entropy

MDPI

Article

Emergence of Network Motifs in Deep Neural Networks

Matteo Zambra [1,*], Amos Maritan [2] and Alberto Testolin [3,4,*]

[1] Department of Civil, Environmental and Architectural Engineering, University of Padova, Via Marzolo 9, 35131 Padova, Italy
[2] Department of Physics and Astronomy, University of Padova; Istituto Nazionale di Fisica Nucleare—Sezione di Padova, Via Marzolo 8, 35131 Padova, Italy; amos.maritan@pd.infn.it
[3] Department of General Psychology, University of Padova, Via Venezia 8, 35131 Padova, Italy
[4] Department of Information Engineering, University of Padova, Via Gradenigo 6/b, 35131 Padova, Italy
* Correspondence: matteo.zambra1@gmail.com (M.Z.); alberto.testolin@unipd.it (A.T.)

Received: 27 December 2019; Accepted: 7 February 2020; Published: 11 February 2020

Abstract: Network science can offer fundamental insights into the structural and functional properties of complex systems. For example, it is widely known that neuronal circuits tend to organize into basic functional topological modules, called network motifs. In this article, we show that network science tools can be successfully applied also to the study of artificial neural networks operating according to self-organizing (learning) principles. In particular, we study the emergence of network motifs in multi-layer perceptrons, whose initial connectivity is defined as a stack of fully-connected, bipartite graphs. Simulations show that the final network topology is shaped by learning dynamics, but can be strongly biased by choosing appropriate weight initialization schemes. Overall, our results suggest that non-trivial initialization strategies can make learning more effective by promoting the development of useful network motifs, which are often surprisingly consistent with those observed in general transduction networks.

Keywords: deep learning; artificial neural networks; network motifs; complex systems

MSC: 90B10; 94C15; 68Q32; 68T05

1. Introduction

The topological structure of complex networks can be characterized by a series of well-known features, such as the small-world and scale-free properties, the presence of cliques and cycles, modularity, and so on, which are instead missing in random networks [1–5]. It has been shown that another distinguishing feature is the presence of so-called network motifs [6], which are recurring patterns of interconnections that might serve as building blocks for the evolution of more complex functional units [7,8]. One might thus hope to "understand the dynamics of the entire network based on the dynamics of the individual building blocks" (see Chapter 3 in Reference [9]). In this respect, we can regard network motifs as basic structural modules which bear (in a topological sense) meaningful insights about the holistic behavior of the system as a whole.

Here we apply this perspective to the study of multi-layer (deep) neural networks, which are one of the most popular frameworks used in modern artificial intelligence applications [10,11]. Despite the impressive performance achieved by deep networks in challenging cognitive tasks, such as image classification [12], automatic machine translation [13] and discovery of sophisticated game strategies [14], such systems are still poorly understood [15]. To quote Reference [16], "the theoretical principles governing how even simple artificial neural networks extract semantic knowledge from

their ongoing stream of experience, embed this knowledge in their synaptic weights, and use these weights to perform inductive generalization, remains obscure". The inscrutability of deep learning models mostly stems from the fact that their behavior is the result of the non-linear interaction between many elements, which motivates the use of network science techniques to reveal emergent topological properties [17].

The primary question we address in the present work is whether is it possible to observe the emergence of well-defined network motifs even if the initial (between layer) topology corresponds to a fully-connected graph, where each node (neuron) is connected to all nodes in the neighboring layers. The underlying assumption is that some traces of the learning dynamics will be nevertheless recorded in the final model topology in the form of basic functional modules, thus opening the possibility to relate local network properties with the functioning of the system as a whole. Furthermore, given that the objective of deep learning is to extract high-order statistical features from the data distribution [18], we ask to what extent the final topology depends on intrinsic properties of the training data. To this aim, we systematically compare the motifs emerging in a deep network trained on two different synthetic environments, created according to different generative models.

2. Methods

2.1. Neural Network Architectures

A simple multi-layer feed-forward network was built and trained using the Keras deep learning framework (See https://keras.io/ for documentation). The external deep learning libraries and motifs mining software were used as provided, while the rest of the system was coded from scratch in Python 3.7.4 (See https://docs.python.org/release/3.7.4/ for documentation). The Supplementary Materials (source codes) are available at https://github.com/MatteoZambra/SM_ML__MScThesis. In order to better assess the robustness of our simulations, three different neural network architectures with varying number of neurons have been considered (see Table 1). In all the cases, the architecture consists of a fully-connected multi-layer perceptron model. A broader overview on the robustness results is given in Appendix A, where we also discuss simulations related to a larger-scale, more realistic machine vision problem (e.g., classification of handwritten digits).

Table 1. Network architectures tested. Note that the name of the network is the seed value used for reproducibility purposes. ReLU stands for Rectified Linear Unit, see Reference [10].

Network	Layer	Units	Activation
240120	Input	31	–
	Hidden 1	20	ReLU
	Hidden 2	10	ReLU
	Output	4	Sotfmax
250120	Input	31	–
	Hidden 1	20	ReLU
	Hidden 2	20	ReLU
	Output	4	Sotfmax
180112	Input	31	–
	Hidden 1	30	ReLU
	Hidden 2	30	ReLU
	Output	4	Sotfmax

2.2. Learning Environments

Inspired by the recent study of Saxe and colleagues [16], two synthetic sets of data were purposefully generated to embed different statistical structures, to investigate whether different environmental conditions would lead to the emergence of specific topological signatures. The first environment, encoded as a binary tree data set, contained a hierarchical structure. The second

environment, encoded as an independent clusters data set, resolved in a more sparse and glassy statistical footprint, as shown in Figure 1. The structure of each data set will be briefly reviewed in the following; for a more detailed description, the reader is referred to the Appendix B. As customary in the machine learning literature, a data instance is thought of as a vector of random variables representing some measurable features. We can conceive each data instance as being generated by sampling from a (probabilistic) graphical model, whose nodes represent the random variables of interest, and whose edges represent their mutual relationships. For both the data sets considered, samples can be divided into four classes, as explained more in detail below.

2.2.1. Binary Tree Data Set

The binary tree data set is thought as a slight modification of the generative model illustrated in Reference [16]. This data generator is designed to create data instances displaying a hierarchical structure, as shown in Figure 1a. Each generator run produces a data instance, whose features attain values among $\{-1, +1\}$. The initial feature (root node) is sampled uniformly and its value diffuses through the tree branches. The rationale underlaid is as follows: If the root node attains the value $+1$, then the left child inherits the $+1$ value and the right child (together with all its progeny) is assigned the value -1. Contrarily, if the root node happens to be -1, then the right child inherits the $+1$ value. From the root node children on, the criterion is probabilistic. One of the children of a $+1$ node inherits the same value with probability ε, that is a threshold set *a priori* to 0.3 (see Figure 2a). Clearly, the smaller this threshold, less likely is the value to flip. To perform such a stochastic flip, for each node one samples a value p uniformly distributed in $[0, 1]$. For example: a node i has the value $+1$. Given ε, sample $p \sim U(0, 1)$. If $p > \varepsilon$, then the left child of i (which has the node index equal to $2 \times i + 1$, where the root node number is (Note that the node number is not the value stored in such node. This latter may be -1 or $+1$ while the former ranges from 0 to the total number of nodes $N = 2^D - 1$. $D = 1, \ldots$ refers instead to the depth of the tree, that is, how many node levels it has. Note that a linear array storage is used for the binary tree data structure). $i = 0$) inherits the value $+1$ and the right child instead inherits -1. If $p \leq \varepsilon$, the left child inherits -1 and the right child inherits $+1$. The features of each data sample are the collection of all the nodes in the tree structure for a given generator run, with the first feature being the root node, the right and the left children contain the second and third features respectively and so forth.

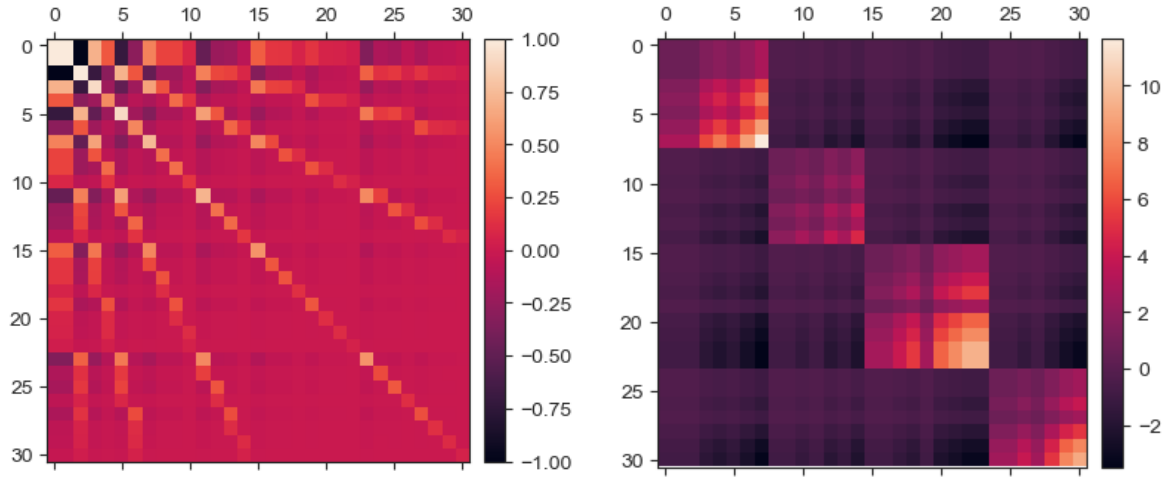
The number of classes is set selecting a level in the tree: The root node (level 1) identifies two axes of distinction, that is whether its value is $+1$ or -1. The data set fed to the network for our analyses is generated accounting for a level of distinction set to 2, which thus generates four different classes (see Appendix B). The choice of the level of detail set to 2 means that one accounts for the classes identified by the equality of the data samples up to the random variables of the second tree level plus the respective outcomes; all the other variables may differ, thus guaranteeing some variability in the patterns belonging to the same class.

2.2.2. Independent Clusters Data Set

The independent clusters data set is designed to endow the data instances ensemble with a block-diagonal statistical signature, as displayed in Figure 1b. For such purpose, a simple graph is created as in Figure 3a. For consistency with the number of features and classes of the binary tree data set, the same number of random variables involved is chosen, and the number of independent groups is set according to the number of classes of the binary tree data set. In the full graph, as the embedding shown in Figure 3a, some connections are gradually eliminated to reproduce the situation in Figure 3b. The rationale beneath is as follows: In a fashion similar to the simulated annealing algorithm [19], a temperature schedule is set, where the initial temperature is chosen to be the reciprocal of the longest edge and similarly the final value is the reciprocal of the shortest edge

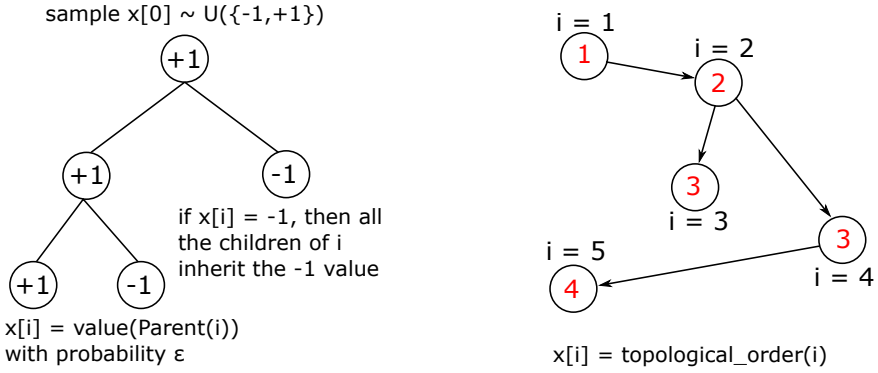
$$\begin{cases} T_0 &= (\max_e\{e \in \mathcal{E}\})^{-1} \\ T_f &= (\min_e\{e \in \mathcal{E}\})^{-1}. \end{cases} \tag{1}$$

The schedule steps are equally spaced and the number of such steps is a parameter that has been fine-tuned to have the desired final scenario of disjoint, arbitrarily intra-connected groups. Note that by thus doing it could be that from a step to the next, shorter links may be erased while retaining longer ones, which were short enough to survive in the last step, hence melting schedule must be set sensibly. Unlike simulated annealing, the temperature rises in this case. For each step, edges greater or equal than the inverse of the temperature value are deleted. Each of these four graphs is directed, to allow for more efficient sampling. Each random variable (i.e., node value) is initialized to the value of -1. Subsequently, one selects randomly a group among the four available and the topological order of such nodes is set: The node with no incoming connection has topological order 1. The nearest neighbors of the latter have topological order 2, and so forth (see Figure 2b). For the sake of simplicity, this data set is created in such a way that a random variable of the selected group bears as value its topological order. This translates in a straight-forward classification task since all the data samples belonging to a class are equal: All the features of a data sample are set to -1, except those corresponding to the nodes of the selected group, which are set equal to their topological ordering (see Appendix B for a thorough explanation). The construction of the structure depicted in Figure 3b is motivated by the necessity to dispose of a data set in which some random variables share a probabilistic relationship and some others are independent. Referring to Figure 1b, the regions in the covariance matrix related to dependent variables are visible as the diagonal blocks, while the background appears more *glassy*. This latter is not fully homogeneous but has rather a chessboard-like textured since the numerical values attained by the non-chosen groups will somehow be related to the values of the chosen group.



