Breaking the Structure of Multilayer Perceptrons with Complex Topologies

Tommaso Boccato [1] Matteo Ferrante [1] Andrea Duggento [1] Nicola Toschi [1 2]

Abstract

Recent advances in neural network (NN) architectures have demonstrated that complex topologies possess the potential to surpass the performance of conventional feedforward networks. Nonetheless, previous studies investigating the relationship between network topology and model performance have yielded inconsistent results, complicating their applicability in contexts beyond those scrutinized. In this study, we examine the utility of directed acyclic graphs (DAGs) for modeling intricate relationships among neurons within NNs. We introduce a novel algorithm for the efficient training of DAG-based networks and assess their performance relative to multilayer perceptrons (MLPs). Through experimentation on synthetic datasets featuring varying levels of difficulty and noise, we observe that complex networks founded on pertinent graphs outperform MLPs in terms of accuracy, particularly within high-difficulty scenarios. Additionally, we explore the theoretical underpinnings of these observations and explore the potential trade-offs associated with employing complex networks. Our research offers valuable insights into the capabilities and constraints of complex NN architectures, thus contributing to the ongoing pursuit of designing more potent and efficient deep learning models.

1. Introduction

Modern neural architectures are widely believed to draw significant design inspiration from biological neuronal networks. The artificial neuron, the fundamental functional unit of neural networks (NNs), is based on the McCulloch-Pitts unit (Fitch, 1944), sharing conceptual similarities with its biological counterpart. Additionally, state-of-the-art convolutional NNs incorporate several operations directly inspired by the mammalian primary visual cortex, such as nonlinear transduction, divisive normalization, and maximum-based pooling of inputs. However, these architectures may be among the few examples where the evolutionary structural and functional properties of neuronal systems have been genuinely relevant for NN design. Indeed, the topology of biological connectomes has not yet been translated into deep learning model engineering.

Due to the ease of implementation and deployment, widely-used neural architectures predominantly feature a regular structure resembling a sequence of functional blocks (e.g., neuronal layers). The underlying multipartite graph of a multilayer perceptron (MLP) is typically controlled by a few hyperparameters that define its basic topological properties: depth, width, and layer sizes. Only recently have computer vision engineers transitioned from chain-like structures (Simonyan & Zisserman, 2014) to more elaborate connectivity patterns (He et al., 2016; Huang et al., 2017; Xie et al., 2019a) (e.g., skip connections, complete graphs). Nevertheless, biological neuronal networks display much richer and less templated wirings at both the micro- and macro-scale (Fornito et al., 2013). Considering synaptic connections between individual neurons, the *C. elegans* nematode features a hierarchical modular (Bassett et al., 2010) connectome, wherein hubs with high betweenness centrality are efficiently interconnected (Barthelemy, 2004; Towlson et al., 2013). Moreover, the strength distribution of the adult Drosophila central brain closely follows a power law with an exponential cutoff (Scheffer et al., 2020).

As a result, the relationship between the graph structure of a NN and its predictive abilities remains unclear. In the literature, there is evidence that complex networks can be advantageous in terms of predictive accuracy and parameter efficiency (Kaviani & Sohn, 2021). However, past attempts to investigate this connection have yielded conflicting results that are difficult to generalize outside the investigated context. The first experiment on complex NNs was performed in 2005 by Simard et al., who trained a randomly rewired MLP on random binary patterns (Simard et al., 2005). Nearly a decade later, Erkaymaz and his collaborators employed the same experimental setup on various

[1]Department of Biomedicine and Prevention, University of Rome Tor Vergata, Rome, Italy [2]A.A. Martinos Center for Biomedical Imaging and Harvard Medical School, Boston, USA. Correspondence to: Tommaso Boccato <tommaso.boccato@uniroma2.it>.

Proceedings of the 2^{nd} Annual Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML) at the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. 2023. Copyright 2023 by the author(s).

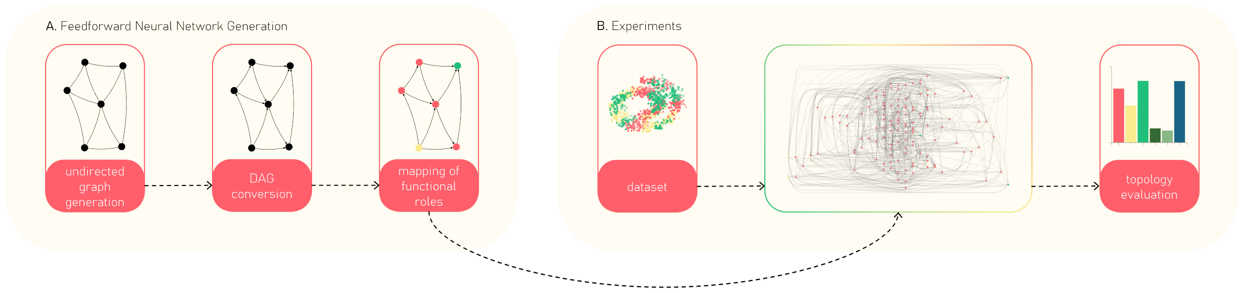
A. Feedforward Neural Network Generation
undirected graph generation
DAG conversion
mapping of functional roles
B. Experiments
dataset
topology evaluation

Figure 1. Overview of the topology exploration process. **Left**: Feedforward neural network (NN) generation. All studied models are constructed using the same three-step procedure. First, we generate an undirected graph with a predetermined degree distribution. Then, we set edge directions and map computational operations to the network nodes. **Right**: Experiments. For each investigated topology, we sample multiple graphs from the same degree distribution. The corresponding NNs are trained on one of the benchmark datasets. The resulting test accuracies are collected and stored for subsequent analyses.

