Neural network response in learning from synthetic data and *network motifs* formation

Contents

1 Introduction, aim and scope **3**

2 Data sets **3**
2.1 Binary tree data set 3
2.2 Independent clusters data set 4
2.2.1 Single pattern generation 6
2.2.2 Complete data set 8

3 Neural network model **8**

4 Results **10**
4.1 Binary tree data set 10
4.2 Independent clusters data set 11
4.2.1 Modularly varying goals 11
4.3 Motifs detection 11

5 Conclusions and remarks **11**
5.1 Further improvements 13

General scope information

Code and documentation available here: GitHub repo.
Main references:

(1) Kashtan, N., Alon, U., *Spontaneous evolution of modularity and network motifs*, 2005. PNAS.

(2) Kashtan, N., Itzkovitz, S., Milo, R., Alon, U., *Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs*, 2004. NCBI.

(3) Kemp, C., Tenenbaum, J.B., *The discovery of structural form*, 2008. PNAS.

(4) Saxe, A.M., McClelland, J.L., Ganguli, S., *A mathematical theory of semantic development in deep neural networks*, 2018. arXiv.

(5) Saxe, A.M., McClelland, J.L., Ganguli, S., *Exact solutions to the nonlinear dynamics of learning in deep linear neural networks*, 2014. arXiv

(6) Goodfellow, I., Bengio, Y., Courville, A., *Deep Learning*, 2015. Online book.

(7) Newman, M.E.J., *Fast algorithm for detecting community structure in networks*, 2003. arXiv.

(8) Newman, M.E.J., Girvan, M., *Finding and evaluating community structure in networks*, 2003. arXiv.

(9) Kirkpatrick S., Gelatt, C.D., Vecchi, M.P., *Optimization by simulated annealing*, 1983. Science

Most of the figures, grouping different plots, are provided at the end of the document, in order not to make the text body heavy to read.

What's new

A new data set has been introduced in this version. Now the neural system is trained both on the binary tree data set and the independent clusters data set. The creation of this latter underlies some physical resemblance and embeds the use of Probabilistic Graphical Models. All is presented below.

At the present stage the `mfinder1.2` program is still used, but it is in order to develop from scratch a new method that finds features of the evolved model which fits better in this present framework: With respect to previous work on network motifs (Kashtan and Alon, 2005 and Kashtan *et al.*, 2004) here one can not neglect the connection strengths of the neural network once it is trained.

1 Introduction, aim and scope

Two data sets are fed to the neural system. These are generated with the purpose to embody different statistical structures. This particular feature has been put in relationship with the dynamics of learning in neural networks, shedding light on the speed of learning semantic distinctions as a function of the singular values of the input-output covariances (Saxe *et al.*, 2018). This proves the relevance of input statistical signature, but nothing is said about the emergence of particular topological patterns in the connections between neurons in the neural network.

Thus the task here is to inspect the topology of the neural system in response to input data sets with sharply different statistical morphology.

2 Data sets

Some previous work is followed for the zero stage (Kemp and Tenenbaum, 2008 and Saxe *et al.*, 2018). A difference however is that in these cited publications is that there synthetic data consists in categories, and the learning system should guess each item's feature. This leads to a difference in the covariance structure (cfr. Figures 3 below and for example Figure 9 in Saxe *et al.*, 2018), and is due to the fact that, for example in the binary tree data structure, in the present case correlation patterns tie together, in some extent, all of the nodes in the binary tree.

The main difference is that here each node of the PGM generated is associated with a *feature*, whilst class labels are assigned according to whether a data item matches some of the previously created, in the case of the binary tree, and according to which one of the independent clusters is selected in the case of the second data set.

2.1 Binary tree data set

The binary tree data set is intended to embed a **hierarchical structure**. The root node encodes a differentiation rule (i.e. whether the given data object *can move* or not). The left child encodes a further rule, that renders the description of such data item more detailed, that is whether a moving specie *can swim* or not. Analogously the right child is associated with the rule that sets whether a non-moving object *does have the bark* or not. In this fashion, all the nodes values being collected in a $N = 2^D - 1$ dimensional array with values among $\{-1, +1\}$ according to the outcome of each decision rule and being D the depth of the tree, a hierarchical order is given to the data set.

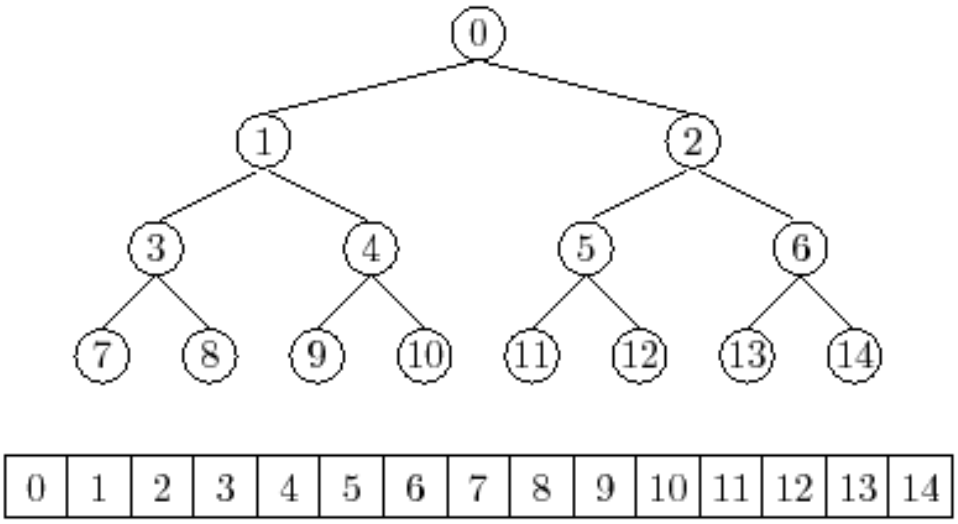


Figure 1: From OpenDataStructure site.

See Figure 1 for a graphical example: node 0 is the root node. The branching of every node represents the positivity or negativity of the decision rules associated with each node. One ends up with the leaves values, which encode the belongingness of each data item to a class.

2.2 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.

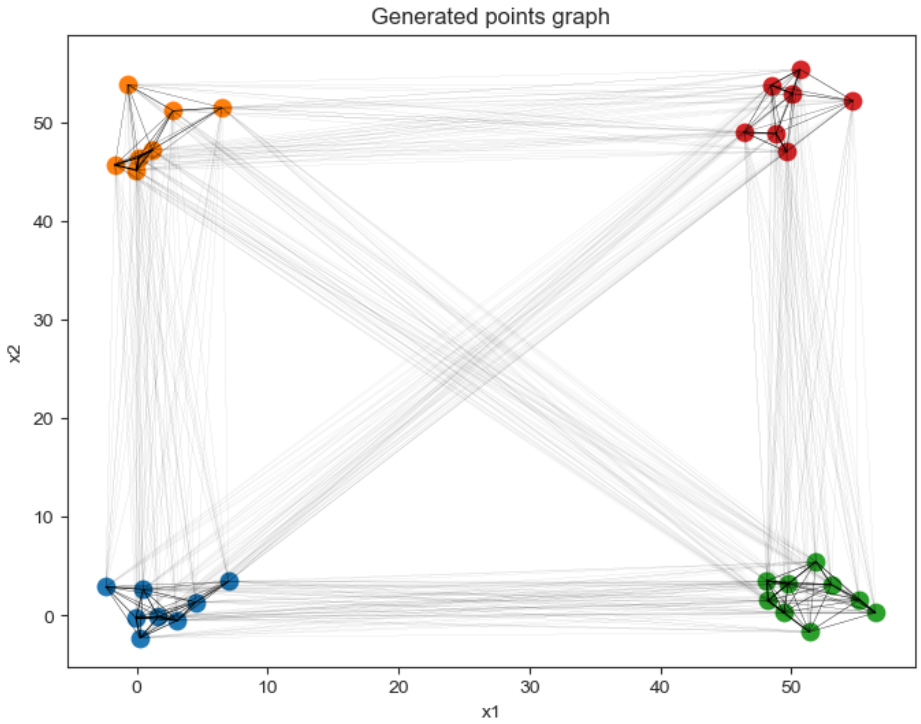
Once points are generated, are turned in a fully connected graph, i.e. create edges between each pair of nodes. In the spirit of the *simulated annealing* algorithm, here it is imagined that such a fully connected graph is a sort of mineral structure, and it is in order to increase the temperature, to 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 shortest 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 Kirkpatrick, *et al.* (1983), in this melting simulation there is not, strictly speaking, an *optimization* perspective inasmuch what matters is the removal of some edges. The physics of the procedure could be revised.

