evolving untracked. Hence, we recognized the representation drift as a primary source of catastrophic forgetting, and propose to model feature and semantic drifts, which are jointly leveraged to gain access to reliable and up-to-date estimates of the feature distribution for former tasks. In a second work, we changed perspective by directing our focus over the input image space, in order to replay input data. The inability to store and replay actual training samples (since we complied with the continual learning settings in their stricter exemplar-free configuration) prompts us to find an alternative solution to the problem. Thus, we elaborated a data retrieval scheme, based on a generative network and web-crawling to obtain coarsely-labeled image samples, joined by a label evaluation framework to provide reliable pseudo-labeling to the retrieved replay images. Label inpainting was further proposed to address the background semantic shift phenomenon and enhance ground-truth labeling. Similarly to the feature-space replay strategy, input-level reiteration of artificially generated samples from past steps allowed to achieve remarkable robustness when learning incrementally through several phases.

In the third part of this work, we extended our research endeavors to face a comprehensive continual learning setup that comprises both evolving domains and tasks. Forward and backward transfer along both domain and task directions have proven to be fundamental to promote generalization aptitude and retain robust performance across learning steps. We showed that methods individually addressing domain or task shifts are ill-equipped to tackle the general continual learning problem, lacking robustness to partial task supervision (in case of sole domain incremental focus) or suffering from domain shift (in case of class incremen-

tal methods). We proposed to extend task-knowledge across domains by domain-stylizing images to recover and reiterate domain-specific visual attributes from former steps, while a robust distillation framework allowed to retain and adapt task information to novel domain distributions.

As future work, we would like to further investigate task and domain shifts in more intertwined and general setups, aiming at complying to realistic constraints typically faced in real-world applications. Data is usually experienced in a continuous flow, with domain shifts happening in modest entity but repeated steps. Thus, it is of primary importance to develop lightweight, robust and flexible frameworks to cope with this more realistic continuous experiencing of an ever-changing surrounding environment.

At the same time, we could further reduce the constraints on the available domain and task supervision, therefore mimicking an agent exploring a new environment. We could, for instance, assume few-shot, or even zero-shot, learning settings, with very limited supervision over the newly encountered domains. These settings could be tackled under an active learning perspective, where valuable but limited ground-truth information could be provided on-the-fly in an online manner to improve robustness when coupled with fast-converging training procedures.

Finally, it would be interesting to address the task-incremental learning in a more general fashion. One direction might be to allow more flexibility in the class-increment paradigm, that is, reducing the constraints on the distribution of incrementally experienced tasks. Differently, we could broaden the notion of task continual learning, where tasks could be intended as not simply limited to solving coarse or dense classification on a set of categories, but they could be understood in a more general sense, where input and/or output spaces can significantly vary, along with the underlying relationship we seek to model.

References

[1] M. Chen, H. Xue, and D. Cai, "Domain adaptation for semantic segmentation with maximum squares loss," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[2] U. Michieli and P. Zanuttigh, "Continual semantic segmentation via repulsion-attraction of sparse and disentangled latent representations," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[3] F. Cermelli, M. Mancini, S. R. Bulò, E. Ricci, and B. Caputo, "Modeling the background for incremental learning in semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[4] M. Klingner, A. Bär, P. Donn, and T. Fingscheidt, "Class-incremental learning for semantic segmentation re-using neither old data nor old labels," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2020.

[5] R. Szeliski, *Computer vision: algorithms and applications*. Springer Nature, 2022.

[6] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.

[7] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[8] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440.

[9] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2881–2890.

[10] F. Yu, V. Koltun, and T. A. Funkhouser, "Dilated residual networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 636–644.

[11] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 40, pp. 834–848, 2018.

[12] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 833–851.

[13] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[14] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3213–3223.

[15] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving video database with scalable annotation tooling," *arXiv preprint arXiv:1805.04687*, 2018.

[16] G. Varma, A. Subramanian, A. M. Namboodiri, M. Chandraker, and C. V. Jawahar, "IDD: A dataset for exploring problems of autonomous navigation in unconstrained environments," in *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[17] G. Neuhold, T. Ollmann, S. Rota Bulo, and P. Kontschieder, "The Mapillary vistas dataset for semantic understanding of street scenes," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017, pp. 4990–4999.

[18] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *Proceedings of the European Conference on Computer Vision (ECCV)*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., vol. 9906. Springer International Publishing, 2016, pp. 102–118.

[19] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3234–3243.

[20] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) challenge," *International Journal of Computer Vision (IJCV)*, vol. 88, no. 2, pp. 303–338, 2010.

[21] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 740–755.

[22] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 633–641.