real-life problems (Erkaymaz et al., 2012; 2014; Erkaymaz & Ozer, 2016; Erkaymaz et al., 2017) (e.g., diabetes diagnosis, performance prediction of solar air collectors). The best-performing models featured a number of rewirings consistent with the small-world regime. However, all assessed topologies were constrained by MLP-random interpolation. In (Annunziato et al., 2007), an MLP and a NN generated following the Barabási-Albert (BA) procedure were compared on a chemical process modeling problem. Both models were trained with an evolutionary algorithm, but the MLP achieved a lower RMSE. The *learning matrix* (Monteiro et al., 2016), a sequential algorithm for the forward/backward pass of arbitrary directed acyclic graphs (DAGs), enabled the evaluation of several well-known complex networks on classification (Monteiro et al., 2016) and regression (Platt et al., 2019) tasks. The experiments included random and small-world networks, two topologies based on “preferential attachment”, a complete graph, and a *C. elegans* subnetwork (Dunn et al., 2004). Nevertheless, the learning matrix’s time complexity limited the network sizes (i.e., 26 nodes), and for each task, a different winning topology emerged, including the MLP. Also Stier et al. successfully trained BA- and WS-based (Watts-Strogatz) NNs with backpropagation (Stier & Granitzer, 2019) on the MNIST classification task (Lecun et al., 1998) by placing the generated networks between two fully-connected layers. While this design choice was made in order to adapt the architecture to the dimensionalities of the input/output, it may represent a confound when disentangling the contributions of the different network modules to the overall classification performance. Some recent works have instead focused on multipartite sparse graphs (Mocanu et al., 2018; You et al., 2020). While these architectures outperformed the complete baselines, their topological complexity was entirely encoded within the connections between adjacent layers.

Another body of work is based on computational graphs defined on a macroscopic scale (Xie et al., 2019b; Wortsman et al., 2019; Roberts et al., 2019). In these NNs, nodes represent compositions of operators while links represent “flows” of tensors or channels. Generally, Neural Architecture Search (NAS), which aims to automate the design of neural architectures, also falls within this category. The discipline, born in its modern variant with (Zoph & Le, 2017), which uses a controller based on reinforcement learning to generate NN architectural hyperparameters, has recently shifted towards techniques capable of formulating topological search in a differentiable manner (Liu et al., 2019; Gu et al., 2021). However, these techniques are applied to cells (small subgraphs with fewer than 10 nodes) that are repeated within the overall architecture. Although NAS has achieved remarkable results in vision and NLP, the constraints imposed on the microscopic topology of the generated networks make these models unsuitable for the tasks for which the networks mentioned in the previous paragraph are employed.

We propose the hypothesis that, given the same number of nodes (i.e., neurons) and edges (i.e., parameters), a complex NN might exhibit superior predictive abilities compared to classical, more regularly structured MLPs. Unlike previous studies, we conduct a systematic exploration (of which we have reported an overview in Figure 1) of random, scale-free, and small-world graphs (Figure 2) on synthetic classification tasks, with particular emphasis on the following:

- **Network size.** The defining properties of a complex topology often emerge in large-scale networks. For example, the second moment of a power-law degree distribution diverges only in the $N \to \infty$ limit (Barabási, 2016), where N is the network size[1]. The networks in (Monteiro et al., 2016; Platt et al., 2019) have 15 and

[1]The proposition holds when the degree exponent is smaller than 3.

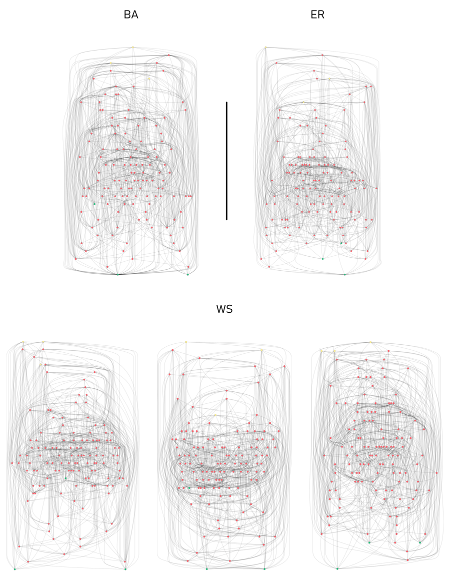

Figure 2. Example feedforward NNs (128 neurons, 732 synaptic connections) based on complex topologies: scale-free (BA), random (ER), and small-world (WS). All graphs are directed and acyclic. Information flows from top to bottom. Input, hidden, and output units are denoted in green, red, and yellow, respectively. Since the networks are defined at the micro-scale, hidden and output nodes implement weighted sums over the incoming edges. In the hidden units, the computational operation is followed by an activation function. The activations of nodes located on the same horizontal layer can be computed in parallel.

26 nodes, respectively. We trained models with 128 neurons.

- **Dataset size.** The *estimation error* achieved by a predictor depends on the training set size: the greater the number of samples, the lower the error (Shalev-Shwartz & Ben-David, 2014). Except for studies based on multipartite graphs, all previous research works in a small-data regime. Our datasets are three times larger than those used before.
- **Hyperparameter optimization.** Learning rate and batch size are crucial in minimizing the loss function. Ref. (Monteiro et al., 2016) is the only one that considers finding the optimal learning rate. The role of batch size has never been investigated. Each DAG, however, could be characterized by its optimal combination of hyperparameters. Hence, we optimized the learning rate and batch size for each topology.

2. Theory

In this section, we briefly report on the theory behind some graph generators from network science. These graph models are involved in generating the NNs employed in our investigation, as discussed in Section 3.

Erdős-Rényi (ER). An ER graph (Erdős et al., 1960), or *random network*, is uniformly sampled from the set of all graphs with N nodes and L edges. For $N \gg \langle k \rangle$, the degree distribution of a random graph is well approximated by a Poisson distribution: $p_k = e^{-\langle k \rangle} \frac{\langle k \rangle^k}{k!}$; k and $\langle k \rangle$ represent node degree and average degree, respectively.

Watts-Strogatz (WS). The WS generator (Watts & Strogatz, 1998) aims to create graphs that exhibit both high clustering and the *small-world* property; this is achieved by interpolating *lattices* with random networks. The generation starts from a ring in which nodes are connected to their immediate neighbors. The links are then randomly rewired with probability p.