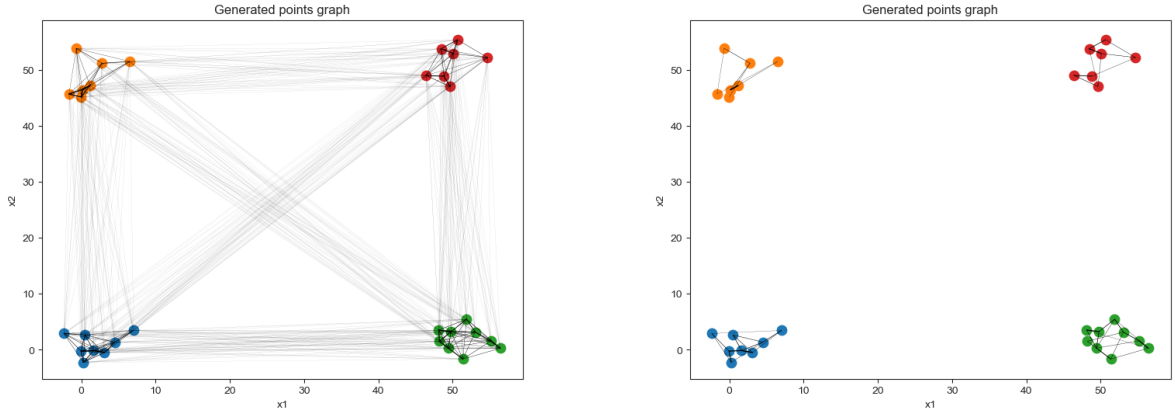
(**a**) Binary tree. (**b**) Independent clusters.

Figure 1. Covariance matrices of the two data sets considered. In (**a**) the variables involved in the covariance computation are all the nodes of the tree structure, from the root node to the leaves. In (**b**) the variables involved are all those constituting the graph in Figure 3. Note that in covariance matrices it is not granted that the elements range in $[-1, +1]$. In fact, the features of the tree-generated data attain values among $\{-1, +1\}$, while in the clusters data set, the random variables involved in the covariance computation attain their topological ordering values. The labels of the features range between 0 and 30, meaning that there are 31 features overall, in both the data sets.



(**a**) Binary tree. (**b**) Independent clusters.

Figure 2. Rationale behind the data sets creation. Note that in the clusters case the red labels represent the topological orderings of the respective nodes. The *i* indices represent the nodes numbers.



(**a**) Fully connected graph. (**b**) *Molten* graph.

Figure 3. Representation of the process for generating data instances from the independent clusters, showing two subsequent stages of the data set generation: The connections between different groups are gradually eliminated in order to obtain independent graphs. Note that the geometric coordinates do not impact the values attained by the nodes; they are temporarily assigned during the creation stage for the purpose of visualization.

2.3. Initial Conditions

Besides varying the statistical structure of the learning environment, we also investigated whether the emergence of different topological signatures could also be related to the use of different initialization schemes for the connection weights. To this aim, we considered three different initialization schemes. In the first case, we used the classic "Normal" initialization method, where each connection weight is randomly sampled from a Gaussian distribution with zero mean and small variance, that is $w \sim \mathcal{N}(0.0, 0.1)$. In the second case, we exploited the "Orthogonal" initialization method proposed in Reference [20], where weight matrices of adjacent layers are constrained to be orthogonal. This scheme has been proved to grant depth-independent training speed, which is desirable as the network becomes deeper. Finally, in the third case we exploited the popular "Xavier" (equivalently called "Glorot") initialization method [21], where the mean is zero and the variance of the Gaussian distribution is defined according to the number of connections of each layer, that is:

$$\sigma^2 = \frac{k}{n_{\text{in}} + n_{\text{out}}}, \tag{2}$$

where k depends on the activation non-linearity and $n_{\text{in, out}}$ are the number of incoming/outgoing connections of each layer. This initialization scheme enjoys widespread popularity, since it has been empirically shown to mitigate optimization issues affecting deep networks.

2.4. Task and Learning Dynamics

The task accomplished by the network is multi-class classification. The data sets are homogeneous in terms of design matrix dimensions and items in the labels set, thus the network architecture is the same in both the cases: Once the parameters are initialized, the data structure containing these parameters and connectivity information is trained both on the binary tree and independent clusters environments.

Stochastic (mini-batch) gradient descent was used to adjust the network's weights. Learning rate was initially set to 0.01, and then decayed using a factor of 10^{-6}. Nesterov acceleration was added with momentum set to 0.6. Given the straightforward structure embedded in our data sets, simple models should be capable of reaching a perfect level of classification accuracy.

2.5. Mining Network Motifs

Network motifs can be defined as "patterns of interconnection occurring in complex networks at numbers that are significantly higher than those in randomized networks" [6]. In the present work, arrangements of four and five nodes are inspected. The statistical significance of such a pattern of connections can be identified by computing the Z-score (equivalently referred to as significance score), which is the number of times a given motif appears in the considered network with respect to the average number of occurrences of the same motif in an ensemble of random replicas of the original network. It is measured as a distance in units of standard deviations:

$$Z = \frac{N_{\text{real}} - < N_{\text{random}} >}{\sigma_{\text{random}}}. \tag{3}$$

Once the neural networks were trained, model parameters were extracted and transposed in a proper graph data structure, so that network motifs mining can be carried on using an external tool. The FANMOD (See http://theinf1.informatik.uni-jena.de/motifs/ for executable, sources, license and relevant papers.) motifs mining software was used to analyse the graph extracted from the model [22]. A comprehensive account on the algorithmic complexity and technical details about network motifs mining tools is beyond the scope of this work. However, a detailed overview of the underlying algorithmic machinery for the FANMOD software is provided in Reference [23], and state-of-the-art advances and performance benchmarks are discussed in Reference [24].

Since retaining all the connection weights of the initial and trained models would imply the presence of redundant network motifs, all weights relatively close to zero were neglected in the subsequent analyses. The identification of negligible weights was carried out by fitting a Normal distribution to the weights histogram, and selecting as exclusion zone the set of the weights $w : p(w) \geq c \max_w p(w)$, where p is the fitted Normal density function. The value of the cut-off threshold c plays an important role: the greater this value, the smaller the exclusion region (see Appendix C for details). The majority of the connection weights fell in this zone, hence the subsequent analyses accounted for the strongest connections, either positive or negative valued, thus letting to emerge only the most significant topological features.

2.6. Biological Analogy: Neurons and Protein Kinases

The internal working of transduction networks is based on the cooperation between processing units, and the subsequent arrangement of those [25] (To deepen the contents of this section, the interested reader is also referred to Reference [9], to which the topic and notation adopted in the following are inspired.). Sensing environmental stimuli, processing this information and eventually transcribing it to gene expression is done by passing this signal through a network whose units are

protein kinases. Such units play the role of nodes in the network, and interactions among those—for example, phosphorylation—are the edges. The activity of these units are modelled through first-order kinetics. The essential items are:

- the kinases of a first layer, the concentration of which is denoted as X_j, with $j = 1, \ldots, n$;
- the target kinase of a second layer, the concentration of which is denoted Y;
- the rate of phosphorylation $r(Y) = Y_0 \sum_j v_j X_j$, being v_j the rate of kinase X_j.

Call Y_0 and Y_p the concentration of un-phosphorylated and phosphorylated kinase Y respectively. The concentration of kinase Y remains constant, that is $Y_0 + Y_p = Y$. Then the rate of change of activated kinase Y is given by the difference between the rate of phosphorylation r and the rate of de-phosphorylation of the same kinase Y, at a rate α. In formulae:

$$\begin{cases} r(Y) &= Y_0 \sum_j v_j X_j \\ Y &= Y_0 + Y_p \\ \dfrac{dY_p}{dt} &= r(Y) - \alpha Y_p. \end{cases} \tag{4}$$

Referring to the case of steady-state dynamics, straight-forward calculations yield that the concentration of active Y is non-linearly proportional to the weighted sum of the inputs X_j, as depicted in Figure 4:

$$Y_p = \frac{\sum_j w_j X_j}{1 + \sum_j w_j X_j}, \tag{5}$$

where $w_j = v_j \alpha^{-1}$. A sensible threshold value for this weighted sum is thought to be 1 approximatively. After the value of the input exceeds 1, the target kinase activity starts to be sensible. Now assume that this simple model involves m target kinases. Then:

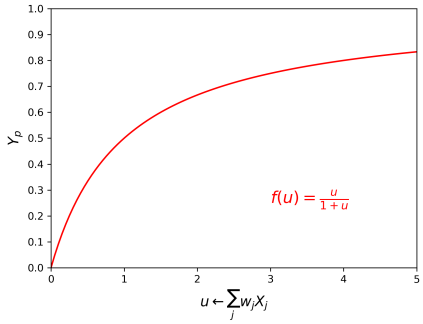


Figure 4. Behavior of the threshold function which quantifies the activity of target kinase, that is Y_p, as a function of the weighted sum of the input signals. A sensible value of the input weighted sum for the target unit to show activity is assumed to be approximatively 1 [9]. Would one not to make such an assumption, then the expression of the hyper-locus referred to in the main text is more generally $\sum_j w_j X_j = k$.

$$\sum_{j=1}^{n} w_{ij} X_j = 1 \quad i = 1, \ldots, m \tag{6}$$

identifies the hyper-plane in the space of the inputs that excludes regions of high and low activity, depending on the connection strengths. It happens that by stacking more of such three-nodes modules (n input signals from the kinases $X_1, \ldots, X_n$ and the target unit Y), one can obtain complex geometries of the activity region in the input space. It is shown that the motifs encountered most often in transduction networks are the so-called diamonds and bi-parallel. As we will discuss later, these motifs match those found in our analyses, as depicted below.

The analogy with binary classification is hinged on the creation of the hyper-plane. Assume that the weighted input is $u = \sum_j w_{ij} X_j$, the numerical value of which is known. To determine whether the unit Y is active, one needs to compare the input u with the hyper-locus that identifies the regions of activity. Assume that the target unit activates once the threshold 1 is exceeded, then:

- If $u \geq 1$ then target unit Y activates and propagated the signal forward in the system to a third layer. But
- If $u < 1$ then Y is not sufficiently triggered to propagate the signal, that is, to phosphorylate the next unit.