Algorithm 1 Binary tree. Single feature generation

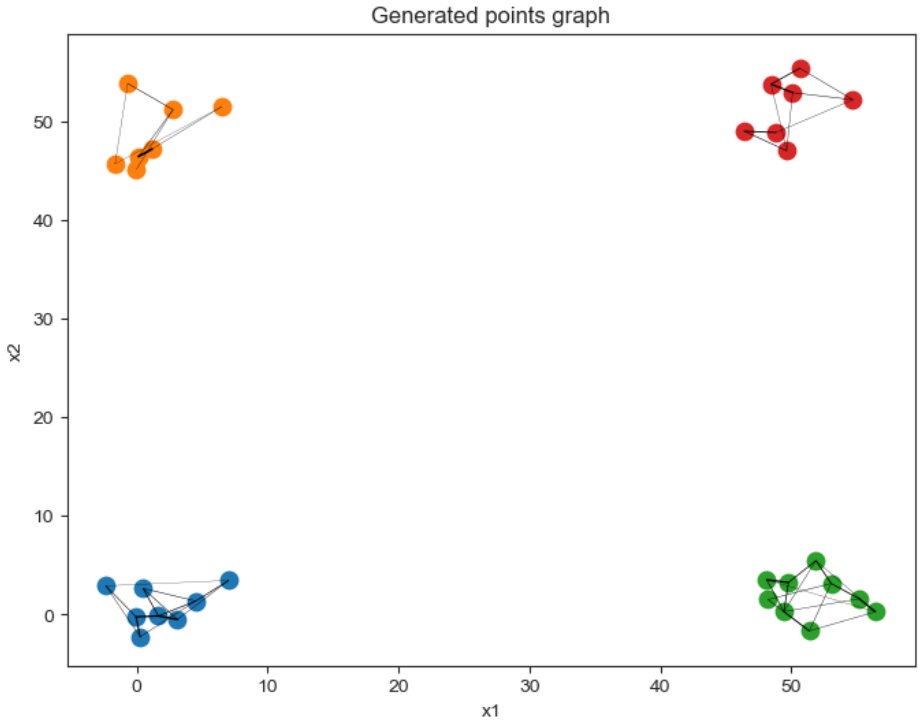
Compute $N = N_{\text{leaves}}$, $n = N_{\text{not leaves}}$. M is a free parameter.
$\texttt{tree} = \mathbf{0}^N$
Define a small $\epsilon \sim O(10^{-1})$ as probabilistic threshold
Value of root $\eta^{(0)} \sim U(\{-1, +1\})$
if Root node has value $+1$ **then**
    The left child inherits the value $+1$
    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 $p > \epsilon$ **then**
            Value of the left child of i = value of i
        else
            Value of the left child of i = flip the value of i
        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 2 Binary tree. *One-hot* activation vectors, i.e. 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



(a) Fully connected graph



(b) *Molten* graph

Figure 2: Two subsequent stages of the clusters data set generation

2.2.1 *Single pattern generation*

Once the independent clusters come to form, it is in order to assign each of the nodes a **topological ordering** in such a way to perform the **ancestral sampling** (Chapter 16

of Goodfellow, *et al.*, 2016). Since the graphs are directed, in the edges data structure created each edge is in the form of a couple (i, j), i.e. 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 appearing 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 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 an 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), i.e. $p(x_i \,|\, \text{Ancestors}(x_i))$. This is explained by recalling the very purpose of graphical models: to show (even graphically) the *causality* of the random variables involved. As distribution it could be chosen 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 (Kemp, 2008, 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 a 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).

The following nomenclatural convention is adopted: vectors are ***bold-italic*** typefaced, such as one single data item. Since all the entries are random variables, in this case one refers to a single entry of the data item vector as x_i, even if it is widespread to typeface random variables with roman capitalisation. So in the following $p(x_i)$ refers to the probability associated with the outcome x_i of the random variable that is encoded by the i-th entry of the data item.

To sample from the conditional $p(x_i \,|\, \text{Ancestors}(x_i))$ 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 $\text{Ancestors}(x_3) = [1, 2]$. Then the pdf to sample from is

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

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

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

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.

2.2.2 *Complete data set*

This procedure is repeated many times as specified by the user. Here a good number is, as in the case of 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 $[1, 0, \ldots, 0]$.

Algorithm 3 Independent clusters. Simulated melting to *partition the graph*

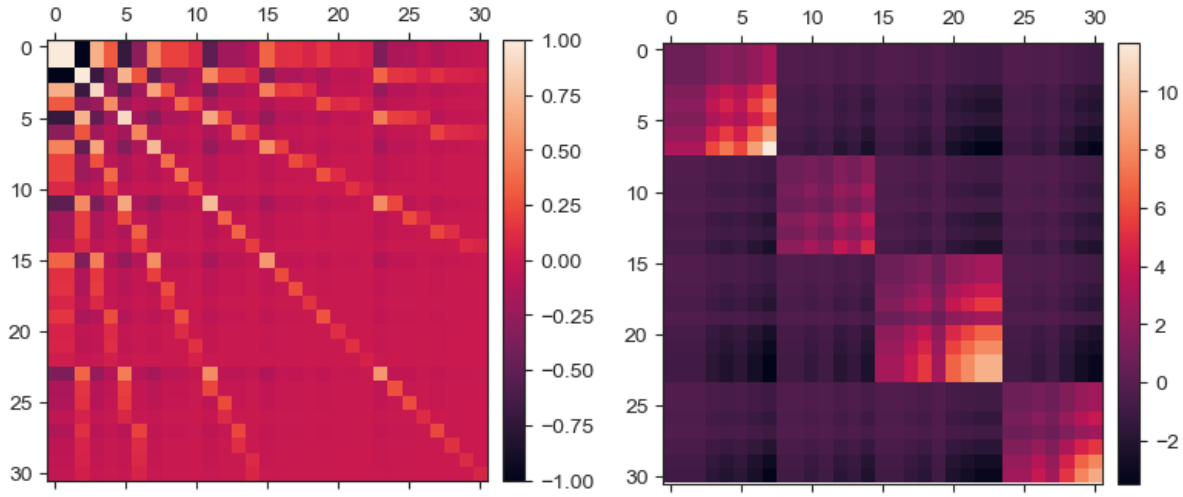
Choose the number of classes N_C
Set $\boldsymbol{\mu}^{(k)} \in \mathbb{R}^2$, $\boldsymbol{\Sigma}^{(k)} \in \mathbb{R}^{2\times 2}$, $k = 1, \ldots, N_C$
Generate $\boldsymbol{X}$ s.t. $\boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{\mu}^{(k)}, \boldsymbol{\Sigma}^{(k)})$, $i = 1, \ldots, M$
Include 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.

3 Neural network model

The two data sets are hand-crafted to have the same number of features and overall classes. This choice could be flexible however. The program that implements the simulation is capable to self-adjust the number of neurons according to the input size and the categories number. First table below gives an overview of the network architecture. The second table reports the setting of the optimization algorithm used.

Algorithm 4 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.
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
$\boldsymbol{y}_i = \text{one-hot}(L)$



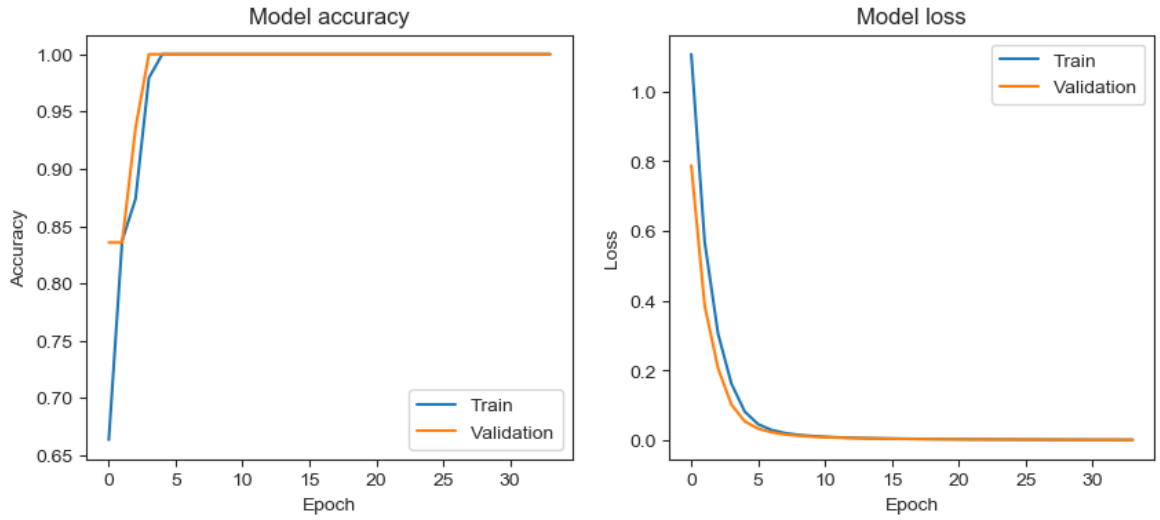
(a) Binary tree data set

(b) Clusters data set

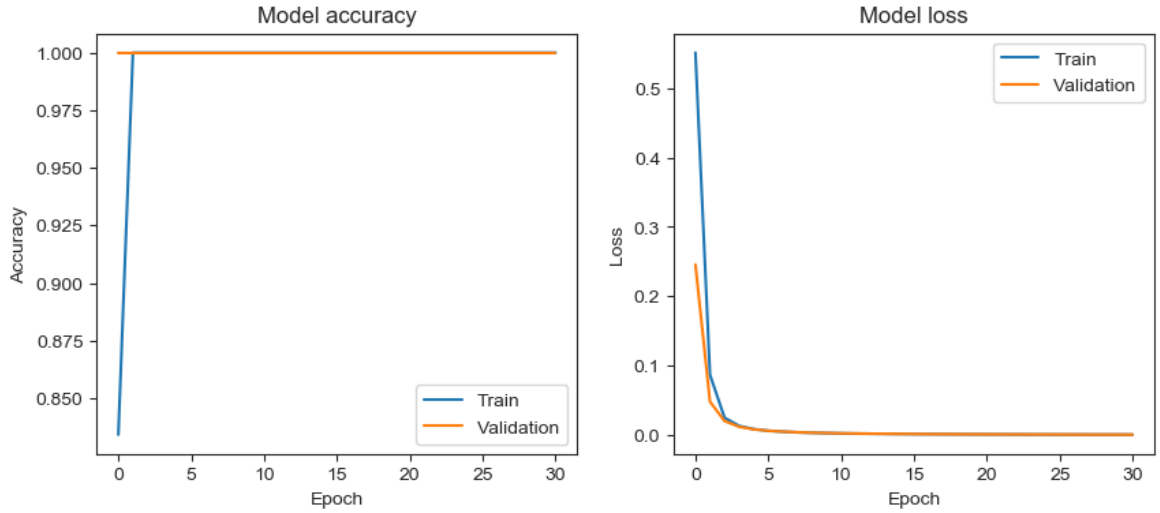
Figure 3: Covariances matrices visualisation of the two different data sets.