[23] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012.

[24] S. Song, S. P. Lichtenberg, and J. Xiao, "Sun rgb-d: A rgb-d scene understanding benchmark suite," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 567–576.

[25] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[26] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[27] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *Proceedings of the International Conference on Machine Learning*, 2015, pp. 97–105.

[28] K. Saito, D. Kim, S. Sclaroff, and K. Saenko, "Universal domain adaptation through self supervision," in *Neural Information Processing Systems (NeurIPS)*, 2020.

[29] M. Bucher, T.-H. Vu, M. Cord, and P. Pérez, "Buda: Boundless unsupervised domain adaptation in semantic segmentation," *arXiv preprint arXiv:2004.01130*, 2020.

[30] Y. Yang and S. Soatto, "FDA: fourier domain adaptation for semantic segmentation," *arXiv preprint arXiv:2004.05498*, 2020.

[31] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Neural Information Processing Systems (NeurIPS)*, 2014, pp. 2672–2680.

[32] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proceedings of the International Conference on Machine Learning*, 2015, pp. 1180–1189.

[33] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[34] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7167–7176.

[35] J. Hoffman, D. Wang, F. Yu, and T. Darrell, "FCNs in the wild: Pixel-level adversarial and constraint-based adaptation," *arXiv preprint arXiv:1612.02649*, 2016.

[36] Y. Chen, W. Li, and L. Van Gool, "Road: Reality oriented adaptation for semantic segmentation of urban scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7892–7901.

[37] Y. Zhang, Z. Qiu, T. Yao, D. Liu, and T. Mei, "Fully convolutional adaptation networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6810–6818.

[38] Y. Li, L. Yuan, and N. Vasconcelos, "Bidirectional learning for domain adaptation of semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[39] H. Huang, Q. Huang, and P. Krähenbühl, "Domain transfer through deep activation matching," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[40] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell, "Cycada: Cycle-consistent adversarial domain adaptation," in *Proceedings of the International Conference on Machine Learning*, 2018.

[41] Y.-C. Chen, Y.-Y. Lin, M.-H. Yang, and J.-B. Huang, "Crdoco: Pixel-level domain transfer with cross-domain consistency," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[42] Y. Luo, P. Liu, T. Guan, J. Yu, and Y. Yang, "Significance-aware information bottleneck for domain adaptive semantic segmentation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[43] Y.-H. Chen, W.-Y. Chen, Y.-T. Chen, B.-C. Tsai, Y.-C. Frank Wang, and M. Sun, "No more discrimination: Cross city adaptation of road scene segmenters," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017, pp. 1992–2001.

[44] L. Du, J. Tan, H. Yang, J. Feng, X. Xue, Q. Zheng, X. Ye, and X. Zhang, "SSF-DAN: separated semantic feature based domain adaptation network for semantic segmentation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[45] X. Zhu, H. Zhou, C. Yang, J. Shi, and D. Lin, "Penalizing top performers: Conservative loss for semantic segmentation adaptation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 568–583.

[46] Z. Murez, S. Kolouri, D. J. Kriegman, R. Ramamoorthi, and K. Kim, "Image to image translation for domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[47] S. Sankaranarayanan, Y. Balaji, A. Jain, S. Nam Lim, and R. Chellappa, "Learning from synthetic data: Addressing domain shift for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 3752–3761.

[48] Y.-H. Tsai, W.-C. Hung, S. Schulter, K. Sohn, M.-H. Yang, and M. Chandraker, "Learning to adapt structured output space for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7472–7481.

[49] Y. Chen, W. Li, X. Chen, and L. Van Gool, "Learning semantic segmentation from synthetic data: A geometrically guided input-output adaptation approach," *arXiv preprint arXiv:1812.05040*, 2018.

[50] W. Chang, H. Wang, W. Peng, and W. Chiu, "All about structure: Adapting structural information across domains for boosting semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 1900–1909.

[51] Y. Luo, L. Zheng, T. Guan, J. Yu, and Y. Yang, "Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[52] J. Yang, W. An, S. Wang, X. Zhu, C. Yan, and J. Huang, "Label-driven reconstruction for domain adaptation in semantic segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[53] M. Biasetton, U. Michieli, G. Agresti, and P. Zanuttigh, "Unsupervised Domain Adaptation for Semantic Segmentation of Urban Scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

[54] U. Michieli, M. Biasetton, G. Agresti, and P. Zanuttigh, "Adversarial learning and self-teaching techniques for domain adaptation in semantic segmentation," *IEEE Transaction on Intelligent Vehicles*, 2020.