The hyper-space $W\boldsymbol{X} \geq \mathbf{1}, \mathbf{1} = \{1\}^n$, identifies the set of weights and activities such that the target units is activated. The subtlety in this analogy is that the weights W cover a relevant role too: In transduction networks change of such weights is subjected to regulatory mechanisms or evolutionary pressure [9], and in the process of gene expression transcription these weights values are given. They do not play the role of adjustable parameters in such a way to minimize a given error metric. In neural networks, on the other hand, weights adjustment is pivotal in the learning process, and such variations are performed on a faster timescale.

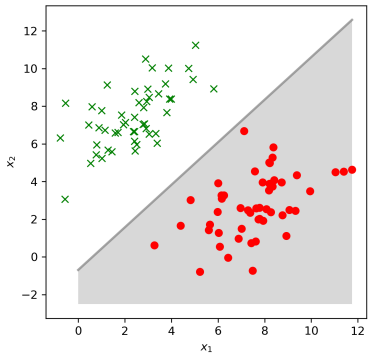
In neural networks one encounters a similar scenario: A neuron is fed with an array of incoming signals, coming from the activities of the previous layers neurons. The weighted sum of these signals is added to an activation threshold value, called bias. The resulting value undergoes a non-linear transformation. In this way it is possible to identify an hyper-plane in the input space that separates the input patterns of signals, as in Figure 5. The "state equations" of a simple one-hidden-layer network are the following:

$$\begin{cases} \boldsymbol{h} &= f\left(\boldsymbol{x}W^{(1)} + \boldsymbol{b}^{(1)}\right) \\ \boldsymbol{y} &= f\left(\boldsymbol{h}W^{(2)} + \boldsymbol{b}^{(2)}\right) \end{cases}, \tag{7}$$

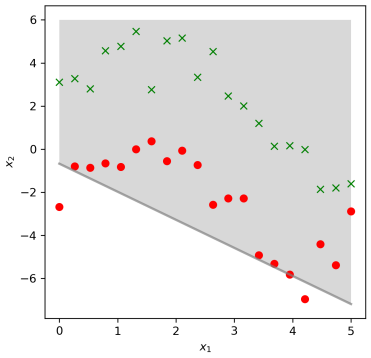
with

$$\begin{aligned} &\boldsymbol{x} \in \mathbb{R}^{N_{\text{input}}}, \quad \boldsymbol{h} \in \mathbb{R}^{N_{\text{hidden}}}, \quad \boldsymbol{y} \in \mathbb{R}^{N_{\text{output}}} \\ &\boldsymbol{b}^{(1)} \in \mathbb{R}^{N_{\text{hidden}}}, \quad \boldsymbol{b}^{(2)} \in \mathbb{R}^{N_{\text{output}}} \\ &W^{(1)} \in \mathbb{R}^{N_{\text{input}} \times N_{\text{hidden}}}, \quad W^{(2)} \in \mathbb{R}^{N_{\text{hidden}} \times N_{\text{output}}}. \end{aligned}$$

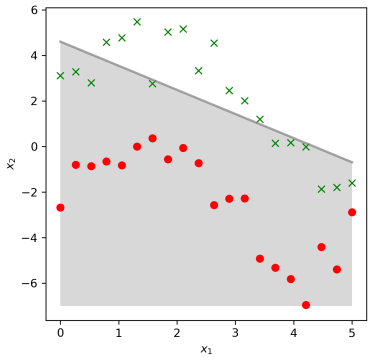
Suppose that this network has one hidden layer with N_{hidden} units, N_{input} input units, N_{output} output units and $f(\cdot)$ is a generic non-linearity. In the framework of neural networks these functions are generally monotonically increasing, as for example the "logistic sigmoid" $\sigma(x) = (1 + \exp(-x))^{-1}$. The vectors $\boldsymbol{b}^{(k)}$ represent the activation thresholds of both the hidden units and output units—also called *biases* and the matrices $W^{(k)}$ are the connection weights, $k = 1, 2$. Here the hyper-plane is identified by the weighted sum in the arguments of $f(\cdot)$, which purpose is to capture higher-order correlations in the input features and the composition of many non-linear blocks allows the synthesis of high-level abstraction of the domain [10]. Figure 5 gives a visual idea of the hyper-planes composition and the result in terms of decision boundary geometry. Signal flow in the system, from the input layer units to the output nodes, is strictly feed-forward and once the guessed label is observed in this latter layer, it is compared with the ground truth. Based on the mismatch, model parameters (connection weights and node biases) are adjusted, in such a way to minimize the prediction error, back-propagating such error in a reverse way along the layers constituting the network [26].



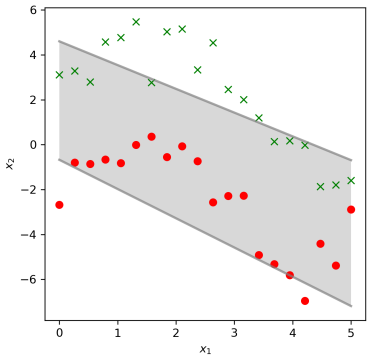
(**a**) Exclusion locus of a single neuron.



(**b**) Exclusion locus of the first triad.



(**c**) Exclusion locus of the second triad.



(**d**) Combination of the two hyper-loci.

Figure 5. In (**a**) the x_1 and x_2 coordinates represent the features of a fictitious data vector, featuring two random variables, in a case of linear separability. Here two input neurons map the input features to a binary label. In (**d**), stacking exclusion hyèer-loci as those in (**b**,**c**), due to a single neuron, one can obtain more intricate decision boundaries. In this graph, it is shown how the joint contribution of two such loci can allow one to go beyond the case of binary classification and linear separable classes, once the problem becomes more complex.

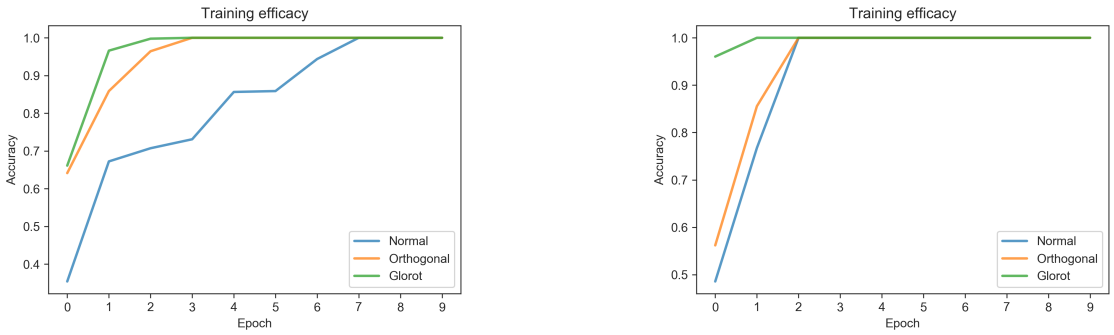
Given these analogies between biological transduction networks and artificial neural networks, it is legitimate to hypothesize that information processing in both classes of systems might be carried out using similar computational structures. Reference [9] argues that "Multi-layer perceptrons allow even relatively simple units to perform detailed computations in response to multiple inputs. The deeper one goes into the layers of perceptrons, the more intricate the computations can become". If one thinks of "intricate computations" as the computation of appropriate decision boundaries, then this task is precisely what is accomplished by multi-layer perceptrons. Individual neurons (absorbing an arbitrary length input) could only discriminate two classes, as in Figure 5a (in this case one has only two input features), but stacking together multiple layers of neurons allows to create more intricate and complex decision loci in the input space, as in Figure 5d. Panels Figure 5b–d refer to the combination of two triads as in Figure 5a, assembled so to form a simple neural network with two input neurons and two output units, with no hidden layers. Figure 5b is the exclusion locus of the triad formed by the input units and the first output unit, Figure 5c analogously refers to the triad in which the output unit involved is the second one. This trivial example shows how hyper-planes designed by simple groups of units arrange to identify less obvious exclusion hyper-subspaces and note that stacking more layers one can go beyond straight lines.

3. Results

As we will point out below, our analyses on network motifs displayed an intriguing consistency with transduction networks. Results for the four-nodes motifs will be presented first, since they allow enjoying a broader perspective on the internal functioning of the artificial networks compared to the biological counterparts. Five-nodes motifs allow for a closer inspection of how the learning environment and the emergence of topological structures relate to each other, but at the same time the emerging patterns are less easily interpretable.

3.1. Learning Efficacy

For all models the training accuracy peaks to the top value of 1.0 in few epochs. However, as shown in Figure 6, the Normal initialization scheme resulted in the slowest learning convergence. The orthogonal initialization scheme allowed convergence in fewer epochs, while the Xavier scheme resulted in the fastest convergence. These findings suggest that initialization plays a crucial role in shaping learning dynamics: one possible explanation could be that the orthogonal and Xavier schemes impress a sharper fingerprint to the initial significance landscape of network motifs, as we will discuss below. In other words, faster convergence toward the optimal set of connection weights might be promoted by biasing the initial set of network motifs. A sharper initial significance landscape is indeed common in the initialization schemes displaying faster convergence.

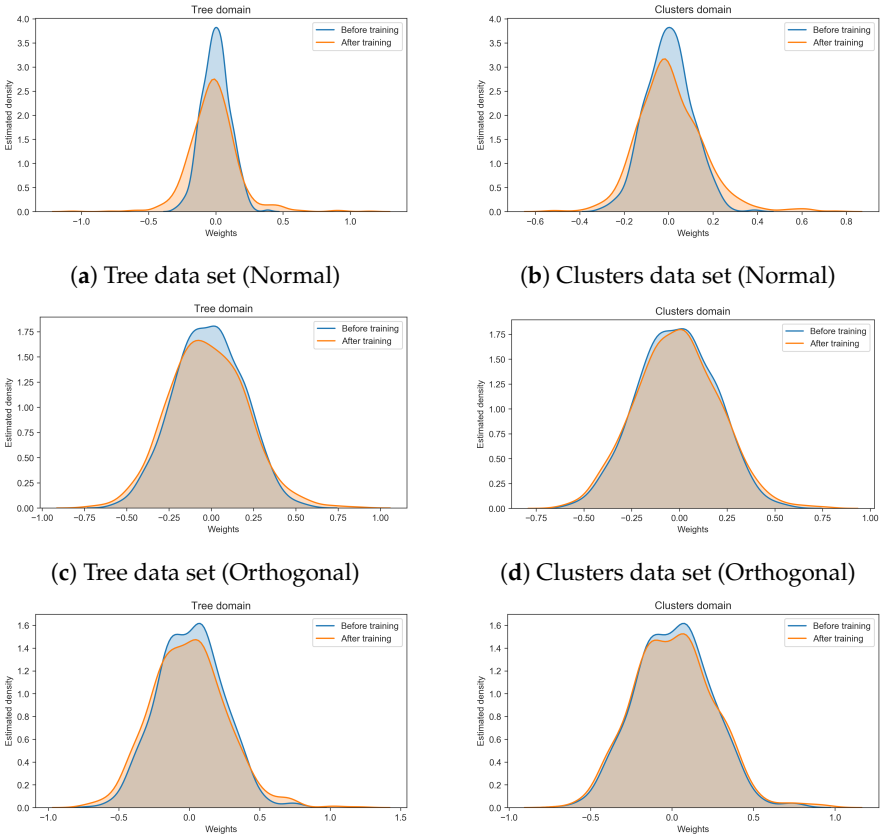


(**a**) Binary tree data set. (**b**) Independent clusters data set.

Figure 6. Efficacy of initialisation schemes for (**a**) binary tree and (**b**) independent clusters data sets. Note that the orthogonal matrices initialisation grants the best performance in terms of training speed and note also that the independent clusters environment is easier to be learned, likely owing to its statistical sparsity.