Architecture		
Layer	Units	Activation
1 (input)	$N = 31$	-
2 (hidden)	20	ReLU
3 (hidden)	10	ReLU
4* (output)	4	Softmax

NASGD	
Learing rate	0.01
Decay	10^{-6}
Momentum	0.6



(a) First layer



(b) Second layer

Figure 4: Frequency plots of the parameters before and after training. Binary tree data set.

4 Results

It is not striking that the model learns in just few iterations to classify correctly the samples. Data sets are easily tractable, in particular the clusters data sets. Here the systems sees non-−1 values in correspondence of a class, then it is straight forward to learn what is the underlying pattern, regardless the fact that the non-−1 values are real numbers, if one implements an ancestral sampling using normal distributions, or as in the present case, where features are simply the topological order of the vertices.

4.1 Binary tree data set

The simple neural network illustrated is trained with the data set. A 0.7 fraction is used for training, a 0.1 of which is used as validation set, and the 0.3 kept apart before is used as test set. Being the data set linearly separable, in few epochs the accuracy metric attains the top value of 1.0.

Test set accuracy	1.0
Test set loss	0.0015

As final result, weights and biases distributions are also reported for the trained network, alongside with the connections strengths visualisation, Figure 12b.

4.2 Independent clusters data set

Since here the map to be learned from the *states space* (i.e. the domain of the data) to the *target space* is straightforward (it does not take a neural network to understand the game), learning takes two epochs to be over, with a smaller value of the test loss.

Test set accuracy	1.0
Test set loss	0.00063

4.3 Modularly varying goals

It is interesting also to inspect the response of the system once exposed to two different data sets formerly discussed. What is immediately clear is that more motifs are observed, likely due to the fact that the neural network undergoes a longer training stage and then the weights are updated forward the optimum for a prolonged period. The fact that the cost is monotonically decreasing for both the two data sets suggest that even an optimal configuration is found, the cost function hyper-surface resembles a mild hill slope, where the parameters configuration point keeps descending.

An explanation could be that the data sets used are indeed so good that the more the model is trained, the more it is specialized on such tasks. It could be in some sense an occurrence of a subtle over-fitting. Attached to this document the complete log of the training agenda is provided.

4.4 Motifs detection

The `mfinder1.2` executable program (documentation here) is exploited in the next stage, that is inspecting the system in search of significant patterns.

As an introductory note: the weights of the neural network as obtained by the Keras build-in function, are real-valued of course. The cited program only can deal with $+1$ valued edges strengths. Then, as an exploratory experiment, the weights exceeding (both in positive and negative value), a given cutoff threshold, are set to $+1$ and to 0 otherwise. By thus doing the resulting graph item can be fed to the `mfinder1.2` program.

5 Conclusions and remarks

As pointed out in Figure 11, binary tree data set-fed neural network exhibits a greater number of motifs, both in diversity and in quantity of patterns found. In particular, the *diamond* motif (id204) is the mostly occurring one.

Nodes	Motif ID	N_{real}	N_{rand} stats	N_{real} Z-score	N_{real} P-val	Unique Val	C real
4	14	25	17.8 ± 2.8	2.61	0.000	4	15.63
4	76	316	267.4 ± 21.9	2.22	0.010	8	197.62
4	204	12	3.0 ± 1.6	5.66	0.000	4	7.50
4	280	224	183.0 ± 13.9	2.95	0.000	7	140.09
4	392	424	337.3 ± 25.4	3.42	0.000	7	265.17
4	904	38	3.7 ± 2.1	16.14	0.000	4	23.76
4	2184	107	91.2 ± 6.3	2.50	0.000	8	66.92

Table 1: Results of the `mfinder1.2` program for the level 2 data set.

Nodes	Motif ID	N_{real}	N_{rand} stats	N_{real} Z-score	N_{real} P-val	Unique Val	C real
4	280	147	127.8 ± 9.4	2.04	0.010	8	129.29
4	392	373	265.6 ± 23.0	4.66	0.000	8	328.06
4	904	18	2.3 ± 1.5	10.74	0.000	6	15.83

Table 2: Results of the `mfinder1.2` program for the clusters data set.

Nodes	Motif ID	N_{real}	N_{rand} stats	N_{real} Z-score	N_{real} P-val	Unique Val	C real
4	14	56	36.0 ± 4.7	4.27	0.000	6	15.44
4	28	294	206.6 ± 17.8	4.91	0.000	5	81.06
4	204	47	9.1 ± 3.2	11.83	0.000	5	12.96
4	280	534	409.9 ± 23.8	5.22	0.000	9	147.23
4	392	1022	762.3 ± 45.9	5.65	0.000	7	281.78
4	904	95	11.7 ± 3.6	23.25	0.000	5	26.19
4	2184	298	246.0 ± 12.2	4.27	0.000	11	82.16

Table 3: Results of the `mfinder1.2` program for the MVG-undergone training.

To sketch an hypothesis, one could start from Figure 3, in which sharply different structures of the covariance matrices are appreciable. While 3a shows a periodic pattern in the covariances between random variables in the tree structure, thus meaning that there are no such variables which do not share anything with some others, Figure 3b clearly shows that there are some blocks within which correlation may be found while all of the variables inter-blocks are not correlated. In the former case the network ends up by displaying a larger number of motifs. This may in some way suggest that the more tangled topology of the network is due to a more *coherent* internal shape. On the other hand, a more *fragmented* statistical signature may resolve in a more self-organized structure in the network morphology.

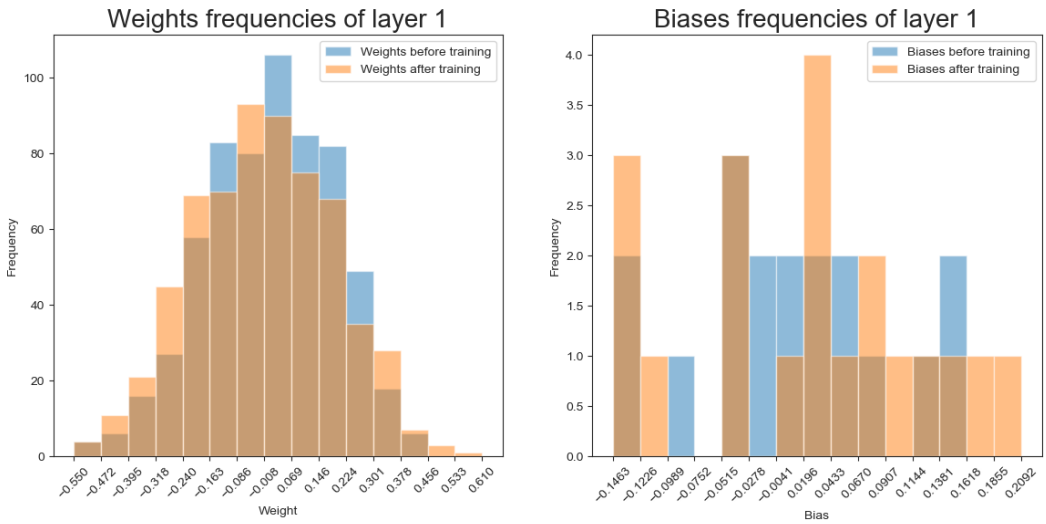
It is also worth a glance to the MVG-undergone system. In this case, a new motif showed up (id28) and again the diamond and the *bifan* (id 904 and 204 respectively) are the motifs that are more frequently observed. Referring to Figure 10c, it could be spotted a greater *diffusion* of the initial weights distribution, which means that the connection of the last layer experience a greater increase of magnitude.

As a final remark, in the MVG case, even if it is observed the mentioned weights increase, there are no edges exhibiting much greater values. This may suggest that

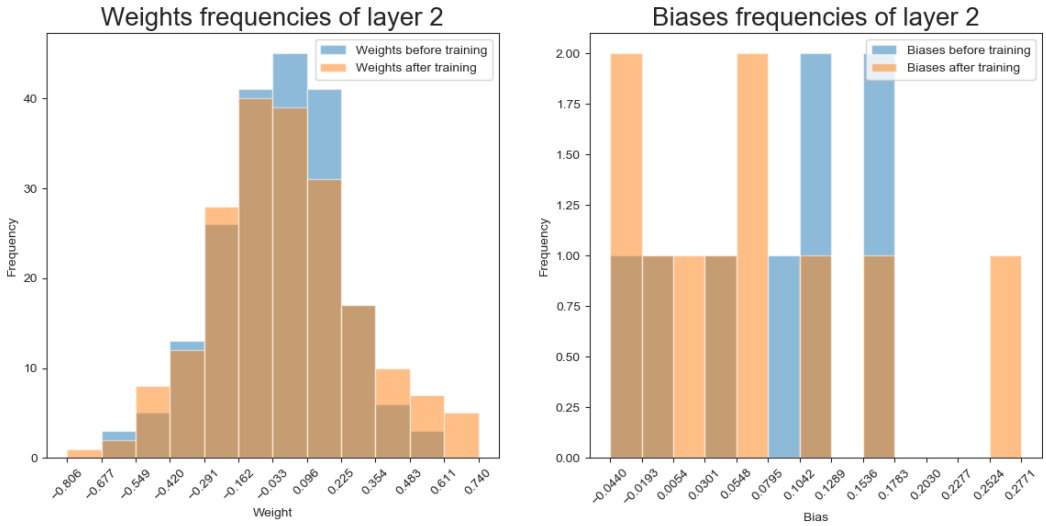
the different statistical signatures of the chosen data sets are different enough not to enforce certain network motifs, indeed related to one particular statistical structure.