Barabási-Albert (BA). The well-known BA model (Albert & Barabási, 2002) can be used to generate networks characterized by the $p_k \propto k^{-3}$ *scale-free* degree distribution. Being the model inspired by the growth of real networks, the generative procedure iteratively attaches nodes with m stubs to a graph that evolves from an initial star of $m+1$ nodes. Node additions respond to the preferential attachment mechanism: the probability that a stub reaches a node is proportional to the degree of the latter.

Multilayer Perceptron (MLP). The underlying networks of MLPs are called multipartite graphs. In a multipartite graph (i.e., a sequence of bipartite graphs) nodes are partitioned into layers, and each layer can only be connected with the adjacent ones; no intra-layer link is allowed. Additionally, inter-layer connections have to form *bicliques* (i.e., fully-connected bipartite graphs).

3. Methods

The following sections present our methodology. Section 3.1 describes how our benchmark datasets are constructed. In Section 3.2, we provide details on the proposed NN generation pipeline. Finally, Section 3.3 details out the experimental protocol.

3.1. Datasets

The foundation of the datasets developed, as displayed in Figure 3, is established by manifold learning generators[2] provided by the `scikit-learn` machine learning library (Pedregosa et al., 2011). To modify the generators for classification purposes, 3D points sampled from one of the available curves (*s curve* and *swiss roll*) are segmented into $\texttt{n\_classes} \times \texttt{n\_reps}$ portions based on their univariate position relative to the primary dimension of the manifold

[2] `https://scikit-learn.org/stable/datasets/sample_generators.html`

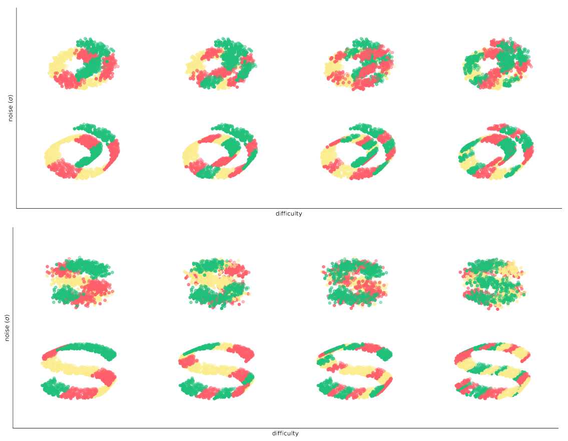

Figure 3. Benchmark classification datasets. **Top**: the *swiss roll*. **Bottom**: the *s curve*. Each dataset is composed of 3D points divided into multiple segments. Classes are color-coded. Datasets differ in terms of difficulty (x axis) and noise (y axis).

samples. As the term implies, `n_classes` refers to the number of classes involved in the considered classification. Each segment is then arbitrarily allocated to a class, maintaining task balance (i.e., precisely `n_reps` segments have the same label). We define `n_reps` as the task *difficulty*. An additional aspect of our datasets is the standard deviation σ of the Gaussian noise that can be added to the points. The generation procedure is finalized with a min-max normalization.

3.2. Feedforward Neural Networks

All trainable models are produced following the same 3-step procedure and share N and L. Consequently, NNs exhibit identical density and parameter counts.

Undirected Graph Generation. The initial step in creating a NN involves sampling an undirected graph using the generators detailed in Section 2. Once N and L are established, all models exhibit a single parameter configuration compatible with the required density[3]. The WS generator is the sole exception: the probability p is allowed to vary between 0 and 1. If the generator is limited to sample networks with a number of links from a finite set (e.g., $L = m + (N - m - 1)m$ according to the BA model), we first generate a graph with slightly higher density than the target before randomly eliminating excess edges. After obtaining the graph, we confirm the existence of a single connected component.

Directed Acyclic Graph (DAG) Conversion. Before performing any calculations, the direction for information propagation through the network links must be determined; this is accomplished by randomly assigning, without replacement, an integer index from $\{1, \dots, N\}$ to the network nodes. It can be shown that the directed graph obtained by setting the direction of each edge from the node with a lower index to the node with a higher index is free of cycles. However, this conversion results in an unpredictable number of sources and sinks. Since classification tasks typically involve a pre-defined number of input features and output classes, it is necessary to resolve such network-task discrepancies. To address this issue, we developed a straightforward heuristic (Appendix A) capable of adjusting DAGs without altering the underlying undirected graphs.

Mapping of Functional Roles. The last step of the presented procedure consists in mapping computational operations to the DAG nodes. Working at the micro-scale (i.e., connections between single neurons), the operations allowed are two. Source nodes implement constant functions; their role, indeed, is to feed the network with the initial conditions for computations. Hidden and sink nodes, instead, perform a weighted sum over the incoming edges, followed by an activation function:

$$a_v = \sigma\left(\sum_u w_{uv} a_u + b\right) \tag{1}$$

where a_v is the activation of node v, σ denotes the activation function[4] (SELU (Klambauer et al., 2017) for hidden nodes and the identity function for sinks), u represents the generic predecessor of v, w_{uv} is the weight associated with edge (u, v) and b the bias.

Algorithm 1 NN Initializtion
```
$\mathcal{L} \leftarrow longest\_path(\mathcal{G})$
for $l \leftarrow 1$ to $|\mathcal{L}| - 1$ do
  $P^l \leftarrow \emptyset$
  for $v \in L_l$ do
    $P^l \cup \{u : (u, v) \in \mathcal{E}\}$
  end for
  $M^l \leftarrow \mathbf{0}$
  $W^l \leftarrow \mathbf{0}$
  for $v \in L_l$ do
    for $u \in \{u : (u, v) \in \mathcal{E}\}$ do
      $m_{uv} \leftarrow 1$
    end for
  end for
  $W^l \leftarrow init(W^l, M^l)$
end for
```

[3]This statement is accurate if the number of MLP layers is predetermined.

[4]Depending on the context, we use the same σ notation for both the standard deviation of the dataset noise and the activation function.