3.2. Emerging Network Motifs

Figure 7 shows how the weights distribution changes during the course of learning. As noted in previous studies [17] the effect of learning is mostly evident in the tails of the distributions and, in our simulations, especially for the Normal initialization scheme. This suggests that the Orthogonal and Xavier initializations might help building more effective motifs since the beginning, by imposing a stronger bias to the initial structure of the network. A statistical analysis revealed that, indeed, although after learning the weights became larger in absolute value for all the initializations and data sets considered (see Table 2), such difference was significant only for the network initialized using the Normal scheme and trained on the tree data set ($p < 0.01$). Figure 6 depicts how the normal initialization scheme renders a slower convergence. This points out that the correlation between the initial distribution and the evolved distribution in the case of normal initialization decreases, implying also a decreased correlation in the motifs significance profiles, see Figure 8d.



(**a**) Tree data set (Normal)

(**b**) Clusters data set (Normal)

(**c**) Tree data set (Orthogonal)

(**d**) Clusters data set (Orthogonal)

(**e**) Tree data set (Xavier)

(**f**) Clusters data set (Xavier)

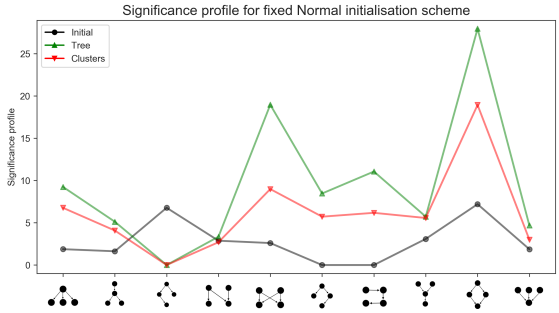
Figure 7. These plots depict the variation that the weights population experiences when trained on different data sets, using different initialization strategies. Results refer to the network 240120.

Table 2. Mean and standard deviation of the weights absolute values, before (i.e., "initial") and after learning on the two different data sets. This analysis was carried out on the set of non-negligible weights, as explained in Section 2.5.

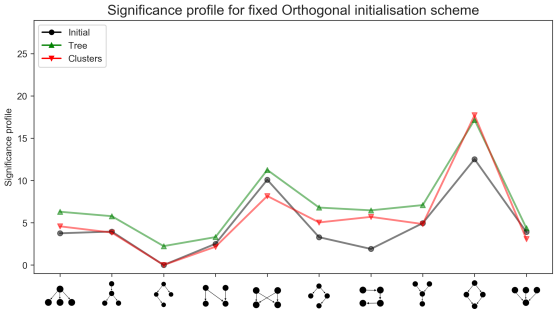
Initialization	Initial	Tree	Clusters
Normal	0.206 ± 0.206	0.283 ± 0.319	0.254 ± 0.265
Orthogonal	0.346 ± 0.358	0.370 ± 0.400	0.366 ± 0.380
Glorot	0.376 ± 0.387	0.410 ± 0.432	0.390 ± 0.404

Figure 8 shows the overall trend of change for the most common 4-nodes motifs: The x-axis gathers the motifs prototypes, the respective y value is the significance score, obtaining a significance profile (Note that significance scores are sometimes normalized, allowing to superimpose (and thus compare) significance profiles referred to different instances of a complex network [27]. However, in the present work normalization is avoided, since the network inspected (hence the size of the system) is the same for all the analyses. Non-normalized scores also allow to better understand the magnitudes of the detected effects.). As evident by comparing Figure 8a–c, the significance profiles resulting from different initialization schemes display a remarkable self-similarity, suggesting that this set of basic structures might support information processing in all the data sets considered. Interestingly, several of such motifs are also consistent with those commonly found in biological transduction networks (the fifth and the ninth motifs from the left), suggesting a potential overlap of computational mechanisms. The fundamental feature of the analogy is the identification of an hyper-plane which classifies the nature of a given input—which comes as a weighted sum, in both transduction and neural networks. As mentioned above, the amount of change in the weights seems to be correlated to the change in the significance profiles—in the case of Normal initialization, the variation is most

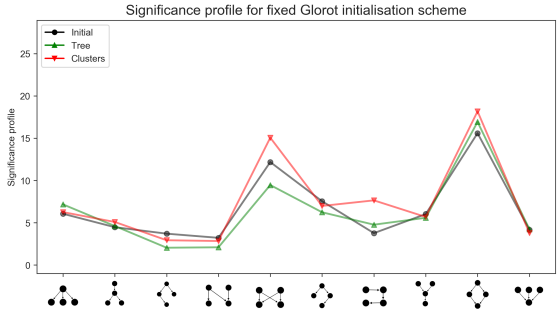
severe, while in the other two cases the significance profile seem to be already biased before learning takes place.



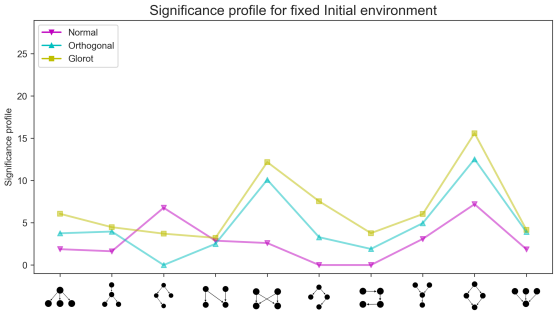
(**a**) Normal initialization scheme.



(**b**) Orthogonal initialization scheme.



(**c**) Xavier initialization scheme.

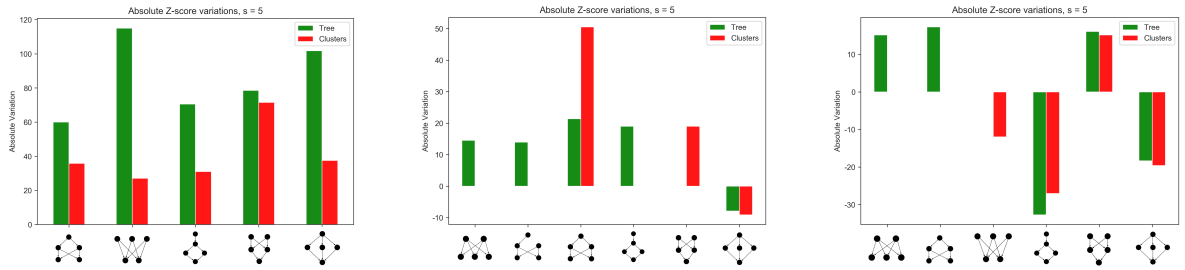


(**d**) Initial profile for schemes.

Figure 8. Four-nodes motifs. Significance profiles accounting for different initialization schemes and the case of the initial landscape for different initialization schemes. Note that in panel a, owing to the small variance, the initial significance profile is flatter. In panel d the profiles depict the fingerprint each initialization scheme impresses to the initial significance landscape, that is, curves therein are the collection of the black curves in the first three panels, that refer to the initial significance profile. The Normal initialization scheme is clearly milder than the other two, due to the values sampled by each initial conditions generation. Note that the seventh motif (from the left) is a chain involving one node of all the four layers: it is displayed folded for graphical convenience. Results refer to network 240120.

Figure 9 shows the overall pattern emerging from the analysis of the most common 5-nodes motifs. On the one hand, it is possible to appreciate the exclusiveness of the motifs characterizing each arrangement of learning environments, suggesting that the learning domain influences the emergence of particular topologies. On the other hand, results are less obvious compared to the case of 4-nodes motifs. Referring again to Figure 8a–d, it is not clear the extent in which the emergence of the most significant motifs is spontaneous or biased by the initial profile (black lines). Orthogonal matrices and Xavier schemes, by their design, sample larger parameters values, hence the model configuration is conditioned by the initial, albeit random, weights landscape. Results in Figure 9 do not provide a definitive answer to this question: While it may seem that different network motifs emerge in response to different initial and learning environments, it is not clear why and when a certain topology is observed. The multi-layer perceptron motif (the last but one motif in panel Figure 9a) and its variations appear through the different scenarios. In Reference [9] it is argued that such a structure can be viewed as a combination of diamond four-nodes motifs (the last but one motif in the panels of Figure 8). Albeit one may be tempted to think that motifs emerge as self-organized modules that encode domain-specific information, it is not clear whether the emergence of different motifs stems from the diversity in learning environments and initial conditions, or whether it might partially due to the noisy by-product of the learning dynamics itself.

Interestingly, a very similar patter of findings emerged from an additional set of simulations carried out using a more realistic input (see Appendix A), that is, images of handwritten digits. Although the deep network architecture was fairly different in this case (featuring three hidden layers and a greater number of neurons), the resulting network motifs match those found in the networks trained on the tree and clusters data sets, and also the change in significance profiles follows a comparable trend.



(**a**) Normal scheme. (**b**) Orthogonal scheme. (**c**) Xavier scheme.

Figure 9. Five-nodes motifs. Total *Z*-score variations accounting for the difference in significance before and after training. Figure refers to most significant motifs, having analysed the weighted graph from the model. Results refer to network 240120.

4. Discussion

In complex networks, individual units by themselves do not accomplish any particularly relevant function, because it is the coordinated arrangement of groups of units (i.e., their *interactions*) that allows for the emergence of system-level, macroscopic properties [28]. In the present work, we thus explored how information processing in deep networks might emerge as a combination of simple network motifs. Starting from these key observations:

- larger motifs may be seen as arrangements of smaller motifs, for example "Diamonds combine to form multi-layer perceptron motifs" [25];
- these smaller motifs arrangement gives rise to more complex computation: "Adding additional layers can produce even more detailed functions in which the output activation region is formed by the intersection of many different regions defined by the different weights of the perceptron" [9];
- domain representation is carried out by the composition of subsequent non-linear modules, which "transform the representation of one level (starting with the raw input) into a representation at a higher, slightly more abstract level" [10];

we hypothesized that in deep neural networks the learning dynamics may rely on the same processing mechanisms used by transduction networks.

Our simulations suggest that this might indeed be the case—network motifs might form spontaneously for the purpose of efficient information processing, so that each module deals with a small number of input features, and subsequent (deeper) processing can rely on fewer signals from previous neurons. High-level features might thus be abstracted in a layer-wise and motifs-wise fashion.

Notably, our analyses also suggest that some weights initialization strategies give a stronger imprinting to the initial significance landscape of possible motifs. The Normal initialization scheme results in a flatter initial landscape, which might underly the slower convergence speed of this type of initialization. The environment may thus be considered to be learned once relevant information processing structures come to develop—if a scheme provides the initial configuration with a preventive signature of such structures, learning will be much faster.

5. Final Remarks and Further Improvements

5.1. Evaluation with Other Classes of Deep Learning Models

Our simulations have been focused on feed-forward neural networks, which are the workhorse of deep learning, but there exist many other classes of models to which our methodology could be applied. Notable examples would be deep networks with *bidirectional* [29,30] and *recurrent* [31] connectivity, which might even allow for the emergence of a richer variety of motifs, or models featuring *convolutional* and LSTM architectures [32–34]. In this respect, it should be stressed that the proposed approach should hold for the analysis of any network-reliant deep learning model.

5.2. Presence of Combinatorial Biases

It would also be useful to better investigate whether particular motifs might emerge simply as a consequence of some combinatorial bias induced by the design topology of the multi-layer perceptron itself. If so, some of the significant four-nodes motifs we detected might appear because of unavoidable initial imprinting due to the topology of the model, and not necessarily because of their critical role in information processing. Further insights into this issue could be gained by also analyzing the case of five-nodes groups, or by enforcing an initial landscape in which the motifs observable are those that appear to be most rare. This way one could observe whether such structures are rejected as a result of learning, favoring instead those discussed above.

5.3. Sensitivity to Free Parameters

Referring to the Appendix C, it should be noted that our simulations involved a certain number of parameters (e.g., the threshold used to discretize the weights, or to exclude negligible weights from the analyses) that were often set according to heuristics. However, we should note that the results presented turned out to be robust to small variations in the choice of such parameters. The choice of mining a weighted or unweighted network also plays a role. Here we presented results related to a weighted analysis, which introduces variability in the discovered motifs. More specifically, motifs were composed of connections that fall in four categories: close to zero (i.e., not present), strongly positive, strongly negative and mildly positive/negative. Also, one could either account for most significant or most typical motifs, the former being those instances in the same group of isomorphic graphs that display the largest significance and the latter are those which have a significance score that is closer to the average, over the same isomorphic group. For a broader account on the problem of graph isomorphism, the reader is referred to References [35,36].