5.1 Further improvements

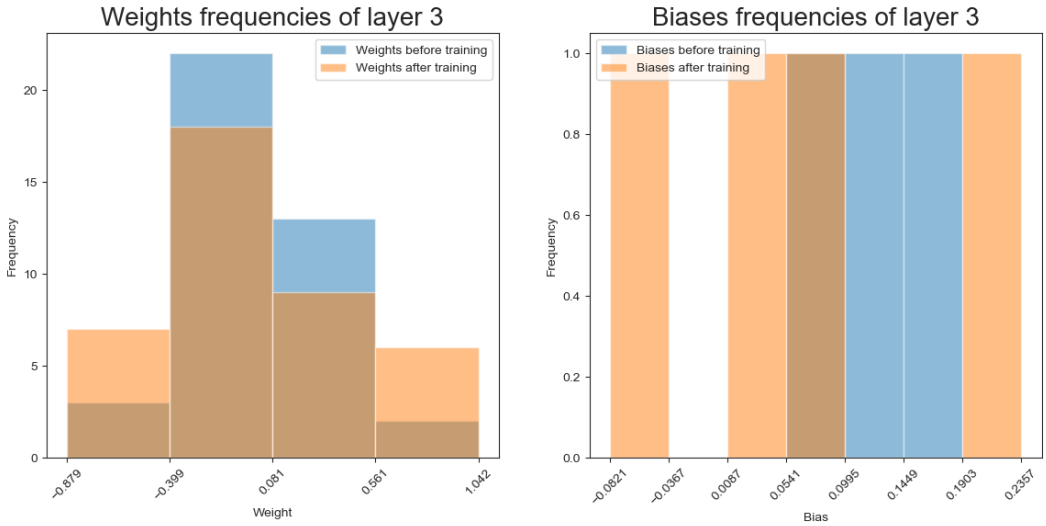
At the present stage the main concern can not be ignored any further: The instrument used to detect network patterns needs to be enhanced in such a way to include the edges strengths information in the motifs inspection stage. The next step in this work then is to create from scratch a procedure to accomplish such a task.



(a) First layer

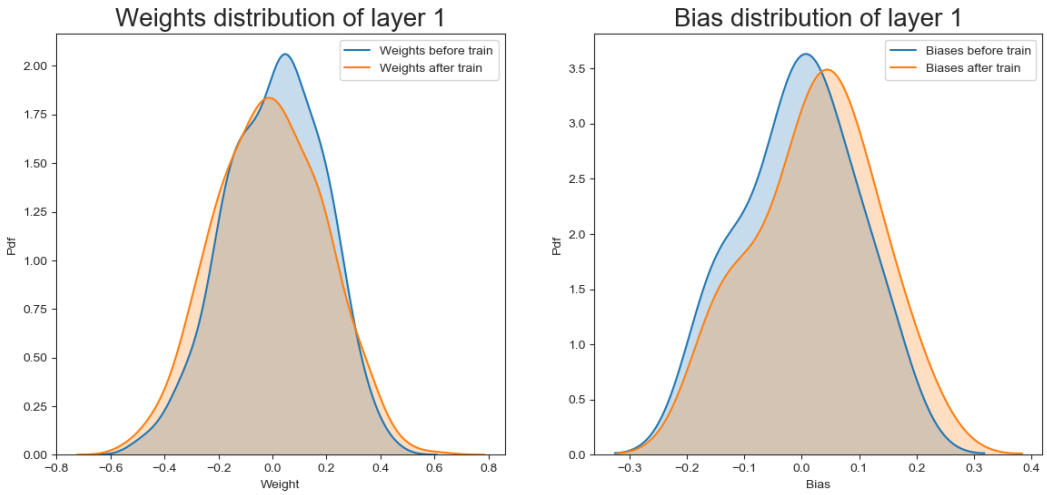


(b) Second layer

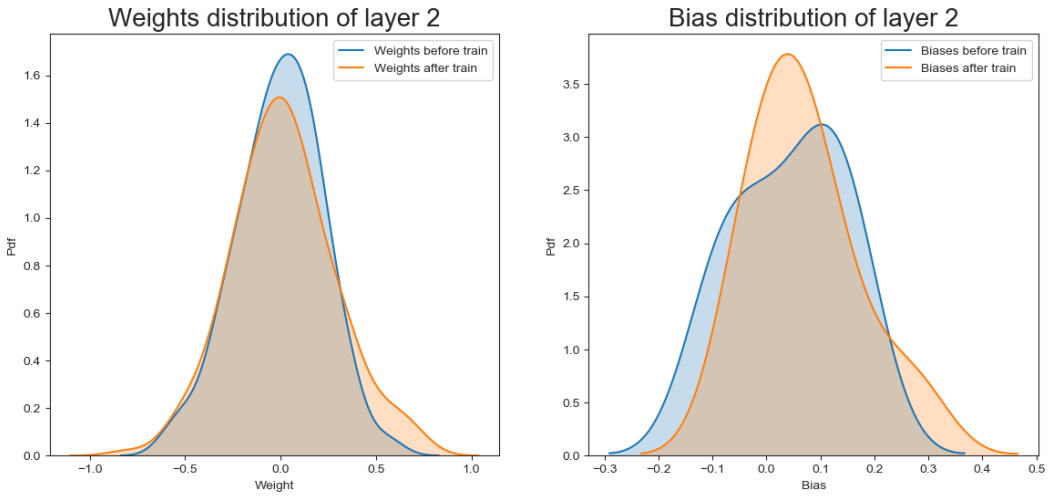


(c) Third layer

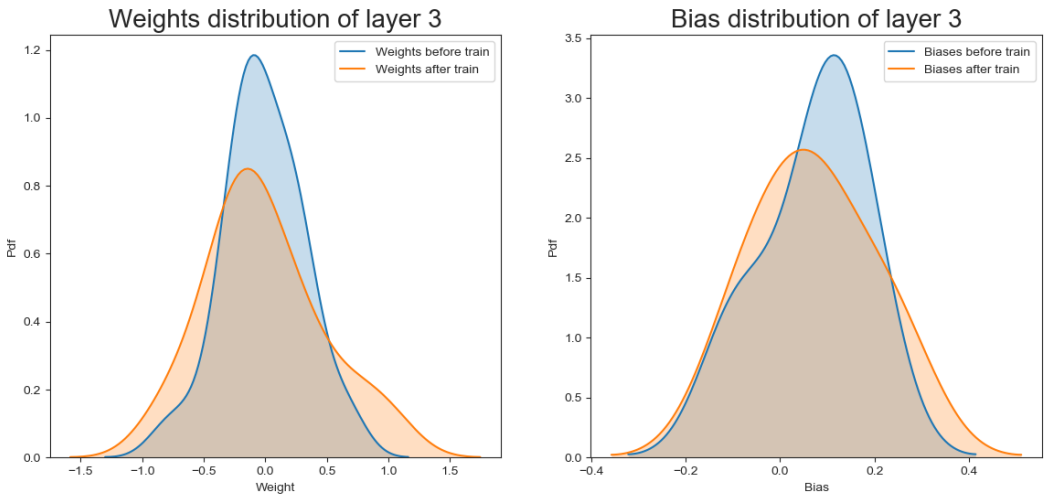
Figure 5: Frequency plots of the parameters before and after training. Binary tree data set.