[55] T.-H. Vu, H. Jain, M. Bucher, M. Cord, and P. Pérez, "Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2517–2526.

[56] T. Vu, H. Jain, M. Bucher, M. Cord, and P. Pérez, "DADA: depth-aware domain adaptation in semantic segmentation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019, pp. 7363–7372.

[57] Y.-H. Tsai, K. Sohn, S. Schulter, and M. Chandraker, "Domain adaptation for structured output via discriminative patch representations," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019, pp. 1456–1465.

[58] Q. Zhou, Z. Feng, G. Cheng, X. Tan, J. Shi, and L. Ma, "Uncertainty-aware consistency regularization for cross-domain semantic segmentation," *arXiv preprint arXiv:2004.08878*, 2020.

[59] C. Qin, L. Wang, Y. Zhang, and Y. Fu, "Generatively inferential co-training for unsupervised domain adaptation," in *Proceedings of the International Conference on Computer Vision Workshops (ICCVW)*, 2019, pp. 1055–1064.

[60] P. Li, X. Liang, D. Jia, and E. P. Xing, "Semantic-aware grad-gan for virtual-to-real urban scene adaption," in *Proceedings of British Machine Vision Conference (BMVC)*, 2018.

[61] Y. Yang, D. Lao, G. Sundaramoorthi, and S. Soatto, "Phase consistent ecological domain adaptation," *arXiv preprint arXiv:2004.04923*, 2020.

[62] R. Gong, W. Li, Y. Chen, and L. V. Gool, "DLOW: domain flow for adaptation and generalization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2477–2486.

[63] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[64] K. Lee, G. Ros, J. Li, and A. Gaidon, "SPIGAN: privileged adversarial learning from simulation," in *International Conference on Learning Representations (ICLR)*, 2019.

[65] J. Choi, T. Kim, and C. Kim, "Self-ensembling with gan-based data augmentation for domain adaptation in semantic segmentation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019, pp. 6830–6840.

[66] W. Hong, Z. Wang, M. Yang, and J. Yuan, "Conditional generative adversarial network for structured domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 1335–1344.

[67] F. Pizzati, R. d. Charette, M. Zaccaria, and P. Cerri, "Domain bridge for unpaired image-to-image translation and unsupervised domain adaptation," in *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, 2020, pp. 2990–2998.

[68] X. Huang, M. Liu, S. J. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 179–196.

[69] A. Dundar, M. Liu, T. Wang, J. Zedlewski, and J. Kautz, "Domain stylization: A strong, simple baseline for synthetic to real image domain adaptation," *arXiv preprint arXiv:1807.09384*, 2018.

[70] Z. Wu, X. Wang, J. Gonzalez, T. Goldstein, and L. Davis, "ACE: adapting to changing environments for semantic segmentation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019, pp. 2121–2130.

[71] L. A. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," in *Neural Information Processing Systems (NeurIPS)*, 2015, pp. 262–270.

[72] ——, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2414–2423Multimodal UnsuperviseMultimodal Unsupervise.

[73] X. Huang and S. J. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017, pp. 1510–1519.

[74] K. Saito, Y. Ushiku, T. Harada, and K. Saenko, "Adversarial dropout regularization," in *International Conference on Learning Representations (ICLR)*, 2018.

[75] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada, "Maximum classifier discrepancy for unsupervised domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 3723–3732.

[76] K. Watanabe, K. Saito, Y. Ushiku, and T. Harada, "Multichannel semantic segmentation with unsupervised domain adaptation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[77] S. Lee, D. Kim, N. Kim, and S.-G. Jeong, "Drop to adapt: Learning discriminative features for unsupervised domain adaptation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019, pp. 91–100.

[78] Y. Zou, Z. Yu, B. Vijaya Kumar, and J. Wang, "Unsupervised domain adaptation for semantic segmentation via class-balanced self-training," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 289–305.

[79] Y. Zou, Z. Yu, X. Liu, B. V. Kumar, and J. Wang, "Confidence regularized self-training," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019, pp. 5982–5991.

[80] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Actes de CAP 05, Conférence francophone sur l'apprentissage automatique*, 2005, pp. 281–296.

[81] Y. Zhang, P. David, and B. Gong, "Curriculum domain adaptation for semantic segmentation of urban scenes," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017, pp. 2020–2030.

[82] Y. Zhang, P. David, H. Foroosh, and B. Gong, "A curriculum domain adaptation approach to the semantic segmentation of urban scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.

[83] C. Sakaridis, D. Dai, S. Hecker, and L. Van Gool, "Model adaptation with synthetic and real data for semantic dense foggy scene understanding," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 687–704.