Algorithm 2 NN Forward Pass

```
for $l \leftarrow 1$ to $|\mathcal{L}| - 1$ do
    $\boldsymbol{a}^l \leftarrow \sigma((M^l \odot W^l)\boldsymbol{x}^l)$
end for
```

We implemented the mapping of functional roles in Python[5], based on the pseudocode of Algorithm 1. Starting from a DAG (i.e., $\mathcal{G} = (\mathcal{N}, \mathcal{E})$), our code returns an NN that can be deployed as a PyTorch `Module`. Its forward pass follows Algorithm 2. The procedure was designed to facilitate systematic experiments with NNs composed of hundreds of neurons. According to our preliminary findings, the forward pass execution time depends heavily on the number of matrix multiplications performed. Previous methods for functional role mapping yield models that require, in the worst case, a number of operations either linear, for sparse DAGs (Monteiro et al., 2016), or quadratic, for dense DAGs (Stier & Granitzer, 2019), in the number of nodes. By computing activations in parallel, our algorithm performs a number of matrix multiplications that is, at most, linear but typically sublinear in the number of nodes, depending on the given graph topology. In Algorithm 1, the input DAG is partitioned into a minimum height layering $\mathcal{L} = \{L_0, \ldots, L_{|\mathcal{L}|-1}\}$ (i.e., the smallest ordered set of layers in which nodes in a layer only receive connections from nodes in previous layers) using the $longest\_path()$ algorithm (Tamassia, 2016; Healy & Nikolov, 2002). For each layer, the algorithm tracks node predecessors, P^l, and utilizes this information to initialize a learnable weight matrix, $W^l \in \mathbb{R}^{|L_l| \times |P^l|}$. Specifically, we allocate memory for all predecessor-target connections (i.e., $(u, v) : u \in P^l,\ v \in L_l$). Some edges, however, may not exist in the original graph; thus, we employ a binary mask $M^l \in \{0, 1\}^{|L_l| \times |P^l|}$ to preserve the input topology and prevent the gradient from updating non-existent connections. Finally, the weight matrix is initialized through $init()$, considering the actual node degrees. Algorithm 2 delineates how the generated NN performs the forward pass. For each layer, the activations of node predecessors, $\boldsymbol{x}^l$, are collected and multiplied by the masked weight matrix (i.e., $M^l \odot W^l$), which is obtained through a point-wise multiplication between the mask and weight matrices. The resulting vector, $\boldsymbol{a}^l$, corresponds to the current layer activations. An example forward pass is depicted in Figure 4.

3.3. Experiments

In this section, we outline the experimental protocol devised to compare the performance of various graph topologies investigated.

[5]The code can be accessed at https://github.com/BoCtrl-C/forward.

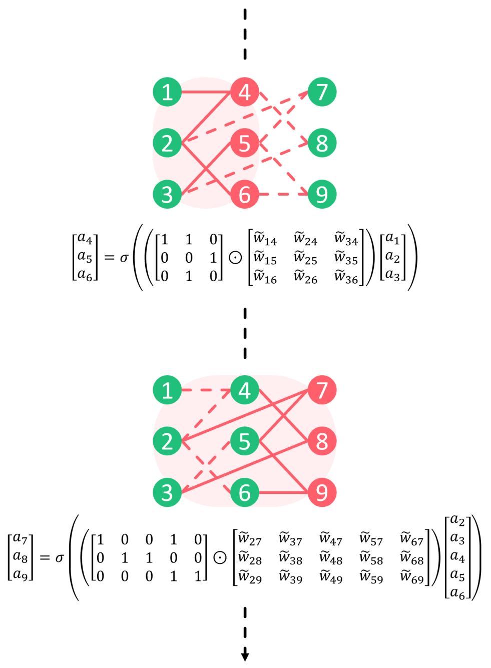
1
2
3
4
5
6
7
8
9
$\begin{bmatrix}a_4\\a_5\\a_6\end{bmatrix} = \sigma\left(\left(\begin{bmatrix}1&1&0\\0&0&1\\0&1&0\end{bmatrix} \odot \begin{bmatrix}\widetilde{w}_{14}&\widetilde{w}_{24}&\widetilde{w}_{34}\\\widetilde{w}_{15}&\widetilde{w}_{25}&\widetilde{w}_{35}\\\widetilde{w}_{16}&\widetilde{w}_{26}&\widetilde{w}_{36}\end{bmatrix}\right)\begin{bmatrix}a_1\\a_2\\a_3\end{bmatrix}\right)$
$\begin{bmatrix}a_7\\a_8\\a_9\end{bmatrix} = \sigma\left(\left(\begin{bmatrix}1&0&0&1&0\\0&1&1&0&0\\0&0&0&1&1\end{bmatrix} \odot \begin{bmatrix}\widetilde{w}_{27}&\widetilde{w}_{37}&\widetilde{w}_{47}&\widetilde{w}_{57}&\widetilde{w}_{67}\\\widetilde{w}_{28}&\widetilde{w}_{38}&\widetilde{w}_{48}&\widetilde{w}_{58}&\widetilde{w}_{68}\\\widetilde{w}_{29}&\widetilde{w}_{39}&\widetilde{w}_{49}&\widetilde{w}_{59}&\widetilde{w}_{69}\end{bmatrix}\right)\begin{bmatrix}a_2\\a_3\\a_4\\a_5\\a_6\end{bmatrix}\right)$

Figure 4. Forward pass computation for an example NN built through Algorithm 1 (from top to bottom). **Top**: computation of the activations of layer 1. **Bottom**: computation of the activations of layer 2. The subnetworks involved are highlighted through the red overlay. Computations follow $\boldsymbol{a}^l = \sigma((M^l \odot W^l)\boldsymbol{x}^l)$, in accordance with Algorithm 2.

Dataset Partitioning. Each generated dataset is randomly divided into three non-overlapping subsets: the train, validation, and test splits. All model trainings are conducted on the train split, while the validation split is used in validation epochs and hyperparameter optimization. Test samples, on the other hand, are accessed only during the evaluation of the final models.