5.4. Scalability

One important research direction would be to apply the proposed framework for the study of larger-scale deep learning systems, which can learn more intricate statistical structure from big data sets where the patterns might not be easily separable (as in the cases presented here). Scalability remains one of the main concerns and an important direction for further research. This problem does not impact training (which can be optimized using high-performing parallel hardware [37,38]) but rather the motifs mining stage. Motifs searching relies on "edges sampling" [24] and this implies larger computational resources for larger graphs.

5.5. Alternatives to Motifs Mining Algorithms

Finally, given that classical motifs search algorithms may present some limitations (e.g., related to the computational complexity, as outlined above), it would be interesting to also investigate approaches based on spectral graph partitioning. Indeed, finding community structure in complex networks is by no means a novel problem (see Reference [39]), and in this respect graph partitioning can be accomplished with spectral techniques. The "spectral" adjective refers to the eigenvalues of the

graph Laplacian [40]. However, a difference to pay attention at in this case would be that while motifs, as referred to above, are intended as patterns on interconnections that may encode functional properties of a given network, partitioning techniques would find community structures which are not granted to match the topologies that one could find with a motif-based approach. Moreover, a definition of sub-graph homogeneity would be necessary. Some previous work has been devoted to such definitions—in Reference [41], the (information-theoretic) entropy of a connections group is used as a measure of homogeneity, while in Reference [42] internal coherence metrics are adopted. One subtlety to account for is that network motifs are not granted to be perfectly homogeneous, hence topologies in which some edges are negative-valued and others are positive-valued could be useful. However, this approach could significantly speed up the sub-graphs research process, so it would constitute an interesting topic for future research.

Supplementary Materials: Code available at https://github.com/MatteoZambra/SM_ML__MScThesis.

Author Contributions: Conceptualization, A.M., A.T.; Data curation, A.T., M.Z.; Formal analysis, A.M., A.T., M.Z.; Funding acquisition, see below; Investigation, M.Z.; Methodology, A.M., A.T., M.Z.; Project administration, A.M., A.T.; Resources, A.M., A.T., M.Z.; Software, M.Z.; Supervision, A.M., A.T.; Validation, M.Z.; Visualization, M.Z.; Writing—Original draft, M.Z.; Writing—Review & editing, A.M., A.T., M.Z. If two or more authors are involved in any of the mentioned contributions, the order is alphabetical. All authors have read and agreed to the published version of the manuscript.

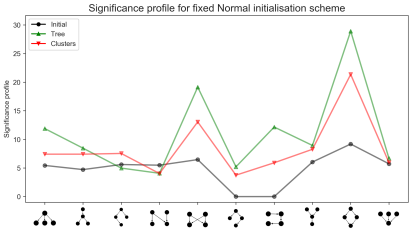
Funding: A.M. acknowledges the support from the University of Padova through "Excellence Project 2018" of the Cariparo Foundation. A.T. acknowledges the support from the University of Padova through the "Deepmath" 2018 STARS Grant.

Conflicts of Interest: The authors declare no conflict of interest.

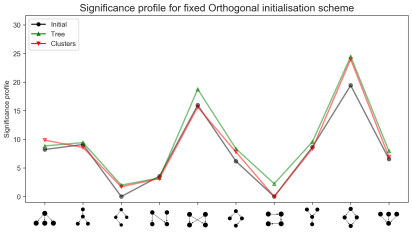
Appendix A. Robustness of Simulations

Appendix A.1. Different Network Architectures for the Synthetic Data Sets

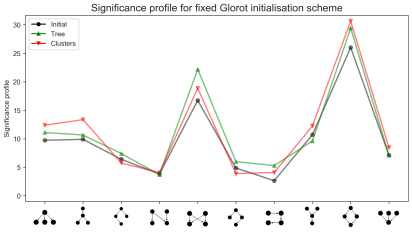
In order to validate the generality of our findings, simulations were repeated by considering different network architectures. Referring to the details displayed in Table 1, three networks with different hidden layers sizes were tested. For reproducibility, the network identifiers represent the seeds used for the simulations. Figures A1 and A2 depict the results presented in the main text referring to all different network architectures. It can be observed that the most significant motif profiles indeed match those discussed in the main text. Figures A3 and A4 depict the largenst significance variations.



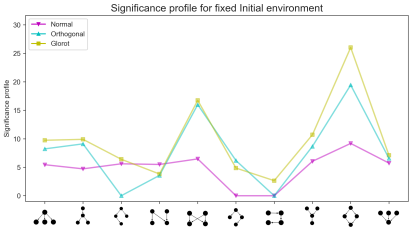
(**a**) Normal initialization scheme.



(**b**) Orthogonal initialization scheme.

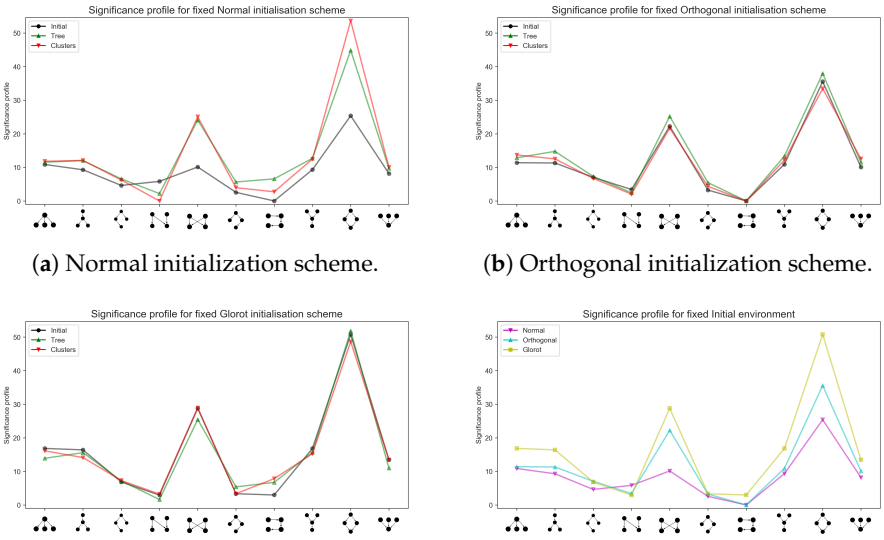


(**c**) Xavier initialization scheme.



(**d**) Initial profile for schemes.

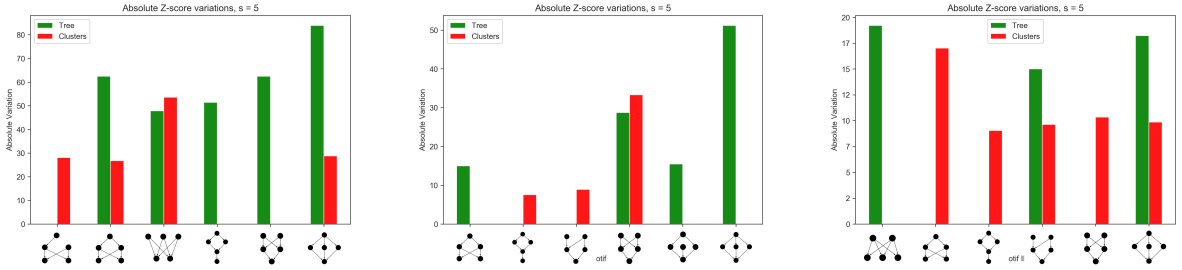
Figure A1. Four-nodes motifs, significance profiles. Seed 250120.



(**a**) Normal initialization scheme. (**b**) Orthogonal initialization scheme.

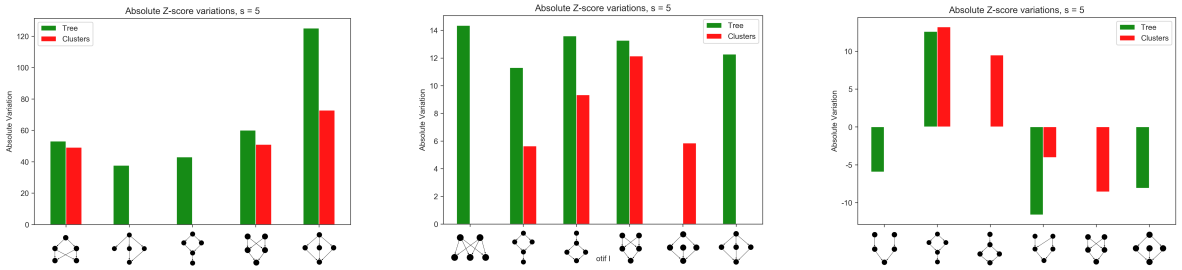
(**c**) Xavier initialization scheme. (**d**) Initial profile for schemes.

Figure A2. Four-nodes motifs, significance profiles. Seed 180112.



(**a**) Normal. (**b**) Orthogonal. (**c**) Xavier.

Figure A3. Largest significance variations for five nodes motifs, seed 250120.



(**a**) Normal. (**b**) Orthogonal matrices. (**c**) Xavier.

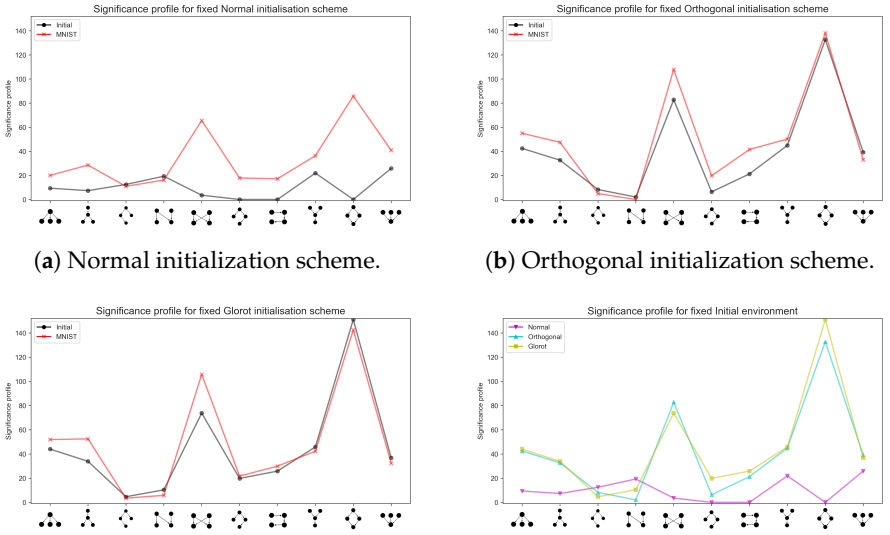
Figure A4. Largest significance variations for five nodes motifs, seed 180112.

Appendix A.2. Training a Larger Network with the MNIST Data Set

A further proof of robustness was achieved by analyzing a deep network trained on a real-world data set. We thus trained a larger network on the popular MNIST data set [32]. The architecture was now composed by a visible layer containing 784 units (the MNIST data set contains images of size 28×28 pixels), three hidden layers of 50, 50 and 20 units respectively and the output layer featuring 10 units (one for each MNIST class). As for the other architectures, the activation function of the hidden units was rectified linear unit (ReLU), while units in the output layer used softmax activation (see Reference [11]).

Quite impressively, these simulations produced results well aligned with those obtained on the artificial data sets, especially in term of the significance profiles for the four-nodes motifs (see Figure A5).