[84] D. Dai, C. Sakaridis, S. Hecker, and L. Van Gool, "Curriculum model adaptation with synthetic and real data for semantic foggy scene understanding," *International Journal of Computer Vision (IJCV)*, pp. 1–23, 2019.

[85] Q. Lian, F. Lv, L. Duan, and B. Gong, "Constructing self-motivated pyramid curriculums for cross-domain semantic segmentation: A non-adversarial approach," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019, pp. 6758–6767.

[86] Y. Chen, W. Li, X. Chen, and L. V. Gool, "Learning semantic segmentation from synthetic data: A geometrically guided input-output adaptation approach," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 1841–1850.

[87] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," *arXiv preprint arXiv:1711.03938*, 2017.

[88] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "BDD100K: A diverse driving dataset for heterogeneous multitask learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[89] T. Sun, M. Segù, J. Postels, Y. Wang, L. V. Gool, B. Schiele, F. Tombari, and F. Yu, "SHIFT: A synthetic driving dataset for continuous multi-task domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[90] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[91] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[92] W.-C. Hung, Y.-H. Tsai, Y.-T. Liou34, Y.-Y. Lin, and M.-H. Yang15, "Adversarial learning for semi-supervised semantic segmentation," in *Proceedings of British Machine Vision Conference (BMVC)*, 2018.

[93] X. Liu, J. Cao, T. Fu, Z. Pan, W. Hu, K. Zhang, and J. Liu, "Semi-supervised automatic segmentation of layer and fluid region in retinal optical coherence tomography images using adversarial learning," *IEEE Access*, 2018.

[94] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 35, no. 8, pp. 1798–1828, 2013.

[95] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 580–587.

[96] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the International Conference on Machine Learning*, 2020, pp. 10 709–10 719.

[97] R. Xu, G. Li, J. Yang, and L. Lin, "Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019, pp. 1426–1435.

[98] K. Saito, D. Kim, S. Sclaroff, T. Darrell, and K. Saenko, "Semi-supervised domain adaptation via minimax entropy," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[99] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 1999.

[100] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[101] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 2015.

[102] Y. Chen, W. Chen, Y. Chen, B. Tsai, Y. F. Wang, and M. Sun, "No more discrimination: Cross city adaptation of road scene segmenters," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017, pp. 2011–2020.

[103] C. Li, D. Du, L. Zhang, L. Wen, T. Luo, Y. Wu, and P. Zhu, "Spatial attention pyramid network for unsupervised domain adaptation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[104] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, pp. 2579–2605, 2008.

[105] J. L. McClelland, B. L. McNaughton, and R. C. O'Reilly, "Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory." *Psychological review*, vol. 102, no. 3, p. 419, 1995.

[106] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*. Elsevier, 1989, vol. 24, pp. 109–165.

[107] R. Kemker, M. McClure, A. Abitino, T. L. Hayes, and C. Kanan, "Measuring catastrophic forgetting in neural networks," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[108] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, 2019.

[109] S. Thrun and L. Pratt, *Learning to learn*. Springer Science & Business Media, 2012.

[110] S. Grossberg, "Adaptive resonance theory: How a brain learns to consciously attend, learn, and recognize a changing world," *Neural Networks*, vol. 37, pp. 1–47, 2013.

[111] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: A survey," *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.

[112] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[113] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 40, no. 12, pp. 2935–2947, 2017.

[114] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, "End-to-end incremental learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 233–248.

[115] K. Shmelkov, C. Schmid, and K. Alahari, "Incremental learning of object detectors without catastrophic forgetting," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017, pp. 3400–3409.

[116] F. Ozdemir and O. Goksel, "Extending pretrained segmentation networks with additional anatomical structures," *International journal of computer assisted radiology and surgery*, vol. 14, no. 7, pp. 1187–1195, 2019.

[117] O. Tasar, Y. Tarabalka, and P. Alliez, "Incremental learning for semantic segmentation of large-scale remote sensing data," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 9, pp. 3524–3537, 2019.

[118] U. Michieli and P. Zanuttigh, "Incremental learning techniques for semantic segmentation," in *Proceedings of the International Conference on Computer Vision Workshops (ICCVW)*, 2019.

[119] ——, "Knowledge distillation for incremental learning in semantic segmentation," *Computer Vision and Image Understanding*, vol. 205, p. 103167, 2021.

[120] A. Douillard, Y. Chen, A. Dapogny, and M. Cord, "Plop: Learning without forgetting for continual semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[121] M. H. Phan, S. L. Phung, L. Tran-Thanh, A. Bouzerdoum *et al.*, "Class similarity weighted knowledge distillation for continual semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[122] G. Yang, E. Fini, D. Xu, P. Rota, M. Ding, T. Hao, X. Alameda-Pineda, and E. Ricci, "Continual attentive fusion for incremental learning in semantic segmentation," *IEEE Transactions on Multimedia*, 2022.