Model Training. Models are trained by minimizing cross entropy with the Adam (Kingma & Ba, 2015) optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$). A scheduler reduces the learning rate by a factor of 0.5 if no improvement is seen on the validation loss for 10 epochs. The training procedure ends when learning stagnates (w.r.t. the validation loss) for 15 epochs, and the model weights corresponding to the epoch in which the minimum validation loss has been achieved are saved.

Hyparameter Optimization. Hyperparameters are optimized through a grid search over a predefined 2D space (i.e., learning rate/batch size). We generate networks of the same topological family starting from 5 different random seeds. In the MLP case, models differ only in the weight initialization. For each parameter pair, the 5 models are trained accordingly, and the resulting best validation losses are collected. Then, the learning rate and batch size that minimize the median validation loss computed across the generation seeds are selected as the optimal hyperparameters of the considered graph family.

Topology Evaluation. Once the optimal learning rate and batch size are found, we train 15 new models characterized by the considered topology and compute mean classification accuracy and standard deviation on the dataset test split. The procedure is repeated for each investigated graph family and a Kruskal-Wallis (H-test) (Kruskal & Wallis, 1952) is performed in order to test the null hypothesis that the medians of all accuracy populations are equal. If the null hypothesis is rejected, a Mann-Whitney (U-test) (Mann & Whitney, 1947) post hoc analysis follows.

4. Results

We obtained the presented results by following the experimental protocol outlined in Section 3 using the specified topologies (i.e., BA, ER, MLP, and WS) and datasets. We set `n_classes` $= 3$ and `n_reps` $\in 3, 6, 9, 12$; for the *swiss roll* dataset, $\sigma \in 0.0, 1.0$, while for the *s curve*, $\sigma \in 0.0, 0.3$. The train, validation, and test split sizes were 1350, 675, and 675, respectively. Given that in a 1-hidden layer MLP (`h1` notation) the number of synaptic connections depends solely on N (i.e., $L = 3 \times H + H \times 3$, with $H = N - 3 - 3$), we chose an MLP with 128 neurons as a reference model and calculated the hyperparameters for the complex networks to achieve graphs with $L = 732$ edges. The additional degree of freedom in the WS generator enabled us to separate the small-world topology into three distinct graph families: `p.5` ($p = 0.5$), `p.7` ($p = 0.7$), and `p.9` ($p = 0.9$). The hyperparameter optimization searched for learning rates in {0.03, 0.01, 0.003, 0.001} and batch sizes in {32, 64}.

Figure 5 displays the mean test accuracy achieved by each group of models as a function of task difficulty. All manifolds, noise levels, and difficulties are represented. Excluding difficulty level 9 in the *swiss roll* dataset, the accuracy curves exhibit a clear decreasing trend. Specifically, as the difficulty increases, the performance of the MLPs degrades more rapidly than that of complex networks. Confidence intervals, on the other hand, are wider in the high-difficulty plot regions. As expected, noisy tasks were more challenging to learn.

In Figure 6, the results obtained by the models for the two highest levels of task difficulty are shown in detail. The H-test null hypothesis is rejected for all experiments, and the U-test statistical annotations are displayed. Regardless of the scenario considered, a complex topology consistently holds the top spot in the mean accuracy ranking. MLPs, in contrast, are always the worst-performing models. Moreover, the MLP performance differs significantly from that of the complex networks, in a statistical sense.

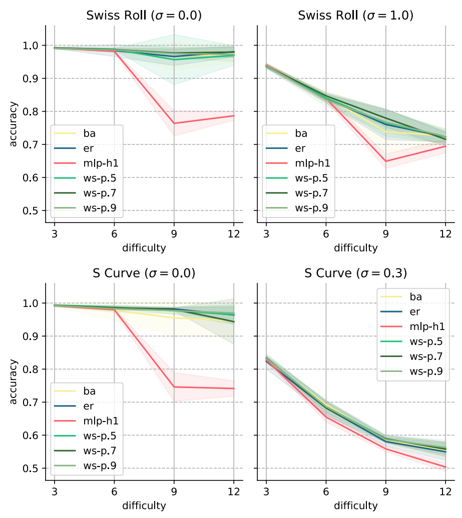

Figure 5. Mean test accuracy as a function of the task difficulty. Confidence intervals ($\pm$ standard deviation) are reported as well. Different subplots correspond to different datasets. Each curve denotes the trend of a specific network topology.

5. Discussion

The most significant finding from the experiments performed is the performance in terms of accuracy attained by the architectures built on complex topologies in the high-difficulty regime. In this context, and in light of the statistical tests carried out, the complex models prove to be a solid alternative to MLPs.

Formally justifying the observed phenomenon is challenging. Fortunately, in 2017, Poggio et al. discussed two theorems (Poggio et al., 2017) that guided our explanation. According to the first theorem[6], a shallow network (e.g., an MLP `h1`) equipped with infinitely differentiable activation functions requires $N = \mathcal{O}(\epsilon^{-n})$ units to approximate a continuous function f of n variables with an approximation error of at most $\epsilon > 0$. This exponential dependency is technically called the *curse of dimensionality*. On the other hand, the second theorem states that if f is compositional and the network presents its same architecture, we can escape the "curse". It is important to remember that a compositional function is defined as a composition of "local" constituent functions, $h \in \mathcal{H}$ (e.g., $f(x_1, x_2, x_3) = h_2(h_1(x_1, x_2), x_3)$, where $x_1,\ x_2,\ x_3$ are the input variables and $h_1,\ h_2$ the constituent functions). In other words, the structure of a compositional function can be represented by a DAG. In this approximation scenario, the required number of units depends on $N = \mathcal{O}(\sum_h \epsilon^{-n_h})$, where n_h is the input dimensionality of function h. If $\max_h n_h = d$, then

[6]We invite the reader to consult ref. (Poggio et al., 2017) for a complete formulation of the theorems.