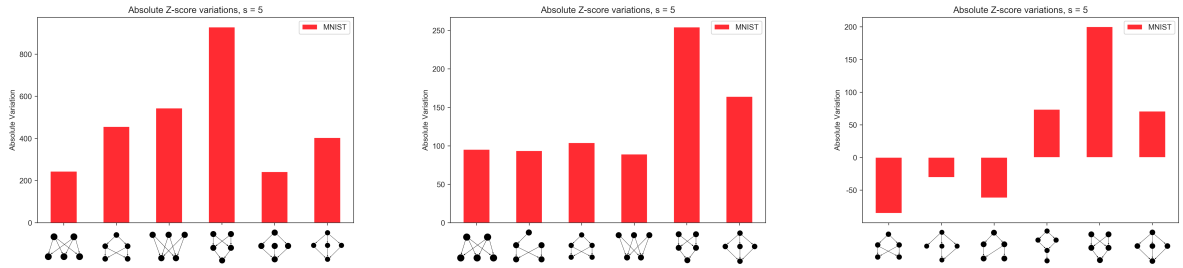
Moreover, the prevalence of the five-nodes *multilayer-perceptron* is clear also in this case, as shown in Figure A6 (recall that this latter may be seen as combination of *diamond* 4-nodes motifs).



(**a**) Normal initialization scheme. (**b**) Orthogonal initialization scheme.

(**c**) Xavier initialization scheme. (**d**) Initial profile for schemes.

Figure A5. Four-nodes motifs, significance profiles. MNIST data set.



(**a**) Normal. (**b**) Orthogonal matrices. (**c**) Xavier.

Figure A6. Largest significance variations for five nodes motifs, MNIST data set.

Appendix B. Data Sets Generation

Data set generation was inspired by previous work [16,43]. A difference is that in these publications, synthetic data consists of categories, and the learning system should guess each item's feature. This leads to a difference in the covariance structure (compare Figure 1 above and Figure 9 in Reference [16]), which is due to the fact that, for example in the binary tree data structure, in the present case correlation patterns tie together all of the nodes in the tree. Each node is associated with a feature (recall, a random variable x_j that is one entry of the random data vector $\boldsymbol{x}$), whilst class labels are assigned according to whether a data item matches some of the previously created data vectors in the case of the binary tree. In the case of independent clusters, the category is assigned according to which one of the independent clusters is selected, see below.

In the real world, data often come as rows of a so-called *design matrix*. Each one datum is then an array of some *features* characterizing the observation. Each one of these features is a random variable, distributed according to some unknown distribution. In this spirit, the data set can be characterised by a multivariate probability distribution. An interesting way to represent multivariate distributions is provided by probabilistic graphical models (PGMs). These models represent the relationships between the random variables contained in a graph—nodes encode random variables, while edges encode the relationships that tie these variables together [11].

Appendix B.1. Binary Tree Data Set

The first example is the binary tree data generating structure. The root node is a random variable, which attains one among the values $\{-1, +1\}$ with equal probability $p = 0.5$. According to the outcome of such random variable, the children inherit the ± 1 value according to some probabilistic decision rule and in the same fashion the children of the children, and so forth down the dynasty, see Algorithm A1. The user sets the depth of the tree D to be created. A data sample is the collection of the $N = 2^D - 1$ random variables that constitute the tree structure. An advantage of the PGM representation is that it renders graphical visualisation ease: Data are often many-dimensional, that is, points in a N-dimensional space.

In this case, the collection of M of such vectors could be thought of as an ensemble of living species. The root node determines whether one item (pattern, data example) can move or not. The children of the root node determine whether if it moves, does it swim? or if it does not move, does it have bark?, and so forth. The levels deeper in the tree structure, bear more information about the data items. In the following, it is shown how the choice of a particular level resolves in the presence of more or fewer classes. As one considers the leaves level, then all of the nodes of a tree (i.e., all of the features in a pattern) must equal, for two items to belong to a given class.

On the other hand, if one considers a shallow level in the tree structure, the nodes which must equal for two data vectors to belong to the same class, are all the nodes up to the last node of the level considered, plus those of the next level. For convenience, the root node lays at level 1. To explain why it is to consider such nodes, the reader should refer to Figure A7. Assume that one wants to gather in a class all the samples that move and swim. Then the sample has to actually move, that is the root node must have the value $+1$, that means that its left child has value $+1$ as well and the right child and its dynasty inherit the -1 value. Then we should consider also the outcome of the stochastic inheritance of the $+1$ value from node 1 to nodes 3 and 4. Based on this trial, we know whether the $+1$ value is attained by node 3 or 4. Node 3 encodes the answer to the question since the sample moves, does it swim? by means of the value $+1$ which means yes. Hence, to say whether two samples belong to the same moving and swimming animals super-class, we should check the equality of the nodes almost up to nodes 3 and 4. In practice, it is easier to check level-wise, thus two samples belong to the same class if all the nodes up to those of the next level match. Next level means next with respect to the level of detail one wants to inspect. In this example, the level is 2. All the subsequent nodes could in principle attain different values but this does not matter. If one wants to differentiate living things based on the fact that such items can move or not, what matters is the value attained by the root node. Then whether two items are respectively a whale or a deer, this does not affect the belongingness to the living thing that can move super-class. In contrast, if one has to differentiate living thing that moves based on the fact that such an item does swim or not, then a further level of detail is needed. Such a finer granularity is encoded by the values the nodes of the next levels attain. If the left children of the root node happen to inherit its $+1$ value, that means that other than being a moving living thing, that item does swim. Therefore, the second tree level encodes this subsequent level of detail. The more detail is embedded (the higher level is chosen), the more the possible classes the data examples may in principle belong to.

The rationale behind such a data generator is first and foremost related to its transparency and statistical structure clarity: There is no real-world consistency in such data, but in this fashion, it is easy to perform classification on them. As explained below, one single pattern generation happens to be a value diffusion down to the tree branches. In this way, one ends up with a N-dimensional binary array, in which many of the slots bear the -1 value. The $+1$ values, on the other hand, lays in correspondence of the slots associated with those nodes which happen to represent a positive answer to the distinction question associated with that node. Consistently with the discussed example: If the living thing encoded in such a $N = 15$ dimensional vector is a moving thing (roughly speaking, an animal), then the root node has the $+1$ value, which in turn means that the 0th slot in the data vector has such value. If this is a water animal, it swims, then the left child of the root node has inherited the

$+1$ value, then the slot 1 in the data vector has the value $+1$ and it implies that the right child of root inherited the value -1, so the slot 2 of the data vector has the value -1. Assume further that other than swimming, this animal *is not a mammal*. Then the left child of the 3-labelled node has inherited the -1 value and this same value is found in slot 7 of the data vector. It means that the $+1$ value is inherited by the right child of node 3, then in the final data vector the $+1$ value appears in slot 8.

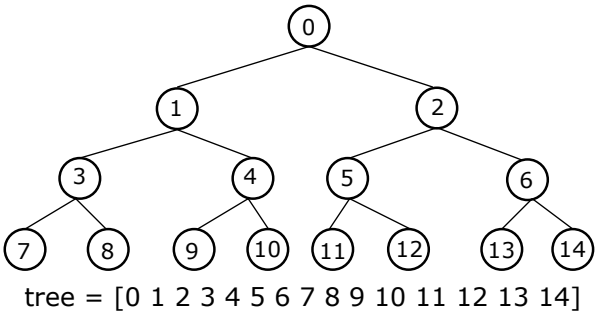


Figure A7. Binary tree data generating structure. Note that the tree data structure is efficiently and easily represented computationally as a linear array. The left and right children of a given node i are $2i + 1$ and $2i + 2$ respectively, with $i = 0, \ldots, N - 1$.

At the end of the day, the final data vector is made up by -1s, except for these said slots, where the $+1$ value ended up in, encoding the positive outcome of those criteria associated with the respective nodes. The terminal (leaves) level could be imagined as the one-hots stratum, that is: all of the leaves attain the -1 value, except for one single leaf, where the $+1$ got to settle, as a consequence of the (stochastic) outcome of all the aforementioned decisions. This lonely $+1$ determines the final category in which the data vector fits in, as one sets the leaves level to be the distinction granularity. In such a case, for two vectors to belong to the same class, it must be that all of the features equal. Otherwise, it could in principle be that a whale, echoing the previously discussed example, has a positive value for the root node, but in another data row, it could be negative. This would mean that a whale is a not moving living being that swims. So, in the label generation stage, one shall differentiate according to all the nodes of the level under consideration and all of their ancestries.

Appendix B.1.1. Single Pattern Generation

One pattern is the collection of *all* the node values of the array-represented tree (that entity formerly dubbed a data vector). As an example, to the non-leaves nodes are associated decision rules, intended to discriminate samples (e.g.,: does the object move?, which can be answered with yes or no, ± 1, is the primal decision rule, that is, axis along which one can set distinctions). The initial value of the root node is inherited and eventually flipped according to probabilistic decision rules with respect to a fixed probabilistic threshold, ε.

In this spirit, referring again to Figure A7, the (non-leaves) nodes ranging from 0 to 6 encode decision rules, (leaves) nodes indexed with $i = 7, \ldots 14$ represent the final details about a sample which are not important for the sake of its classification. The following criteria are implemented:

(1) The probabilistic threshold is fixed a priori. The smaller its value, the less variability in the data set.
(2) Root attains the values ± 1 with probability $p = 0.5$.
(3) Root's children attain values $+1$ or -1 in a mutually exclusive fashion. The following convention is adopted: *if the root node attains the value* $+1$*, then the left child inherits the same value. Else, the left child attains the value* -1 *and the right child has assigned the value* $+1$.

(4) From the third level (children of root's children), the progeny of any node that has value -1 also has to have -1 value. On the other hand, if one node has value $+1$, its value is inherited (again mutually exclusively) by its children according to a probabilistic decision rule.

The aforementioned probabilistic decision rule is a Metropolis-like criterion: Sample a random variable $p \sim U([0,1])$, then, given the probabilistic threshold ε,

- If $p > \varepsilon$, the left child inherits the $+1$ value, and the right child, alongside with its progeny, assume the opposite value;
- Else, is it the right child to assume the value $+1$.

Appendix B.1.2. Complete Data Set

Repeating the above procedure M times, one ends up with a data matrix $\boldsymbol{X} \in \{-1,+1\}^{M\times N}$, that is, each row of $\boldsymbol{X}$, $\boldsymbol{x}^{\mu}$, $\mu = 1,\ldots,M$, is one single N-dimensional data vector, in the same terminology as above, that is a N-featured data vector (one pattern).

To complete the creation of a synthetic set of data, one needs the label associated with each one of the data items. Here the choice of the probabilistic threshold ϵ turns out to be crucial. The higher this quantity, the more the total number of different classes the data example may fall into. On the other hand, if ϵ is small enough, there is a low probability of flipping a feature value, then it is more likely to observe repeatedly the same configuration.

To create the labels, encoded as one-hot activation vectors, one arbitrarily assumes the identity matrix to be the labels matrix. Then the whole data set is explored in a row-wise fashion. Since the data set has a hierarchical structure, it is possible to select the granularity of the distinction made in order to differentiate patterns in different classes. It depends on the choice of a level in the binary tree: If the level chosen is high (far away from the root node) then one ends up with a fine-grained distinction. On the other hand, if the chosen level is low, the distinction is made according to super-classes, for example, whether a given object can move. The finer the granularity, the more detailed the distinction between patterns. Obviously, in this latter case, the data set exhibits a greater number of distinct classes. See the discussion above.

By this observation, the label matrix is created according to the level of distinction chosen. The node values to be considered (i.e., the entries of each $\boldsymbol{x}$ data vector) are all those that encode the values of the nodes up to the last one of the level selected. Referring again to the tree in Figure A7, if it suffices to identify the move or not distinction, then one could safely check both the root node only or the root node with its children, that is nodes $\{0,1,2\}$, because the inheritance from the root node to its children is conventionally based on the root value solely. But if one wants to consider whether an object can move alongside with the further if it moves, does it swim? and if it does not move, does it have bark? distinctions, then one should consider also the children nodes of node 1, that is the answer to the decision rule asked by node 1. Hence to determine whether two data items fall in that same category, we check that all the first $2^{L+1}-1$ nodes have the same value. Here $L = 2$, in fact we consider nodes $i \in [0, 2^{L+1}-1] \equiv [0,7] = \{0,1,2,\ldots,6\}$.