[123] G. Yang, E. Fini, D. Xu, P. Rota, M. Ding, M. Nabi, X. Alameda-Pineda, and E. Ricci, "Uncertainty-aware contrastive distillation for incremental semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2022.

[124] F. Cermelli, D. Fontanel, A. Tavera, M. Ciccone, and B. Caputo, "Incremental learning in semantic segmentation from image labels," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[125] C.-B. Zhang, J.-W. Xiao, X. Liu, Y.-C. Chen, and M.-M. Cheng, "Representation compensation networks for continual semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[126] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[127] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.

[128] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Díaz-Rodríguez, "Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges," *Information fusion*, vol. 58, pp. 52–68, 2020.

[129] P. P. Busto and J. Gall, "Open set domain adaptation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017, pp. 754–763.

[130] J. Zhuo, S. Wang, S. Cui, and Q. Huang, "Unsupervised open domain recognition by semantic discrepancy minimization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 750–759.

[131] J. N. Kundu, R. M. Venkatesh, N. Venkat, A. Revanur, and R. V. Babu, "Class-incremental domain adaptation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[132] R. Volpi, D. Larlus, and G. Rogez, "Continual adaptation of visual representations via domain randomization and meta-learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[133] R. Volpi, P. De Jorge, D. Larlus, and G. Csurka, "On the road to online adaptation for semantic image segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[134] Q. Wang, O. Fink, L. Van Gool, and D. Dai, "Continual test-time domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[135] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2022.

[136] D. Isele and A. Cosgun, "Selective experience replay for lifelong learning," in *Proceedings of AAAI Conference on Artificial Intelligence*, 2018.

[137] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. S. Torr, and M. Ranzato, "Continual learning with tiny episodic memories," *arXiv preprint arXiv:1902.10486*, 2019.

[138] M. D. Lange and T. Tuytelaars, "Continual prototype evolution: Learning online from non-stationary data streams," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[139] E. Verwimp, M. D. Lange, and T. Tuytelaars, "Rehearsal revealed: The limits and merits of revisiting samples in continual learning," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[140] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *Neural Information Processing Systems (NeurIPS)*, 2017.

[141] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio, "Gradient based sample selection for online continual learning," *Neural Information Processing Systems (NeurIPS)*, 2019.

[142] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with A-GEM," in *International Conference on Learning Representations (ICLR)*, 2019.

[143] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu, "Large scale incremental learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[144] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Neural Information Processing Systems (NeurIPS)*, 2017, pp. 2990–2999.

[145] F. Lavda, J. Ramapuram, M. Gregorova, and A. Kalousis, "Continual classification learning using generative models," *arXiv preprint arXiv:1810.10612*, 2018.

[146] J. Ramapuram, M. Gregorova, and A. Kalousis, "Lifelong generative modeling," *Neurocomputing*, 2020.

[147] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, Z. Zhang, and Y. Fu, "Incremental classifier learning with generative adversarial networks," *arXiv preprint arXiv:1802.00853*, 2018.

[148] N. Kamra, U. Gupta, and Y. Liu, "Deep generative dual memory network for continual learning," *arXiv preprint arXiv:1710.10368*, 2017.

[149] C. He, R. Wang, S. Shan, and X. Chen, "Exemplar-supported generative reproduction for class incremental learning." in *Proceedings of British Machine Vision Conference (BMVC)*, 2018, p. 98.

[150] Y. Xiang, Y. Fu, P. Ji, and H. Huang, "Incremental learning using conditional adversarial networks," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[151] F. Zhu, X.-Y. Zhang, C. Wang, F. Yin, and C.-L. Liu, "Prototype augmentation and self-supervision for incremental learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[152] L. Yu, B. Twardowski, X. Liu, L. Herranz, K. Wang, Y. Cheng, S. Jui, and J. van de Weijer, "Semantic drift compensation for class-incremental learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[153] Z. Li and D. Hoiem, "Learning without forgetting," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[154] H. Jung, J. Ju, M. Jung, and J. Kim, "Less-forgetting learning in deep neural networks," *arXiv preprint arXiv:1607.00122*, 2016.