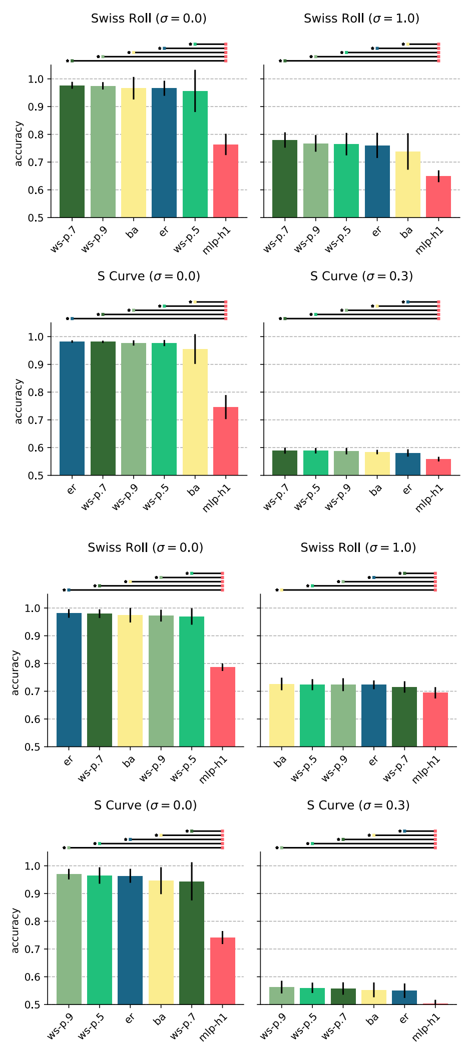

Figure 6. Mean test accuracy at the highest difficulty levels. **Top**: difficulty $= 9$. **Bottom**: difficulty $= 12$. The bars display both means and standard deviations. Each bar corresponds to a specific network topology and is represented by a consistent color across all histograms (following the color scheme from Figure 5). Statistical annotations appear above the histograms, with each segment indicating a significant difference between two accuracy distributions.

$\sum_h \epsilon^{-n_h} \leq \sum_h \epsilon^{-d} = |\mathcal{H}|\epsilon^{-d}$.

The primary advantage of complex networks is their potential to avoid the curse of dimensionality when relevant graphs for the function to be learned are present. Under the assumption that the function linking the *swiss roll* and *s curve* points to the ground truth labels is compositional (intuitively, in non-noisy datasets, each class is a union of various segments), we conjecture that our complex NNs can exploit this compositionality. In the high-difficulty regime, the necessary network size for MLP `h1` to achieve the same accuracy as complex models likely exceeds the size set for experiments. While one could argue that the datasets employed were compositionally sparse by chance, according to (Poggio, 2022), all *efficiently computable functions* must be *compositionally sparse* (i.e., their constituent functions have "small" d). Performance differences on noisy datasets are less noticeable, possibly due to the minimal overlap between the functions to be approximated and the studied topologies. Notably, our setup does not precisely match the theorem formulations in (Poggio et al., 2017) (e.g., SELUs are not infinitely differentiable), but Poggio et al. argue that the hypotheses can likely be relaxed. No statistically significant differences emerged between the complex graph families from the results of Section 4. Various explanations exist for this outcome: all tested topologies could be complex enough to include relevant subgraphs of the target f functions; the random DAG conversion heuristic might have perturbed hidden topological properties of the original undirected networks; or the degree distribution of a network may not be the most relevant topological feature in a model's approximation capabilities.

However, the higher accuracy in complex networks comes with trade-offs. Although the methodology in Section 3.2 improves the scalability of complex NNs and enables experimentation with arbitrary DAGs, it is important to note that 1-hidden layer MLPs typically have faster forward pass computation. In these models, the forward pass requires only two matrix multiplications, whereas, in NNs built using Algorithm 1, the number of operations depends on the DAG *height*.

6. Conclusion

In this study, we investigated the application of complex NN architectures for classification tasks. Our experiments demonstrated that complex topologies can outperform traditional MLPs, particularly in high-difficulty scenarios. This observation is corroborated by the theoretical framework presented in (Poggio et al., 2017), which indicates that complex networks can circumvent the curse of dimensionality by exploiting the compositionality of the function being approximated.

Despite the considerable performance improvements offered by complex networks, there are trade-offs to consider. Specifically, the scalability of these models and the computational cost of the forward pass pose potential challenges. However, our methodological advancements facilitate experimentation with arbitrary DAGs, helping to mitigate some of the scalability concerns.

Finally, it is worth noting that, although the used models exhibit a topology closer to that of real neuronal networks, there are significant structural differences, including the absence of cycles within computational graphs. Transitioning from feedforward complex networks to recurrent networks necessitates the utilization of *backpropagation through time* and novel notions of layering and forward pass. These modifications would introduce design and implementation challenges, solutions to which lie beyond the scope of this work.

In summary, our research suggests that complex NN architectures hold promise within the field of machine learning, particularly for tackling challenging classification tasks. We anticipate that our findings will inspire further investigation into complex topologies and their potential applications across various deep learning domains.

Acknowledgements

This work is supported and funded by: #NEXTGENERATIONEU (NGEU); the Ministry of University and Research (MUR); the National Recovery and Resilience Plan (NRRP); project MNESYS (PE0000006, to NT) - *A Multiscale integrated approach to the study of the nervous system in health and disease* (DN. 1553 11.10.2022); the MUR-PNRR M4C2I1.3 PE6 project PE00000019 Heal Italia (to NT); the NATIONAL CENTRE FOR HPC, BIG DATA AND QUANTUM COMPUTING, within the spoke *"Multiscale Modeling and Engineering Applications"* (to NT); the European Innovation Council (Project CROSSBRAIN - Grant Agreement 101070908, Project BRAINSTORM - Grant Agreement 101099355); the Horizon 2020 research and innovation Programme (Project EXPERIENCE - Grant Agreement 101017727). Tommaso Boccato is a PhD student enrolled in the National PhD in Artificial Intelligence, XXXVII cycle, course on Health and Life Sciences, organized by Università Campus Bio-Medico di Roma.