(a) First layer

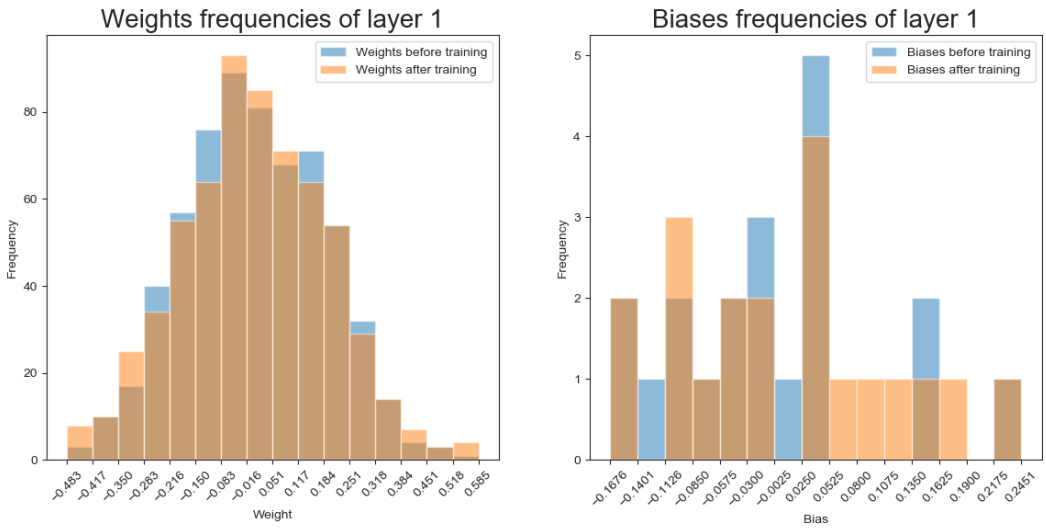


(b) Second layer

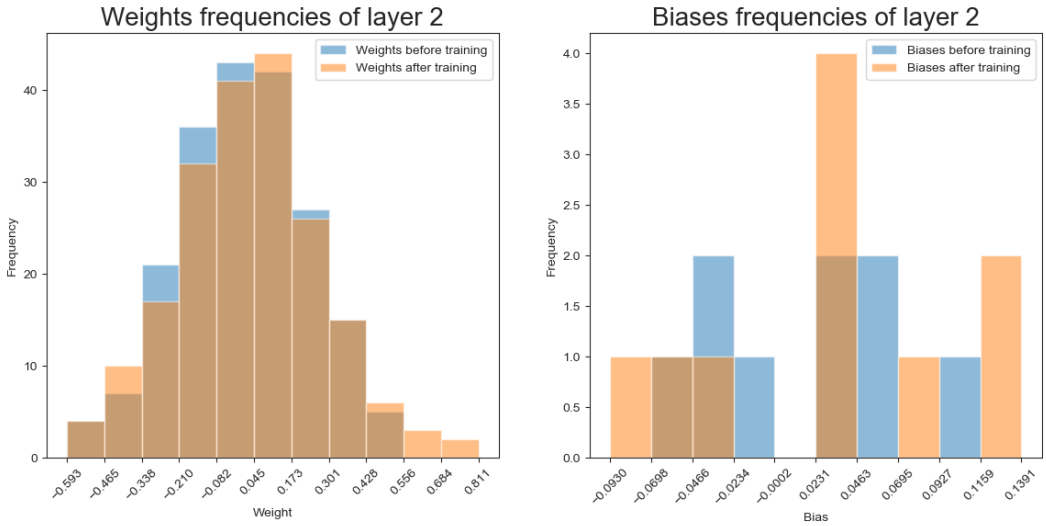


(c) Third layer

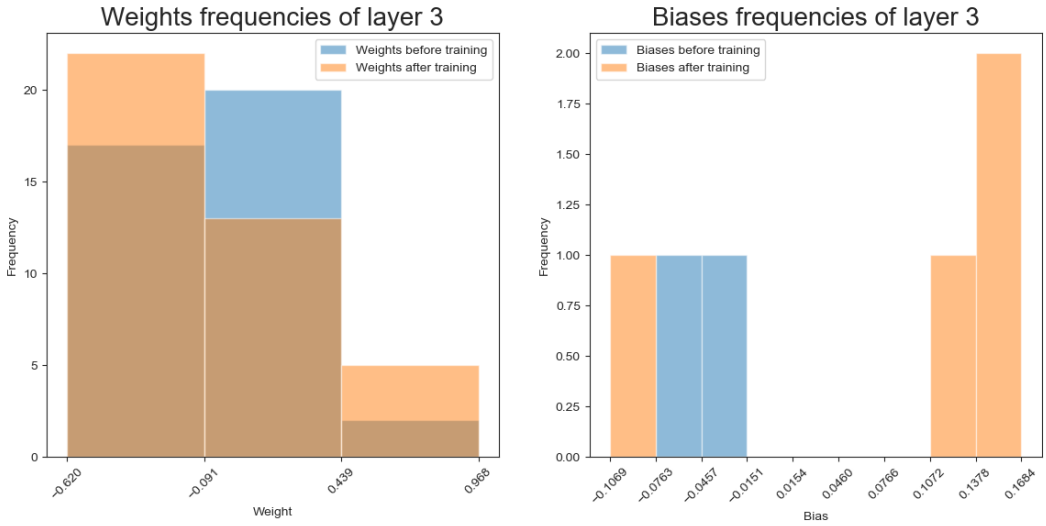
Figure 6: KDE plots of the parameters before and after training. Binary tree data set.



(a) First layer

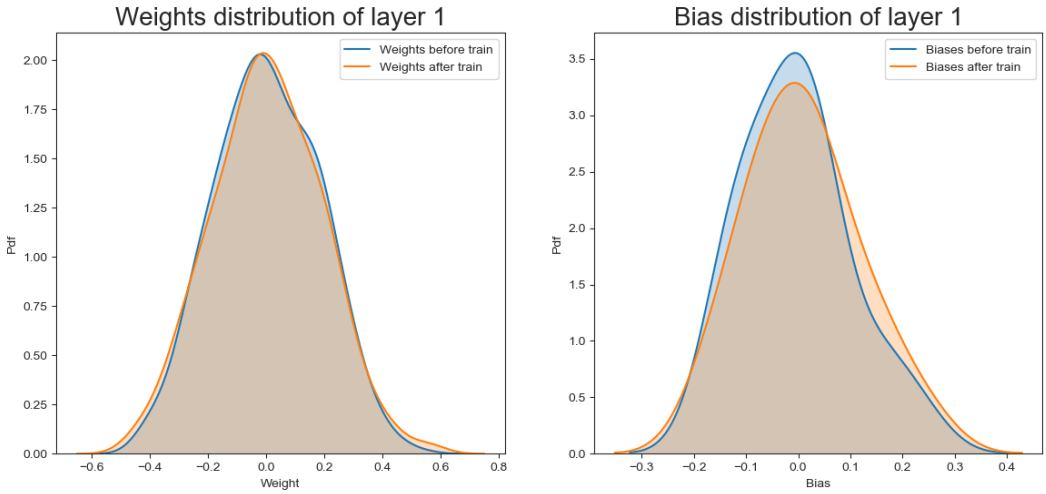


(b) Second layer

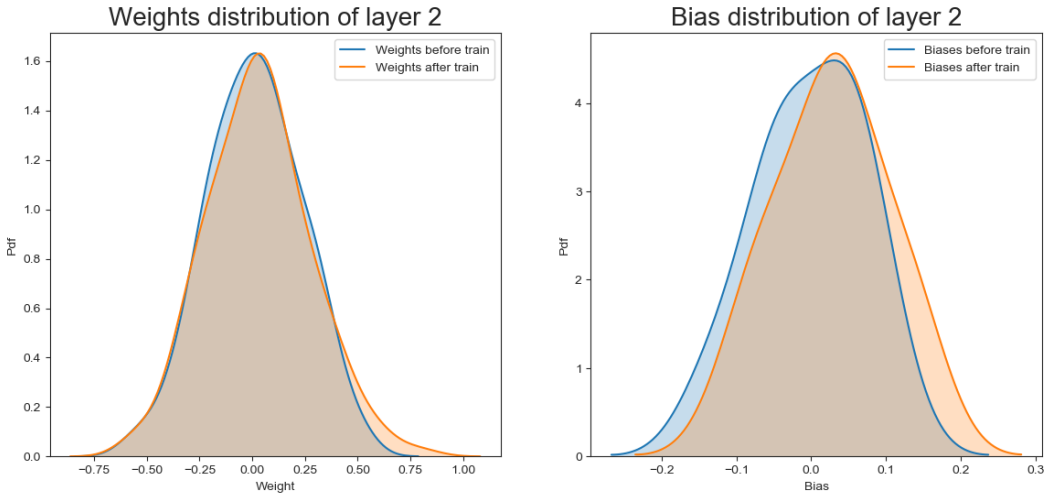


(c) Third layer

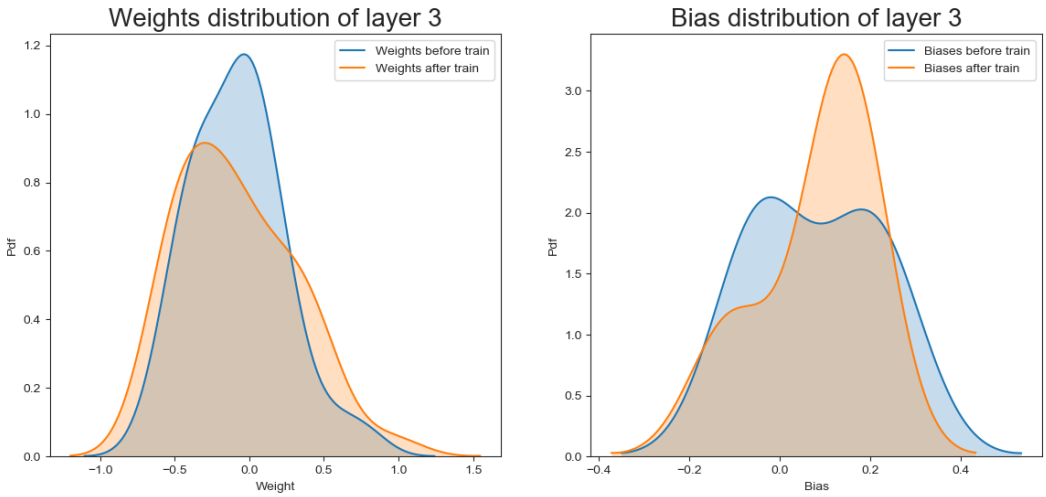
Figure 7: Frequency plots of the parameters before and after training. Clusters data set.