By thus doing the data set is generated. The matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ are saved to a proper data structure which can be easily managed by the program that implements the artificial neural network described in the main text.

Algorithm A1 Binary tree. Single feature generation

Compute $N = N_{\text{leaves}}$, $n = N_{\text{not leaves}}$. M is a free parameter

`tree` = $\mathbf{0}^N$

Define a small $\varepsilon \sim O(10^{-1})$ as probabilistic threshold

Value of root $\eta^{(0)} \sim U(\{-1, +1\})$

if Root node has value $+1$ **then**

    The left child inherits the value $+1$

    And the right child inherits the value -1

else

    The left child inherits the value -1

    And the right child inherits the value $+1$

end if

for All the other nodes indexed $i = 1, \ldots, n$ **do**

    if Node i has $+1$ value **then**

        Sample $p \sim U([0, 1])$

        if $p > \varepsilon$ **then**

            Left child of i = $+1$; Right child of i = -1

        else

            Left child of i = -1; Right child of i = $+1$

        end if

    else

        Both the children of i inherit its -1 value

    end if

end for

$\boldsymbol{x}^\mu \leftarrow$ values generated, $\quad \mu = 1, \ldots, M$

Algorithm A2 Binary tree. *One-hot* activation vectors, that is, labels

Choose level of distinction L

$\boldsymbol{Y} = \mathbb{I}$

for $\mu = 1, \ldots, M$ **do**

    for $\nu = i, \ldots, M$ **do**

        if the first $2^{L+1} - 2$ entries of $\boldsymbol{x}^\mu$ and $\boldsymbol{x}^\nu$ equal **then**

            $\boldsymbol{y}^\nu \leftarrow \boldsymbol{y}^\mu$

        end if

    end for

end for

for $i = 1, \ldots, N$ **do**

    if $\boldsymbol{Y}[:, i]$ equals $\mathbf{0}^N$ **then**

        Eliminate column i of $\boldsymbol{Y}$

    end if

end for

Appendix B.2. Independent Clusters Data Set

The generation of the second data set is performed as follows: Generating some cloud of points distributed according to a bivariate Gaussian distribution, with means spread apart and covariances sufficiently small, in such a way that the points of different groups do not overlap with the others. The 2-dimensionality has, of course, nothing to do with the number of features, which as said before is

the total number of points generated, that is the nodes of the probabilistic graph representation. This 2-dimensionality serves solely to draw the PGM and subsequently to partition the graph.

Once points are generated, are turned in a fully connected graph, that is, create edges between each pair of nodes. In the spirit of the simulated annealing algorithm, here it is imagined that such a fully connected graph is a sort of mineral structure, and it is to increase the temperature, to simulate a melting process that destroys some of the over-abundant edges, according to some metric, for example, the distance between points. For this reason, it comes handy the 2-dimensional representation: Distance is simply the norm of the vector from a node to another. The distance for which the edge is removed is temperature-dependent: the higher the temperature, the shorter the maximum edge length allowed. At the end of this simulated melting process, it is expected the graph to exhibit some independent components, provided the melting schedule is properly set. Moreover, these independent groups are not fully connected within themselves. The melting schedule is designed in a way to remove some of these intra-edges. This simulates the random variables of each group not to be dependent on all of the others in the same could. Note that, unlike how exposed in Reference [19], in this melting simulation there is not, strictly speaking, a optimization perspective inasmuch what matters is the removal of some edges. The physics of the procedure could be improved.

Algorithm A3 Independent clusters. Simulated melting to *partition the graph*

Choose the number of classes N_C

Set $\boldsymbol{\mu}^{(k)} \in \mathbb{R}^2$, $\boldsymbol{\Sigma}^{(k)} \in \mathbb{R}^{2\times 2}$, $k = 1, \ldots, N_C$

Generate $\boldsymbol{X}$ s.t. $\boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{\mu}^{(k)}, \boldsymbol{\Sigma}^{(k)})$, $i = 1, \ldots, M$

Include the indexes of the points generate in a list, which is the set of the vertices $\mathcal{V}$ of the graph $\mathcal{G}$

Fully connect the vertices to form a fully connected graph and group the vertices and the set of the edges $\mathcal{E}$ in the graph data structure, $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$.

Note that since 2-dimensional coordinates will be useful, $\mathcal{V}$ is a dictionary of keys (nodes indexes $i = 1, \ldots, M$) and values (list with the point coordinates, $(x_i^{(1)}, x_i^{(2)})$).

for T increasing **do**

    for All the edges $e = 1, \ldots, |\mathcal{E}|$ **do**

        if Length of edge $e > \frac{1}{T}$ (for example) **then**

            Remove edge e

        end if

    end for

end for

Plot the remaining edges and check if only independent fully connected components have survived.

Algorithm A4 Independent clusters. Single pattern generation

Here i indexes a single random variable. This kernel is used as many times as the number of samples the user wants to generate. $\boldsymbol{x}$ is the whole data item, initialised with each slot set to -1. **Note**: in the data set actually generated, the value of the nodes are set to their topological orders, with no ancestral sampling implemented.

Set $\boldsymbol{x} = \{-1\}^N$

Sample $L \sim \mathcal{U}(\{1, \ldots, N_c\})$

for all the vertices $i = 1, \ldots, n_k$ in cluster L **do**

  if Topological Order of i is 1 **then**

    $x_i \sim p(x_i) \sim \mathcal{N}(0, k_i^{-2})$

  else

    $x_i \sim p(x_i) \prod_{j \in \text{Ancestors}(x_i)} (4\pi k_j^2)^{-1/2} \exp\left(-\frac{1}{2}\frac{x_j^2}{k_j^2}\right)$

  end if

end for

$\boldsymbol{y}_i = \text{one-hot}(L)$

Appendix B.2.1. Single Pattern Generation

Once the independent clusters come to form, it is to assign each of the nodes a topological ordering in such a way to perform the ancestral sampling [11]. Since the graphs are directed, in the edges data structure created each edge is in the form of a couple (i, j), that is, edge from node i to node j. Then if one node appears only on the left slot of such representation, it has topological order 1, in that no edge ends up at that node. Conversely, each node that appears on the right has almost one ancestor. For each edge then, each right node is saved to a proper data structure, and it is kept track of the ancestors of each node. In this way, it is possible to assign both the topological order and to keep a list of all the ancestors. It will be useful in the stage of sampling to dispose of such a list.

As a zero model however it is done as follows: A data item is initially initialized with all the features values of -1. Since each vertex in the graph encodes a feature, and the belongingness of each vertex to a group is a piece of information known from the points generation stage, an integer ranging from 1 to the number of classes $N_c = 4$ is sampled uniformly. The nodes corresponding to this label number are assigned different values, according to their topological order. This is trivial to do since for each vertex belonging to the selected group one simply puts in the corresponding slots in the data vector the topological order of such vertices.

A further improvement could be rather this approach: once a label is sampled, one could sample from the distribution $p(x_i)$, for the vertices with topological order 1 in that cluster. The values associated with nodes having topological order 2 is still sampled from that distribution, but must be conditioned to the values sampled for their ancestors (nodes of order 1), that is, $p(x_i \mid \text{Ancestors}(x_i))$. This is explained by recalling the very purpose of graphical models: to show (even graphically) the *causality* of the random variables involved. The distribution could be chosen to be a Gaussian with mean zero and variance proportional to the degree of that node. Gaussian is believed to fit since nearby features are expected to have similar values ([43], but differently from this work, here one does not generate the features vector, hence sampling from the multivariate Gaussian having zero mean and variance dependent on the inverse of the Laplacian matrix of the graph. Here it suffices to sample one value for a single node, and hence the degree of a node could be a good compromise, being such quantity one of the ingredients of the Laplacian).

To sample from the conditional $p(x_i \mid \text{Ancestors}(x_i))$ the following rationale may be implemented: The distribution is referred to all the nodes up to i, then could be viewed as a multivariate distribution. Then a value is sampled from that multivariate distribution, but keeping constants the values of the

random variables sampled yet. As an example: Assume that node 3 of cluster 1 is to be assigned the value x_3 and that Ancestors$(x_3) = [1, 2]$. Then the distribution to sample from is

$$p(x_3 \mid x_1, x_2) \sim \exp\left(-\frac{1}{2}(x_1, x_2, x_3)^T \, \mathbf{\Sigma}^{-1} \, (x_1, x_2, x_3)\right) \tag{A1}$$

with $\mathbf{\Sigma} = \text{diag}(k_i)$, $i = 1, \ldots, 3$, being k_i the degree of node i. The above formula may be broken in products, owing to the fact that the variance matrix is diagonal, that is

$$\begin{aligned} p(x_3 \mid x_1, x_2) &\sim \exp\left(-\frac{1}{2}\frac{x_1^2}{k_1^2}\right) \exp\left(-\frac{1}{2}\frac{x_2^2}{k_2^2}\right) X_3 \\ X_3 &\sim \mathcal{N}\left(0, k_3^{-2}\right) \end{aligned} \tag{A2}$$

the first two factors being the values that the Gaussian probability density function attains at the values sampled for the ancestors x_1 and x_2 and the third factor is the value sampled from the Gaussian having zero mean and variance k_3^2.

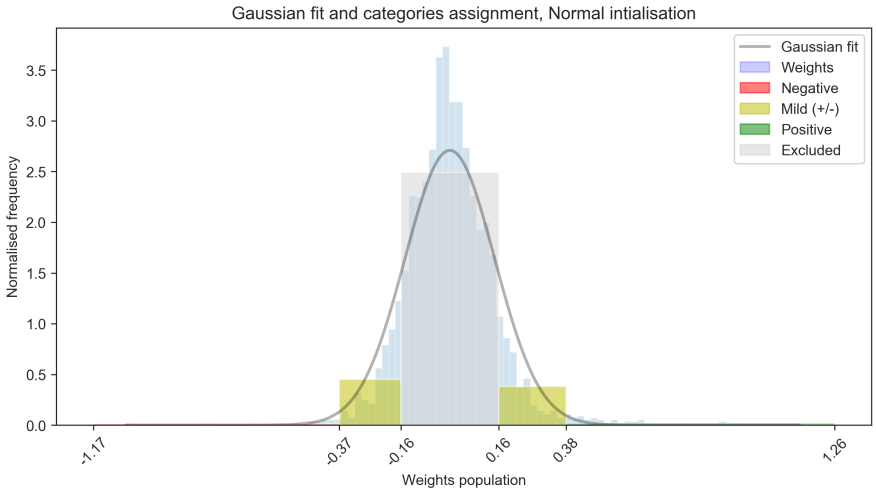
Appendix B.2.2. Complete Data Set

This procedure is repeated many times as specified by the user. Here a good number is, as in the case of the binary tree, $M = 2000$ items. In the complete data set hence one has features in which the only values not being -1 lay in correspondence of the indexes of the data array that match with the nodes of the graph that belongs to the category given by the label of that feature. Labels are again one-hot vectors. For example, assume that the first cluster is selected. If this first cluster comprises the vertices ranging from 1 to 5, where node 1 has order 1, 2 and 3 have order 2, 4 has order 3 and five has order 4, then that data item has values $[1, 2, 2, 3, 4, -1, \ldots, -1]$ and the corresponding label is $[1, 0, \ldots, 0]$.