References

Albert, R. and Barabási, A.-L. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

Annunziato, M., Bertini, I., De Felice, M., and Pizzuti, S. Evolving Complex Neural Networks. In Basili, R. and Pazienza, M. T. (eds.), *AI*IA 2007: Artificial Intelligence and Human-Oriented Computing*, volume 4733, pp. 194–205. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. ISBN 978-3-540-74781-9 978-3-540-74782-6. doi: 10.1007/978-3-540-74782-6_18. URL http://link.springer.com/10.1007/978-3-540-74782-6_18. Series Title: Lecture Notes in Computer Science.

Barabási, A. *Network Science*. Cambridge University Press, 2016. ISBN 9781107076266. URL https://networksciencebook.com/.

Barthelemy, M. Betweenness centrality in large complex networks. *The European physical journal B*, 38(2):163–168, 2004.

Bassett, D. S., Greenfield, D. L., Meyer-Lindenberg, A., Weinberger, D. R., Moore, S. W., and Bullmore, E. T. Efficient physical embedding of topologically complex information processing networks in brains and computer circuits. *PLoS computational biology*, 6(4):e1000748, 2010.

Dunn, N. A., Lockery, S. R., Pierce-Shimomura, J. T., and Conery, J. S. A neural network model of chemotaxis predicts functions of synaptic connections in the nematode *Caenorhabditis elegans*. *J. Comput. Neurosci.*, 17 (2):137–147, 2004. doi: 10.1023/B:JCNS.0000037679.42570.d5. URL https://doi.org/10.1023/B:JCNS.0000037679.42570.d5.

Erdős, P., Rényi, A., et al. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

Erkaymaz, O. and Ozer, M. Impact of small-world network topology on the conventional artificial neural network for the diagnosis of diabetes. *Chaos, Solitons & Fractals*, 83:178–185, February 2016. ISSN 09600779. doi: 10.1016/j.chaos.2015.11.029. URL https://linkinghub.elsevier.com/retrieve/pii/S096007791500394X.

Erkaymaz, O., Özer, M., and Yumuşak, N. Performance Analysis of A Feed-Forward Artifical Neural Network With Small-World Topology. *Procedia Technology*, 1:291–296, 2012. ISSN 22120173. doi: 10.1016/j.protcy.2012.02.062. URL https://linkinghub.elsevier.com/retrieve/pii/S2212017312000631.

Erkaymaz, O., Özer, M., and Yumuşak, N. Impact of small-world topology on the performance of a feed-forward artificial neural network based on 2 different real-life problems. *Turkish Journal of Electrical Engineering and Computer Sciences*, 22, 2014.

Erkaymaz, O., Ozer, M., and Perc, M. Performance of small-world feedforward neural networks for the diagnosis of diabetes. *Applied Mathematics and Computation*, 311:22–28, October 2017. ISSN 00963003. doi: 10.1016/j.amc.2017.05.010. URL https://linkinghub.elsevier.com/retrieve/pii/S0096300317302989.

Fitch, F. B. Warren s. mcculloch and walter pitts. a logical calculus of the ideas immanent in nervous activity. bulletin of mathematical biophysics, vol. 5 (1943), pp. 115–133. *Journal of Symbolic Logic*, 9(2):49–50, 1944. doi: 10.2307/2268029.

Fornito, A., Zalesky, A., and Breakspear, M. Graph analysis of the human connectome: Promise, progress, and pitfalls. *NeuroImage*, 80:426–444, 2013. ISSN 1053-8119. doi: https://doi.org/10.1016/j.neuroimage.2013.04.087. URL https://www.sciencedirect.com/science/article/pii/S1053811913004345. Mapping the Connectome.

Gu, Y.-C., Wang, L.-J., Liu, Y., Yang, Y., Wu, Y.-H., Lu, S.-P., and Cheng, M.-M. DOTS: Decoupling Operation and Topology in Differentiable Architecture Search. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12306–12315, Nashville, TN, USA, June 2021. IEEE. ISBN 978-1-66544-509-2. doi: 10.1109/CVPR46437.2021.01213. URL https://ieeexplore.ieee.org/document/9578254/.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Healy, P. and Nikolov, N. S. How to layer a directed acyclic graph. In Mutzel, P., Jünger, M., and Leipert, S. (eds.), *Graph Drawing*, pp. 16–30, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2017. doi: 10.1109/CVPR.2017.243.

Kaviani, S. and Sohn, I. Application of complex systems topologies in artificial neural networks optimization: An overview. *Expert Systems with Applications*, 180:115073, 2021. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2021.115073. URL https://www.sciencedirect.com/science/article/pii/S0957417421005145.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. Self-normalizing neural networks. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/5d44ee6f2c3f71b73125876103c8f6c4-Paper.pdf.

Kruskal, W. H. and Wallis, W. A. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952. ISSN 01621459. URL http://www.jstor.org/stable/2280779.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Liu, H., Simonyan, K., and Yang, Y. DARTS: differentiable architecture search. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=S1eYHoC5FX.

Mann, H. B. and Whitney, D. R. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1): 50–60, 1947. ISSN 00034851. URL http://www.jstor.org/stable/2236101.

Mocanu, D. C., Mocanu, E., Stone, P., Nguyen, P. H., Gibescu, M., and Liotta, A. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 9(1): 2383, December 2018. ISSN 2041-1723. doi: 10.1038/s41467-018-04316-3. URL http://www.nature.com/articles/s41467-018-04316-3.

Monteiro, R. L. S., Carneiro, T. K. G., Fontoura, J. R. A., da Silva, V. L., Moret, M. A., and Pereira, H. B. d. B. A Model for Improving the Learning Curves of Artificial Neural Networks. *PLOS ONE*, 11(2):e0149874, February 2016. ISSN 1932-6203. doi: 10.1371/journal.pone.0149874. URL https://dx.plos.org/10.1371/journal.pone.0149874.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Platt, G. M., Yang, X.-S., and Silva Neto, A. J. (eds.). *Computational Intelligence, Optimization and Inverse Problems with Applications in Engineering*. Springer International Publishing, Cham, 2019. ISBN 978-3-319-96432-4

978-3-319-96433-1. doi: 10.1007/978-3-319-96433-1. URL http://link.springer.com/10.1007/978-3-319-96433-1.