(a) First layer

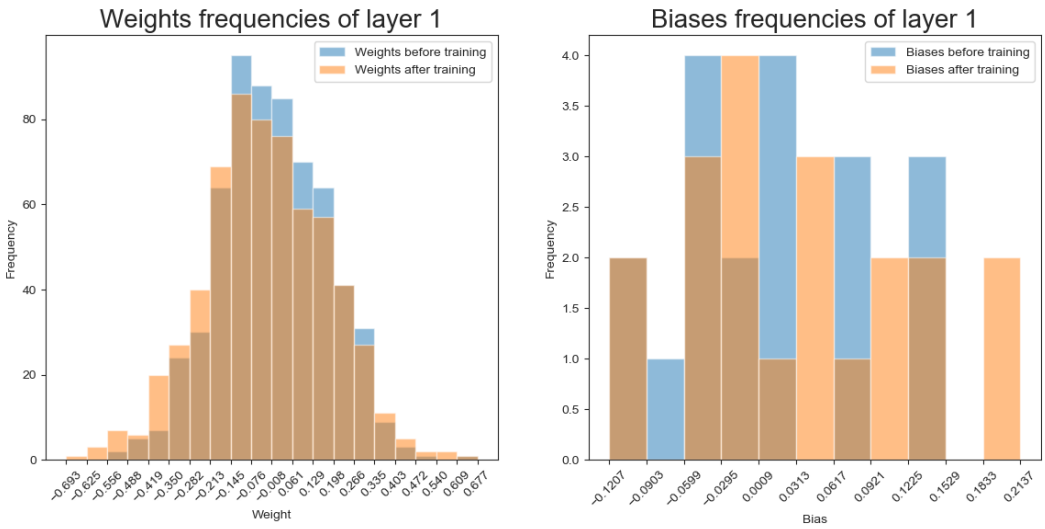


(b) Second layer

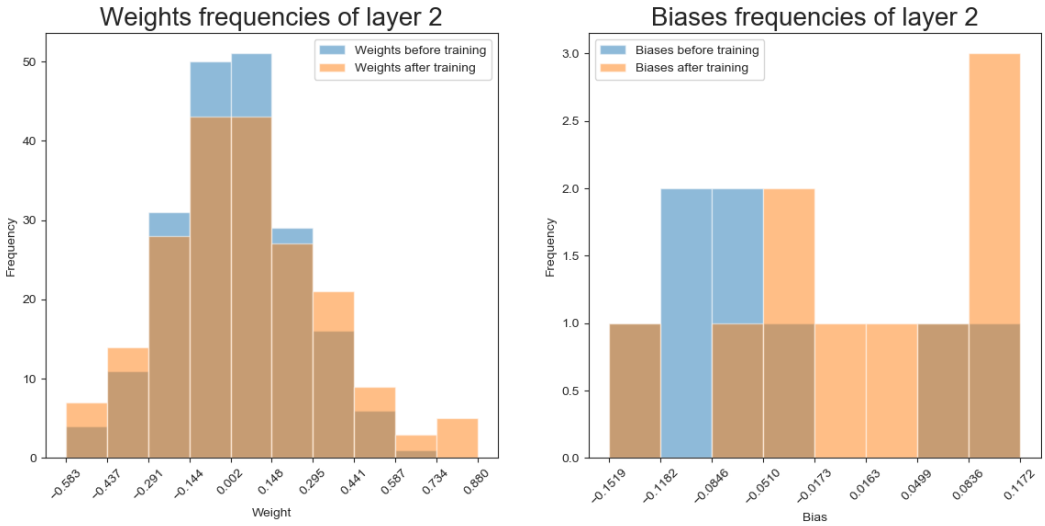


(c) Third layer

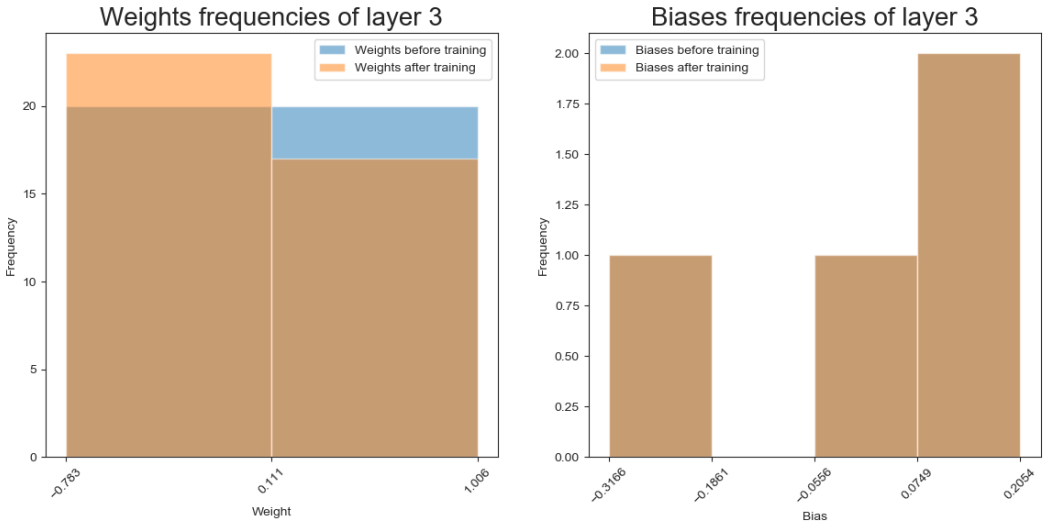
Figure 8: KDE plots of the parameters before and after training. Clusters data set.



(a) First layer

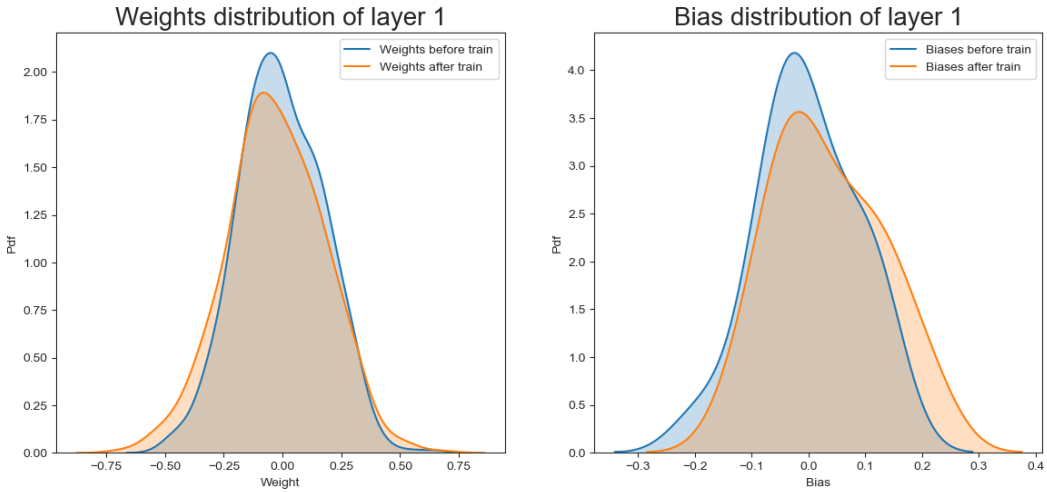


(b) Second layer

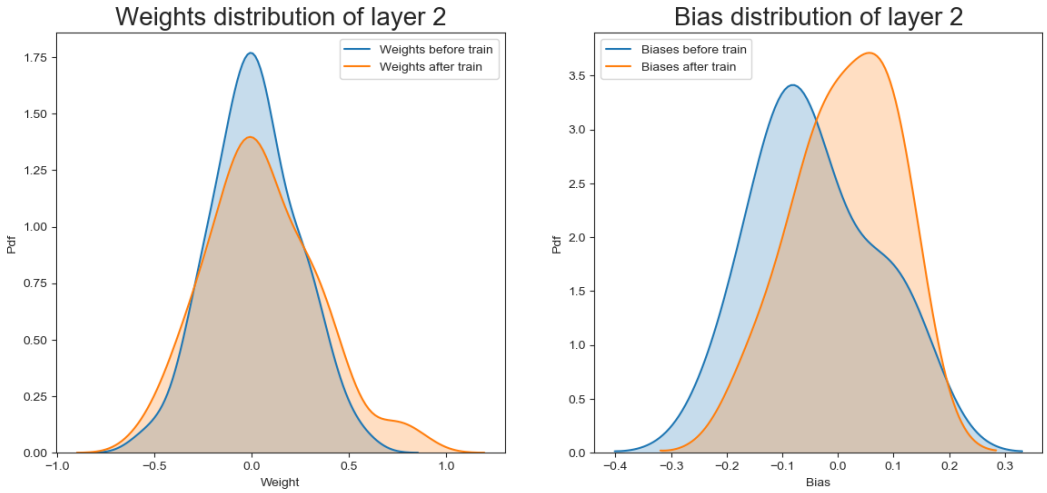


(c) Third layer

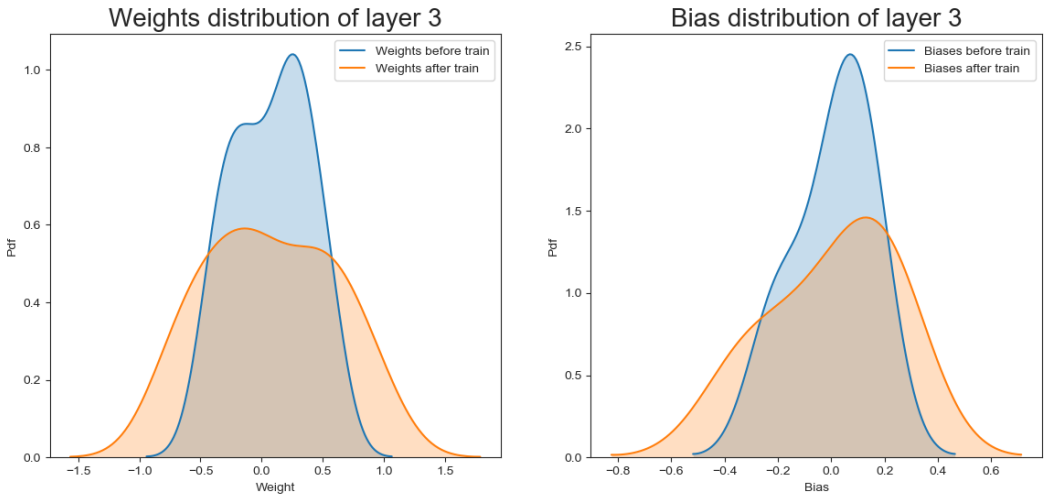
Figure 9: Frequency plots of the parameters before and after training. MVG.



(a) First layer

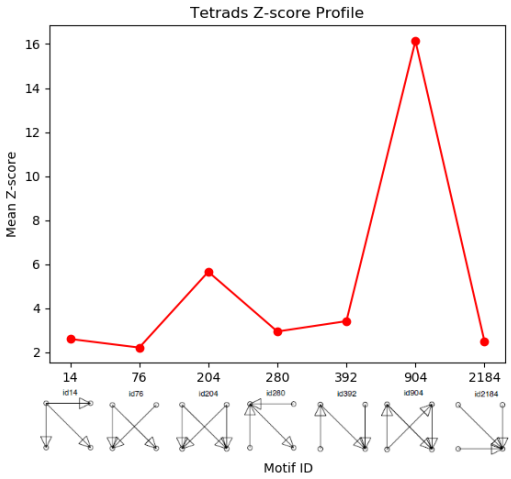


(b) Second layer

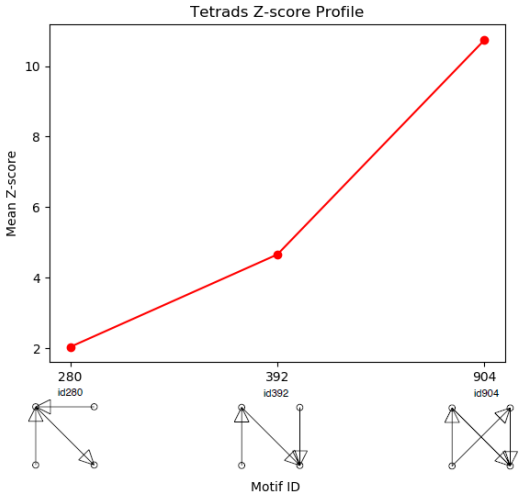


(c) Third layer

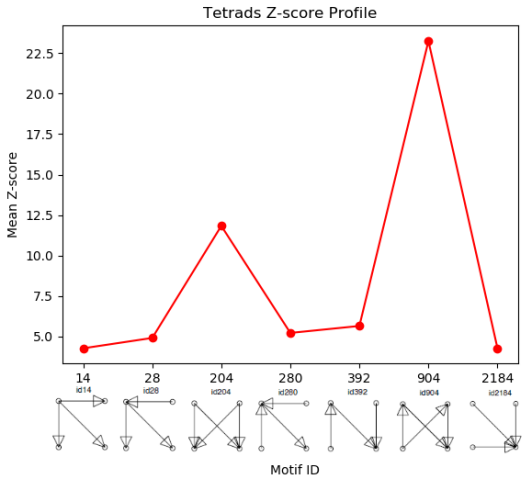
Figure 10: KDE plots of the parameters before and after training. MVG.