[155] J. Zhang, J. Zhang, S. Ghosh, D. Li, S. Tasci, L. Heck, H. Zhang, and C.-C. J. Kuo, "Class-incremental learning via deep model consolidation," in *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[156] A. Rannen, R. Aljundi, M. B. Blaschko, and T. Tuytelaars, "Encoder based lifelong learning," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[157] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin, "Learning a unified classifier incrementally via rebalancing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[158] P. Dhar, R. V. Singh, K.-C. Peng, Z. Wu, and R. Chellappa, "Learning without memorizing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5138–5146.

[159] A. Douillard, M. Cord, C. Ollion, T. Robert, and E. Valle, "Podnet: Pooled outputs distillation for small-tasks incremental learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, Eds., 2020.

[160] H. Ahn, J. Kwak, S. F. Lim, H. Bang, H. Kim, and T. Moon, "SS-IL: Separated softmax for incremental learning." in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[161] H. Cha, J. Lee, and J. Shin, "Co^{2}L: Contrastive continual learning," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[162] C. Simon, P. Koniusz, and M. Harandi, "On learning the geodesic path for incremental learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[163] X. Hu, K. Tang, C. Miao, X. Hua, and H. Zhang, "Distilling causal effect of data in class-incremental learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[164] G. Wu, S. Gong, and P. Lid, "Striking a balance between stability and plasticity for class-incremental learning," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[165] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[166] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, 2017.

[167] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang, "Overcoming catastrophic forgetting by incremental moment matching," *Neural Information Processing Systems (NeurIPS)*, 2017.

[168] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[169] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. S. Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[170] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proceedings of the International Conference on Machine Learning*, 2017.

[171] X. Liu, M. Masana, L. Herranz, J. Van de Weijer, A. M. Lopez, and A. D. Bagdanov, "Rotate your networks: Better weight consolidation and less catastrophic forgetting," in *Proceedings of International Conference on Pattern Recognition*, 2018.

[172] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[173] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra, "Pathnet: Evolution channels gradient descent in super neural networks," *arXiv preprint arXiv:1701.08734*, 2017.

[174] A. Mallya, D. Davis, and S. Lazebnik, "Piggyback: Adapting a single network to multiple tasks by learning to mask weights," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[175] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in *Proceedings of the International Conference on Machine Learning*, 2018.

[176] R. Aljundi, P. Chakravarty, and T. Tuytelaars, "Expert gate: Lifelong learning with a network of experts," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[177] A. Rosenfeld and J. K. Tsotsos, "Incremental learning through deep adaptation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 42, no. 3, pp. 651–663, 2018.

[178] J. Xu and Z. Zhu, "Reinforced continual learning," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[179] Y.-X. Wang, D. Ramanan, and M. Hebert, "Growing a brain: Fine-tuning by increasing model capacity," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2471–2480.

[180] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 40, no. 12, pp. 2935–2947, 2018.

[181] Y. Liu, B. Schiele, and Q. Sun, "Adaptive aggregation networks for class-incremental learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[182] S. Yan, J. Xie, and X. He, "DER: dynamically expandable representation for class incremental learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[183] C. Bucilua, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proc. of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 535–541.

[184] F. Tung and G. Mori, "Similarity-preserving knowledge distillation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019, pp. 1365–1374.

[185] J. Mańdziuk and L. Shastri, "Incremental class learning approach and its application to handwritten digit recognition," *Information Sciences*, vol. 141, no. 3-4, pp. 193–217, 2002.

[186] G. Nguyen, S. Chen, T. Do, T. J. Jun, H.-J. Choi, and D. Kim, "Dissecting catastrophic forgetting in continual learning by deep visualization," *arXiv preprint arXiv:2001.01578*, 2020.

[187] Y. Xian, Z. Akata, G. Sharma, Q. Nguyen, M. Hein, and B. Schiele, "Latent embeddings for zero-shot classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 69–77.

[188] X. Peng, Z. Huang, X. Sun, and K. Saenko, "Domain agnostic learning with disentangled representations," in *Proceedings of the International Conference on Machine Learning*. PMLR, 2019, pp. 5102–5112.

[189] A. Achille, T. Eccles, L. Matthey, C. Burgess, N. Watters, A. Lerchner, and I. Higgins, "Life-long disentangled representation learning with cross-domain latent homologies," in *Neural Information Processing Systems (NeurIPS)*, 2018.

[190] K. Javed and M. White, "Meta-learning representations for continual learning," in *Neural Information Processing Systems (NeurIPS)*, 2019.

[191] T. J. Draelos, N. E. Miner, C. C. Lamb, J. A. Cox, C. M. Vineyard, K. D. Carlson, W. M. Severa, C. D. James, and J. B. Aimone, "Neurogenesis deep learning: Extending deep networks to accommodate new classes," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 526–533.