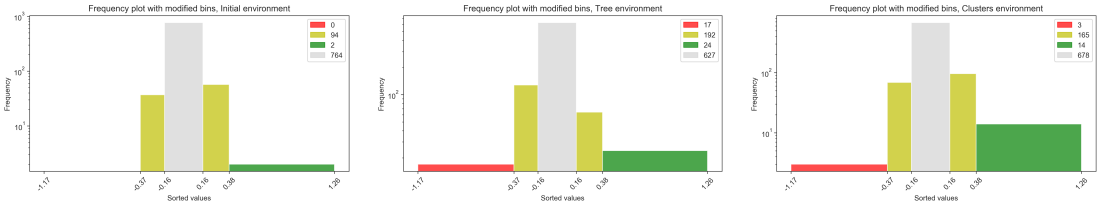
Appendix C. Pre-Processing

Once the model is trained, it is necessary to pre-process the parameters data structures to subsequently use the motif detection tools. The FANMOD (FAst Network MOtif Detection) software has been utilised. In the case of unweighted networks, one simply sets all the connection strengths to 1 (those that fall in the non-zero category, see below), contrarily, weights must be discretised, in that motif mining software deal with colored networks, which means that edges tags must be categorical. The categories supported by the software are limited (maximum seven).

The choice of the weights to retain to search the network motifs is indeed a crucial aspect. Assume that all the connection weights of the network are gathered. A Gaussian distribution is fitted on such population, which we could call $E = \{w_i\}_{i=1}^n$, where n is the number of edges in the graph. As a zero hypothesis, it is thought to fit. Due to the initialization schemes adopted, the mean is aligned to zero. Thus one must identify a region in the neighborhood of zero that comprehends all those weights that might be redundant. Refer to Figure A8. Such an exclusion region could be identified by choosing two boundary values in the weights population, symmetrically placed from the mean (zero). Call p the Gaussian density function fitted to the weights population and c the weights cut-off threshold value. The values in $W_0 = \{w \in E : p(w) \geq c \max_w p(w)\}$ identify this exclusion region. Call $w_0^+ = \max_w W_0$ and $w_0^- = \min_w W_0$ the minimum and maximum values of the set W_0, respectively. Of course the parameter c plays an important role. The larger c, the smaller the exclusion region. If $c \simeq 1$, then the neighborhood of zero selected is negligible, then almost all the weights are retained. Note that this process is thought for distributions which exhibit a major weights density centered around zero. The case of a learning process that pushes the distribution mean far from zero should be treated in a different way. The vast majority of the weights around the average values should in that case be retained, excluding those in the neighborhood of zero. However, being the weights choice heuristics-reliant, it is arbitrary, and of course susceptible of improvements.



(**a**) Gaussian fit of the entire weights population.



(**b**) Initial configuration. (**c**) Tree data set. (**d**) Clusters data set.

Figure A8. Gaussian fit of the weights population and the respective subdivision histograms. Results refer to the normal initalization scheme, network 240120.

A further step is required: The tails identified, that is the support of the Gaussian curve disjoint from the exclusion zone (bounded by the maximum and minimum values of the weights in the population E), must be further sectioned to identify one more category of weights values. The first category has been isolated yet, that is the small weights that are excluded. The identification of such region implies that there are *positive* and *negative* weights. Recall that connection strengths are real-value quantities but motifs mining software support a limited number of categories in which network edges are classified. Hence it is thought appropriated to define almost a third category: *mildly positive and negative* values. Due to the shape of the Gaussian itself, it is clear that "outermost" weight values are more rare, hence the tails are unevenly divided. Call $w^+ = \max_w E$ and $w^- = \min_w E$ the minimum and maximum of the weights population respectively. Then define the distances $d^+ = |w^+ - w_0^+|$ and $d^- = |w^- - w_0^-|$, and choose a value k that defines the boundaries of the *mild* values sets. The mildly positive weights are those that fall in the interval $\left(w_0^+, k\,d^+\right]$. Analogously the mildly negative values are those that fall in $\left[w^-, -k\,d^-\right)$. Mildly positive and negative weights are gathered in the same group. The remainder of the weights is categorized as positive and negative, according to whether they are on the right or left side of the mild weights sets boundaries. Refer again to Figure A8 for a graphical explanation.

It is clear that also the choice of k is crucial. A sensible value of this parameter should be small so that the positive and mildly positive values are more evenly distributed. The value chosen to produce the results presented in the main text is $k = 1/5$. The smaller this value, the lesser the mild weights.

Histograms in Figure A8b–d give a graphical explanation of this heuristic: The central bin, the most populated, is not taken into account in the subsequent analysis. The extremal tails are the positive and negative edges strengths values. The two intermediate bins, between the central one and the extremal ones, are those set to be the neutral weights.

In this way, the motifs featuring values falling in the central bin can be preventively discarded in further analyses in that they are not thought to be relevant. One has at this point the graph representations amenable to the motifs mining program. Whether colors shall be accounted for or not, it absorbs the input file formatted according to the above-discussed partition and performs the analyses. If the analysis to be carried out is on an unweighted network, then all the connection strengths are set equal to 1, except those related to the central bin items, which are not accounted (no connection).

References

1. Newman, M. *Networks: An Introduction*; Oxford University Press, Inc.: New York, NY, USA, 2010.
2. Strogatz, S.H. Exploring complex networks. *Nature* **2001**, *410*, 268–276. [CrossRef]
3. Caldarelli, G. *Complex Networks*; EOLSS Publications: Abu Dhabi, UAE, 2010.
4. Newman, M.E.; Barabasi, A.L.; Watts, D.J. *The Structure and Dynamics of Networks: (Princeton Studies in Complexity)*; Princeton University Press: Princeton, NJ, USA, 2006.
5. Latora, V.; Nicosia, V.; Russo, G. *Complex Networks: Principles, Methods and Applications*; Cambridge University Press: Cambridge, UK, 2017.
6. Milo, R.; Shen-Orr, S.; Itzkovitz, S.; Kashtan, N.; Chklovskii, D.; Alon, U. Network Motifs: Simple Building Blocks of Complex Networks. *Science* **2002**, *298*, 824–827. [CrossRef] [PubMed]
7. Lenski, R.E.; Ofria, C.; Pennock, R.T.; Adami, C. The evolutionary origin of complex features. *Nature* **2003**, *423*, 139. [CrossRef] [PubMed]
8. Vespignani, A. Evolution thinks modular. *Nat. Genet.* **2003**, *35*, 118–119. [CrossRef] [PubMed]
9. Alon, U. *An Introduction to Systems Biology: Design Principles of Biological Circuits*; Chapman & Hall/CRC Mathematical and Computational Biology, Taylor & Francis: London, UK, 2006; pp. 27–30, 106–115.
10. LeCun, Y.; Bengio, Y.; Hinton, G.E. Deep learning. *Nature* **2015**, *521*. [CrossRef] [PubMed]
11. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016. Available online: http://www.deeplearningbook.org (accessed on 16 December 2019).
12. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
13. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. *Adv. Neural Inf. Process. Syst.* **2014**, *27*, 3104–3112.
14. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484. [CrossRef]
15. Montavon, G.; Samek, W.; Müller, K.R. Methods for interpreting and understanding deep neural networks. *Digit. Signal Process.* **2018**, *73*, 1–15. [CrossRef]
16. Saxe, A.M.; McClelland, J.L.; Ganguli, S. A mathematical theory of semantic development in deep neural networks. *Proc. Natl. Acad. Sci. USA* **2019**, *116*, 11537–11546. [CrossRef]
17. Testolin, A.; Piccolini, M.; Suweis, S. Deep learning systems as complex networks. *J. Complex Netw.* **2018**, *521*. [CrossRef]
18. Testolin, A.; Zorzi, M. Probabilistic models and generative neural networks: Towards an unified framework for modeling normal and impaired neurocognitive functions. *Front. Comput. Neurosci.* **2016**, *10*, 73. [CrossRef] [PubMed]
19. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by Simulated Annealing. *Science* **1983**, *220*, 671–680. [CrossRef] [PubMed]
20. Saxe, A.; McClelland, J.; Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In Proceedings of the International Conference on Learning Representations, Banff, AB, Canada, 14–16 April 2014.
21. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; pp. 249–256.
22. Wernicke, S.; Rasche, F. FANMOD: A tool for fast network motif detection. *Bioinformatics* **2006**, *22*, 1152–1153. [CrossRef]

23. Wernicke, S. Efficient Detection of Network Motifs. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2006**, *3*, 347–359. [CrossRef] [PubMed]
24. Masoudi-Nejad, A.; Schreiber, F.; Kashani, Z.R. Building blocks of biological networks: A review on major network motif discovery algorithms. *IET Syst. Biol.* **2012**, *6*, 164–174. [CrossRef]
25. Alon, U. Network motifs: Theory and experimental approaches. *Nat. Rev. Genet.* **2007**, *8*. [CrossRef]
26. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
27. Milo, R.; Itzkovitz, S.; Kashtan, N.; Levitt, R.; Shen-Orr, S.; Ayzenshtat, I.; Sheffer, M.; Alon, U. Superfamilies of Evolved and Designed Networks. *Science* **2004**, *303*, 1538–1542. [CrossRef]
28. Wuchty, S.; Oltvai, Z.N.; Barabási, A.L. Evolutionary conservation of motif constituents in the yeast protein interaction network. *Nat. Genet.* **2003**, *35*, 176–179. [CrossRef]
29. Salakhutdinov, R.; Hinton, G. Deep boltzmann machines. In *Artificial Intelligence and Statistics*; van Dyk, D., Welling, M., Eds.; PMLR: Clearwater, FL, USA, 2009; pp. 448–455.
30. Zorzi, M.; Testolin, A.; Stoianov, I.P. Modeling language and cognition with deep unsupervised learning: A tutorial overview. *Front. Psychol.* **2013**, *4*, 515. [CrossRef]
31. Testolin, A.; Stoianov, I.; Sperduti, A.; Zorzi, M. Learning orthographic structure with sequential generative neural networks. *Cogn. Sci.* **2016**, *40*, 579–606. [CrossRef] [PubMed]
32. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
33. Voulodimos, A.; Doulamis, N.; Doulamis, A.; Protopapadakis, E. Deep Learning for Computer Vision: A Brief Review. *Comput. Intell. Neurosci.* **2018**, *2018*, 7068349. [CrossRef] [PubMed]
34. Hochreiter, S.; Schmidhuber, J. Long Short-term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
35. Piperno, A. Isomorphism Test for Digraphs with Weighted Edges. In *Proceedings of the 17th International Symposium on Experimental Algorithms (SEA 2018)*; Leibniz International Proceedings in Informatics (LIPIcs); Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik: Dagstuhl, Germany, 2018; Volume 103, pp. 30:1–30:13. [CrossRef]
36. McKay, B.D.; Piperno, A. Practical graph isomorphism, II. *J. Symb. Comput.* **2014**, *60*, 94–112. [CrossRef]
37. Raina, R.; Madhavan, A.; Ng, A.Y. Large-scale deep unsupervised learning using graphics processors. In Proceedings of the 26th Annual International Conference on Machine Learning, Montreal, QC, Canada, 14–18 June 2009; pp. 873–880.
38. Testolin, A.; Stoianov, I.; De Filippo De Grazia, M.; Zorzi, M. Deep unsupervised learning on a desktop PC: A primer for cognitive scientists. *Front. Psychol.* **2013**, *4*, 251. [CrossRef]
39. Newman, M.E.J. Modularity and community structure in networks. *Proc. Natl. Acad. Sci. USA* **2006**, *103*, 8577–8582. [CrossRef]
40. Newman, M.E. The structure and function of complex networks. *SIAM Rev.* **2003**, *45*, 167–256. [CrossRef]
41. Choobdar, S.; Ribeiro, P.; Silva, F. Motif Mining in Weighted Networks. In Proceedings of the 12nd IEEE ICDM Workshop on Data Mining in Networks, Brussels, Belgium, 10 December 2012; pp. 210–217. [CrossRef]
42. Onnela, J.P.; Saramäki, J.; Kertész, J.; Kaski, K. Intensity and coherence of motifs in weighted complex networks. *Phys. Rev. E* **2005**, *71*. [CrossRef]
43. Kemp, C.; Tenenbaum, J.B. The discovery of structural form. *Proc. Natl. Acad. Sci. USA* **2008**, *105*, 10687–10692. [CrossRef]