(a) Z-scores of the motifs found for the network fed with the tree data set

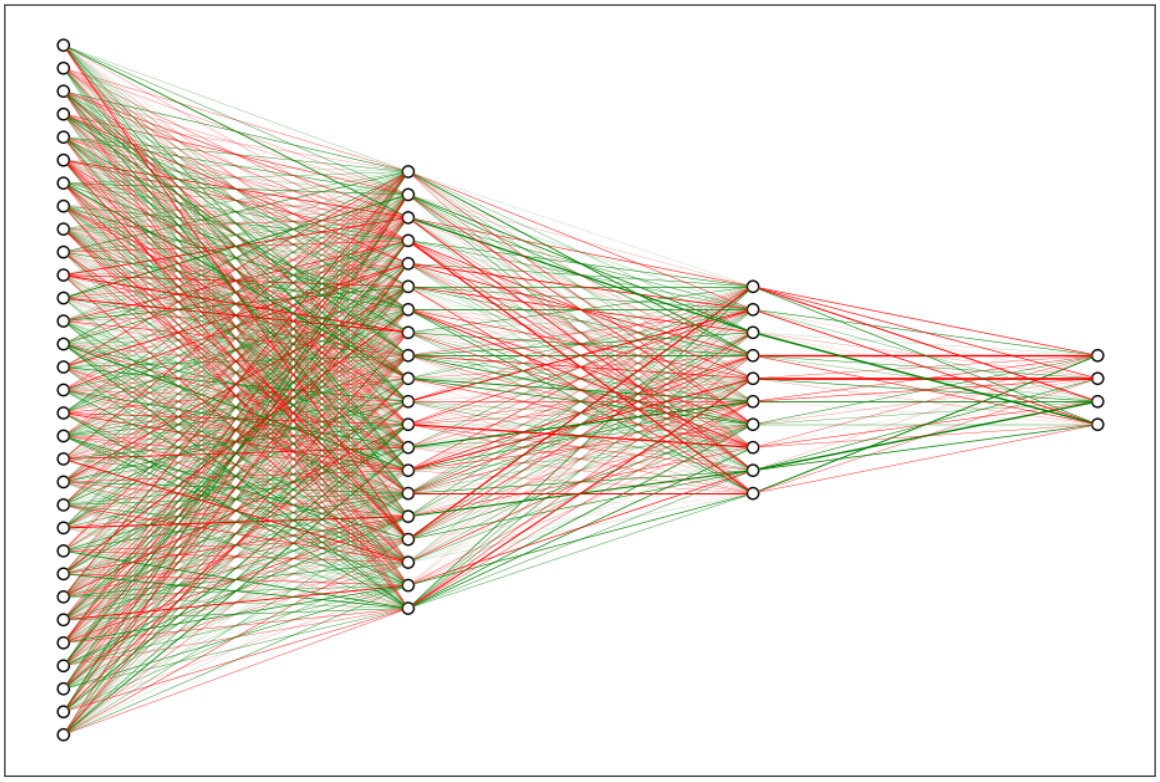


(b) Z-scores of the motifs found for the network fed with the clusters data set

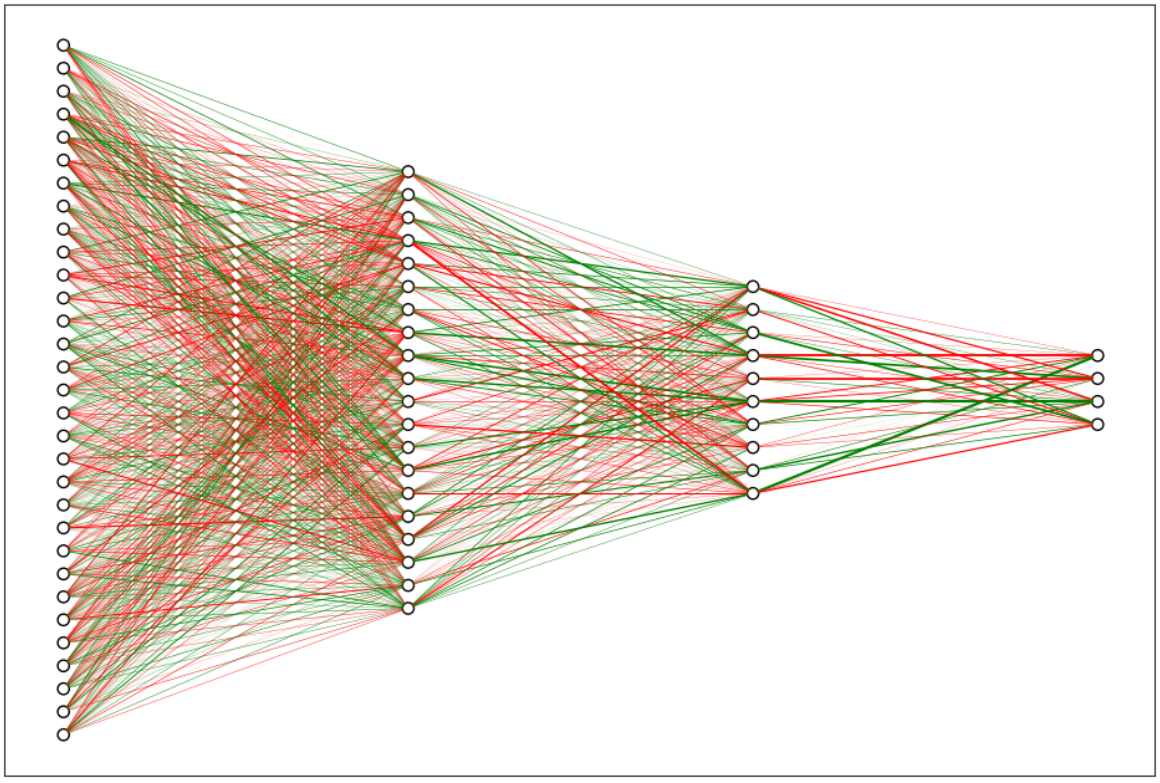


(c) Z-scores of the motifs found for the network undergone the MVG training

Figure 11: Z-scores for the two different data sets and the MVG version of training.

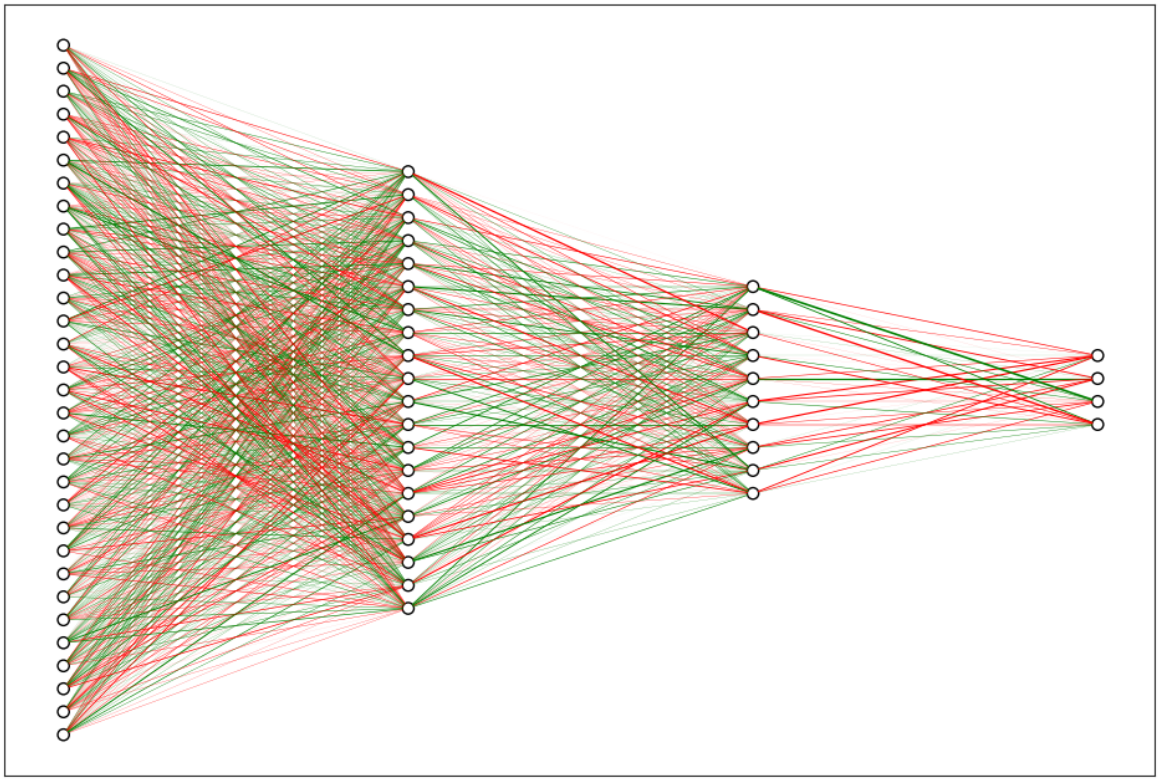


(a) Visualisation of the untrained network topological structure with connection strengths highlighted in red for negative valued weights and green for positive values ones.