[192] Q. Hou, M.-M. Cheng, J. Liu, and P. H. Torr, "Webseg: Learning semantic segmentation from web searches," *arXiv preprint arXiv:1803.09859*, 2018.

[193] D. Modolo and V. Ferrari, "Learning semantic part-based models from google images," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 40, no. 6, pp. 1502–1509, 2017.

[194] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[195] H. Pouransari and S. Ghili, "Tiny imagenet visual recognition challenge," in *CS231N course, Stanford Univ., Stanford, CA, USA*, 2014.

[196] C. Wah, P. W. Steve Branso and, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," California Institute of Technology, Tech. Rep. CNS-TR-2011-001, 2011.

[197] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[198] Z. Mai, R. Li, J. Jeong, D. Quispe, H. Kim, and S. Sanner, "Online continual learning in image classification: An empirical survey," *Neurocomputing*, vol. 469, pp. 28–51, 2022.

[199] M. Jazbec, M. Ashman, V. Fortuin, M. Pearce, S. Mandt, and G. Rätsch, "Scalable gaussian process variational autoencoders," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AIStat)*, 2021.

[200] J. Lucas, G. Tucker, R. B. Grosse, and M. Norouzi, "Don't blame the ELBO! A linear VAE perspective on posterior collapse," in *Neural Information Processing Systems (NeurIPS)*, 2019.

[201] Y. Yacoby, W. Pan, and F. Doshi-Velez, "Failure modes of variational autoencoders and their effects on downstream tasks," in *ICML 2020 Workshop on Uncertainty and Robustness in Deep Learning*, 2021.

[202] B. He and M. Ozay, "Feature kernel distillation," in *International Conference on Learning Representations (ICLR)*, 2022.

[203] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Neural Information Processing Systems (NeurIPS)*, 2015.

[204] S. Zhao, J. Song, and S. Ermon, "Infovae: Information maximizing variational autoencoders," *CoRR*, vol. abs/1706.02262, 2017.

[205] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations (ICLR)*, 2014.

[206] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Neural Information Processing Systems (NeurIPS)*, 2016.

[207] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *International Conference on Learning Representations (ICLR)*, 2018.

[208] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[209] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *International Conference on Learning Representations (ICLR)*, 2019.

[210] "Tensorflow module of BigGAN-deep 512, https://tfhub.dev/deepmind/biggan-deep-512/1. Accessed on 18/03/2020."

[211] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the International Conference on Machine Learning*, 2019, pp. 6105–6114.

[212] "TensorFlow module of EfficientNet-b2, https://tfhub.dev/google/efficientnet/b2/classification/1. Accessed on 18/03/2020."

[213] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[214] C. Peng, K. Zhao, and B. C. Lovell, "Faster ilod: Incremental learning for object detectors based on faster rcnn," *Pattern recognition letters*, 2020.

[215] K. Joseph, S. Khan, F. S. Khan, and V. N. Balasubramanian, "Towards open world object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[216] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira, "Analysis of representations for domain adaptation," in *Neural Information Processing Systems (NeurIPS)*, 2006.

[217] Y. Tian and S. Zhu, "Partial domain adaptation on semantic segmentation," *IEEE Transactions on Circuits and Systems for Video Technology*, 2021.

[218] T. Jing, H. Liu, and Z. Ding, "Towards novel target discovery through open-set domain adaptation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[219] K. You, M. Long, Z. Cao, J. Wang, and M. I. Jordan, "Universal domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[220] K. Saito and K. Saenko, "Ovanet: One-vs-all network for universal domain adaptation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[221] X. Ma, J. Gao, and C. Xu, "Active universal domain adaptation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[222] J. He, X. Jia, S. Chen, and J. Liu, "Multi-source domain adaptation with collaborative learning for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[223] R. Gong, D. Dai, Y. Chen, W. Li, and L. Van Gool, "mdalu: Multi-source domain adaptation and label unification with partial datasets," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[224] Z. Liu, Z. Miao, X. Pan, X. Zhan, D. Lin, S. X. Yu, and B. Gong, "Open compound domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[225] T. Isobe, X. Jia, S. Chen, J. He, Y. Shi, J. Liu, H. Lu, and S. Wang, "Multi-target domain adaptation with collaborative consistency learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[226] Y. Zhao, Z. Zhong, Z. Luo, G. H. Lee, and N. Sebe, "Source-free open compound domain adaptation in semantic segmentation," *IEEE Transactions on Circuits and Systems for Video Technology*, 2022.

[227] R. A. Marsden, F. Wiewel, M. Döbler, Y. Yang, and B. Yang, "Continual unsupervised domain adaptation for semantic segmentation using a class-specific transfer," in *Proceedings of International Joint Conference on Neural Networks*, 2023.

[228] P. Garg, R. Saluja, V. N. Balasubramanian, C. Arora, A. Subramanian, and C. V. Jawahar, "Multi-domain incremental learning for semantic segmentation," in *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, 2022.

[229] T. Kalb, M. Roschani, M. Ruf, and J. Beyerer, "Continual learning for class- and domain-incremental semantic segmentation," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, 2021.

[230] D. Shenaj, F. Barbato, U. Michieli, and P. Zanuttigh, "Continual coarse-to-fine domain adaptation in semantic segmentation," *Image and Vision Computing*, 2022.

[231] C. Simon, M. Faraki, Y.-H. Tsai, X. Yu, S. Schulter, Y. Suh, M. Harandi, and M. Chandraker, "On generalizing beyond domains in cross-domain continual learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[232] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, "Erfnet: Efficient residual factorized convnet for real-time semantic segmentation," *TITS*, 2018.

List of Publications

Journals

[233] M. Toldo, A. Maracani, U. Michieli, and P. Zanuttigh, "Unsupervised domain adaptation in semantic segmentation: a review," *Technologies*, vol. 8, no. 2, 2020.

[234] M. Toldo, U. Michieli, G. Agresti, and P. Zanuttigh, "Unsupervised domain adaptation for mobile semantic segmentation based on cycle consistency and feature alignment," *Image and Vision Computing*, 2020.

[235] U. Michieli, M. Toldo, and M. Ozay, "Federated learning via attentive margin of semantic feature representations," *IEEE Internet of Things Journal*, 2022.

[236] F. Barbato, U. Michieli, M. Toldo, and P. Zanuttigh, "Road scenes segmentation across different domains by disentangling latent representations," *Submitted to The Visual Computer*, 2022.

[237] M. Toldo, U. Michieli, and P. Zanuttigh, "Learning with style: Continual semantic segmentation across tasks and domains," *Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2022.

Conference Proceedings

[238] T. Spadotto, M. Toldo, U. Michieli, and P. Zanuttigh, "Unsupervised domain adaptation with multiple domain discriminators and adaptive self-training," in *Proceedings of International Conference on Pattern Recognition*, 2020.

[239] M. Toldo, U. Michieli, and P. Zanuttigh, "Unsupervised domain adaptation in semantic segmentation via orthogonal and clustered embeddings," in *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, 2021.

[240] F. Barbato, M. Toldo, U. Michieli, and P. Zanuttigh, "Latent space regularization for unsupervised domain adaptation in semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR W)*, 2021.

[241] A. Maracani, U. Michieli, M. Toldo, and P. Zanuttigh, "RECALL: replay-based continual learning in semantic segmentation," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[242] M. Toldo and M. Ozay, "Bring evanescent representations to life in lifelong class incremental learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[243] D. Shenaj, E. Fanì, M. Toldo, D. Caldarola, A. Tavera, U. Michieli, M. Ciccone, P. Zanuttigh, and B. Caputo, "Learning across domains and devices: Style-driven source-free domain adaptation in clustered federated learning," in *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, 2023.

Book Chapters

[244] U. Michieli, M. Toldo, and P. Zanuttigh, "Domain adaptation and continual learning in semantic segmentation," in *Advanced Methods and Deep Learning in Computer Vision*. Elsevier, 2022, pp. 275–303.

Acknowledgments

It feels like just yesterday that I started my journey to the Ph.D., yet three intense and exciting years have passed by, leaving but enriching experiences I will carry with me in the years to come. As with any journey in life, much of the enjoyment comes from the people who travel with you, guiding you in the right directions, sharing the good and bad times, and supporting you when most needed. I would like to express my gratitude to each one of them, for all of this would not have been possible without their presence. First and foremost, I want to dedicate a special thanks to my advisor prof. Pietro Zanuttigh, for pushing me into undertaking this adventure and for offering me his guidance in my Ph.D. endeavors. I would also like to thank all the people in the LTTM Lab that I had the chance to cross path with: Umberto Michieli, Gianluca Agresti, Adriano Simonetto, Francesco Barbato, Donald Shenaj, Sebastiano Verde, Mel Mazen, Elena Camuffo, Daniele Mari, Giulia Rizzoli, Andrea Maracani and Teo Spadotto. A special mention must be done for Umberto, and his constant advice and motivation. Without his help, my journey would have lost much of its potential. I want to extend my gratitude to Mete Ozay, whose supervisory role has taught me invaluable lessons and made me a better researcher. Last, but certainly not least, I would like to thank all my friends for the encouragement, and my family for the endless patience and unconditional support shown in these years.