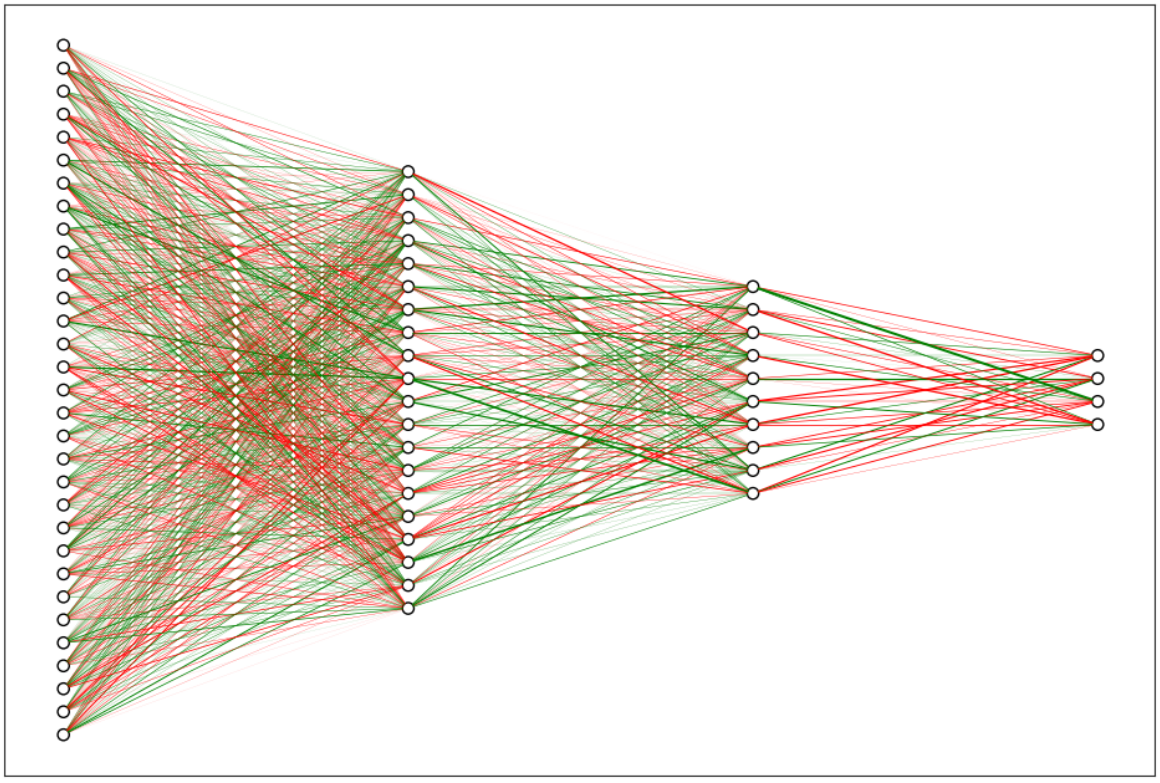


(b) Visualisation of the trained network topological structure with connection strengths highlighted in red for negative valued weights and green for positive values ones.

Figure 12: Visualisation of the network strengths for the binary tree data set.

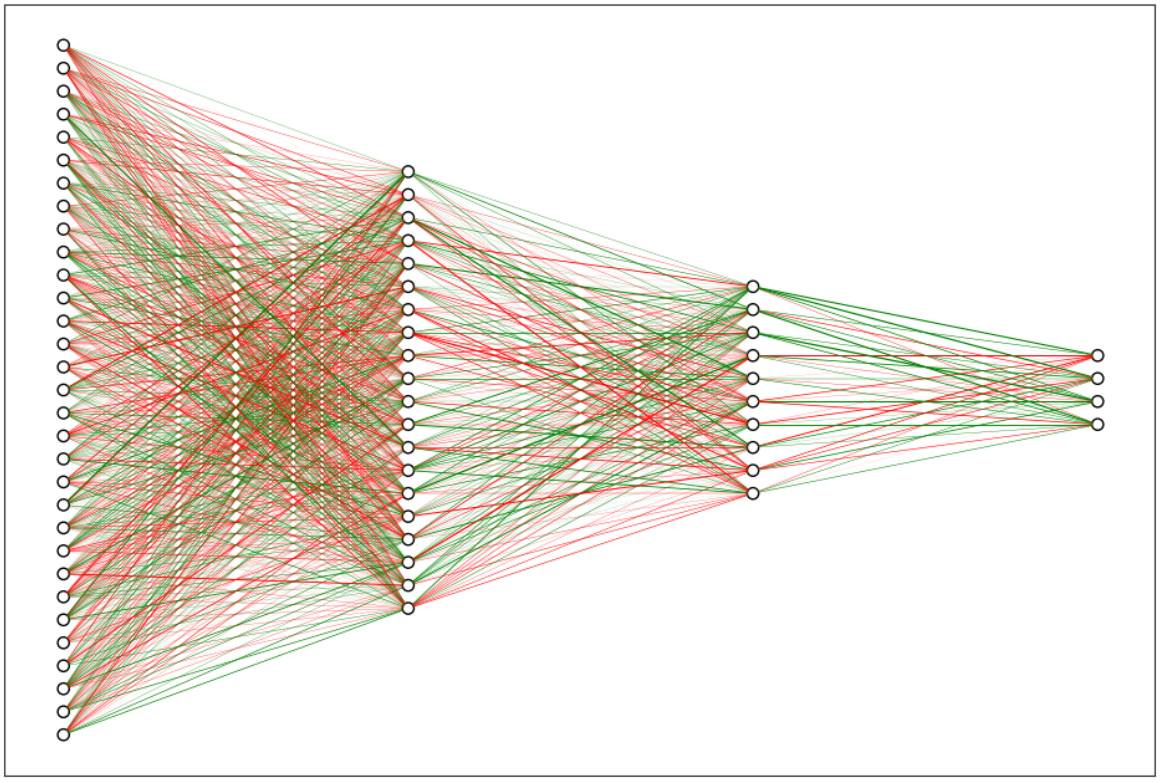


(a) Visualisation of the untrained network topological structure with connection strengths highlighted in red for negative valued weights and green for positive values ones.

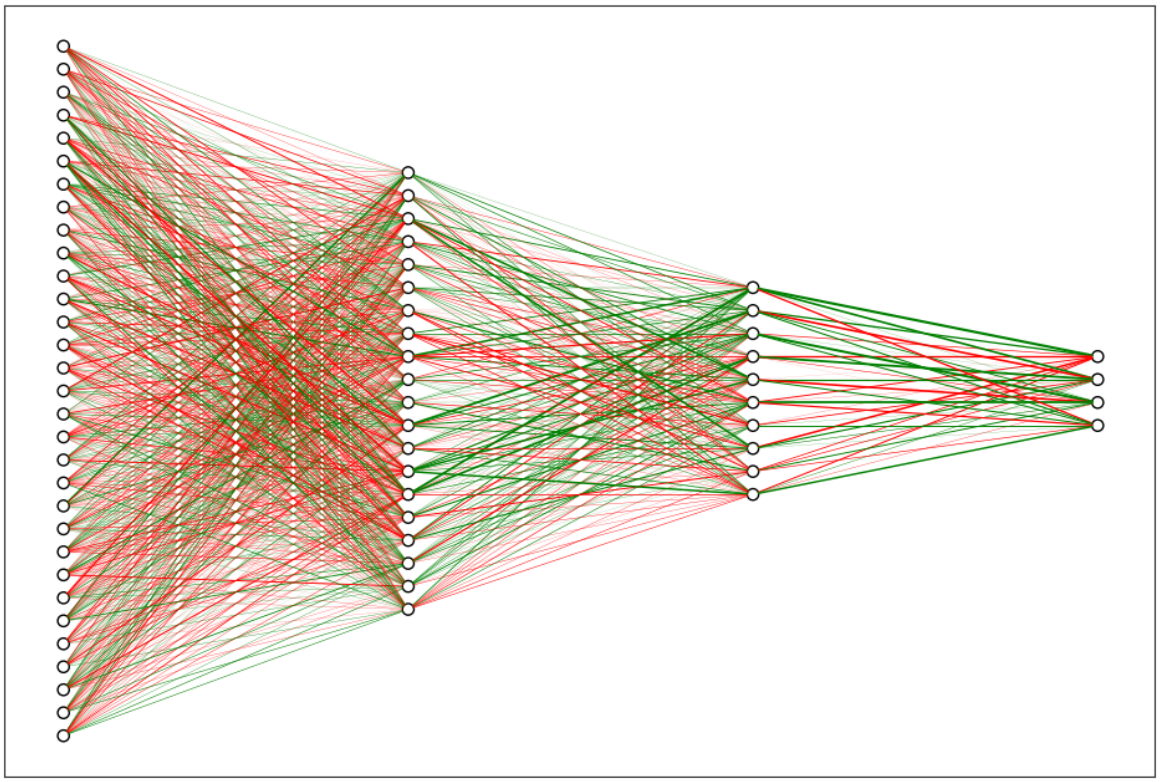


(b) Visualisation of the trained network topological structure with connection strengths highlighted in red for negative valued weights and green for positive values ones.

Figure 13: Visualisation of the network strengths for the clusters data set.



(a) Visualisation of the untrained network topological structure with connection strengths highlighted in red for negative valued weights and green for positive values ones.



(b) Visualisation of the trained network topological structure with connection strengths highlighted in red for negative valued weights and green for positive values ones.

Figure 14: Visualisation of the network strengths for the MVG-training undergone system.