Poggio, T. It is Compositional Sparsity: a framework for ML. pp. 9, 2022.

Poggio, T., Mhaskar, H., Rosasco, L., Miranda, B., and Liao, Q. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review. *International Journal of Automation and Computing*, pp. 1–17, 03/2017 2017. doi: 10.1007/s11633-017-1054-2. URL http://link.springer.com/article/10.1007/s11633-017-1054-2?wt_mc=Internal.Event.1.SEM.ArticleAuthorOnlineFirst.

Roberts, N., Yap, D. A., and Prabhu, V. U. Deep Connectomics Networks: Neural Network Architectures Inspired by Neuronal Networks. *arXiv:1912.08986 [cs, stat]*, December 2019. URL http://arxiv.org/abs/1912.08986. arXiv: 1912.08986.

Scheffer, L. K., Xu, C. S., Januszewski, M., Lu, Z., Takemura, S.-y., Hayworth, K. J., Huang, G. B., Shinomiya, K., Maitlin-Shepard, J., Berg, S., Clements, J., Hubbard, P. M., Katz, W. T., Umayam, L., Zhao, T., Ackerman, D., Blakely, T., Bogovic, J., Dolafi, T., Kainmueller, D., Kawase, T., Khairy, K. A., Leavitt, L., Li, P. H., Lindsey, L., Neubarth, N., Olbris, D. J., Otsuna, H., Trautman, E. T., Ito, M., Bates, A. S., Goldammer, J., Wolff, T., Svirskas, R., Schlegel, P., Neace, E., Knecht, C. J., Alvarado, C. X., Bailey, D. A., Ballinger, S., Borycz, J. A., Canino, B. S., Cheatham, N., Cook, M., Dreher, M., Duclos, O., Eubanks, B., Fairbanks, K., Finley, S., Forknall, N., Francis, A., Hopkins, G. P., Joyce, E. M., Kim, S., Kirk, N. A., Kovalyak, J., Lauchie, S. A., Lohff, A., Maldonado, C., Manley, E. A., McLin, S., Mooney, C., Ndama, M., Ogundeyi, O., Okeoma, N., Ordish, C., Padilla, N., Patrick, C. M., Paterson, T., Phillips, E. E., Phillips, E. M., Rampally, N., Ribeiro, C., Robertson, M. K., Rymer, J. T., Ryan, S. M., Sammons, M., Scott, A. K., Scott, A. L., Shinomiya, A., Smith, C., Smith, K., Smith, N. L., Sobeski, M. A., Suleiman, A., Swift, J., Takemura, S., Talebi, I., Tarnogorska, D., Tenshaw, E., Tokhi, T., Walsh, J. J., Yang, T., Horne, J. A., Li, F., Parekh, R., Rivlin, P. K., Jayaraman, V., Costa, M., Jefferis, G. S., Ito, K., Saalfeld, S., George, R., Meinertzhagen, I. A., Rubin, G. M., Hess, H. F., Jain, V., and Plaza, S. M. A connectome and analysis of the adult Drosophila central brain. *eLife*, 9:e57443, September 2020. ISSN 2050-084X. doi: 10.7554/eLife.57443. URL https://elifesciences.org/articles/57443.

Shalev-Shwartz, S. and Ben-David, S. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014. doi: 10.1017/CBO9781107298019.

Simard, D., Nadeau, L., and Kröger, H. Fastest learning in small-world neural networks. *Physics Letters A*, 336(1):8–15, February 2005. ISSN 03759601. doi: 10.1016/j.physleta.2004.12.078. URL https://linkinghub.elsevier.com/retrieve/pii/S0375960105000022.

Simonyan, K. and Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv e-prints*, art. arXiv:1409.1556, September 2014.

Stier, J. and Granitzer, M. Structural analysis of sparse neural networks. *Procedia Computer Science*, 159:107–116, 2019. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2019.09.165. URL https://www.sciencedirect.com/science/article/pii/S1877050919313432. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 23rd International Conference KES2019.

Tamassia, R. *Handbook of Graph Drawing and Visualization*. Chapman & Hall/CRC, 1st edition, 2016. ISBN 113803424X.

Towlson, E. K., Vértes, P. E., Ahnert, S. E., Schafer, W. R., and Bullmore, E. T. The rich club of the c. elegans neuronal connectome. *Journal of Neuroscience*, 33(15): 6380–6387, 2013.

Watts, D. J. and Strogatz, S. H. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

Wortsman, M., Farhadi, A., and Rastegari, M. Discovering neural wirings. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/d010396ca8abf6ead8cacc2c2f2f26c7-Paper.pdf.

Xie, S., Kirillov, A., Girshick, R., and He, K. Exploring randomly wired neural networks for image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019a.

Xie, S., Kirillov, A., Girshick, R., and He, K. Exploring Randomly Wired Neural Networks for Image Recognition. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1284–1293, Seoul, Korea

(South), October 2019b. IEEE. ISBN 978-1-72814-803-8. doi: 10.1109/ICCV.2019.00137. URL https://ieeexplore.ieee.org/document/9010992/.

You, J., Leskovec, J., He, K., and Xie, S. Graph Structure of Neural Networks. *arXiv:2007.06559 [cs, stat]*, August 2020. URL http://arxiv.org/abs/2007.06559. arXiv: 2007.06559.

Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=r1Ue8Hcxg.

A. Task Alignment

When converting an undirected network into a DAG using the procedure outlined in Section 3.2, the resulting DAG inherits a fixed number of sources and sinks. However, these numbers may not align with the requirements of a specific computational task, such as the number of features and classes involved in a classification. Consequently, it may be necessary to modify the DAG without altering the underlying undirected graph to match the desired number of sources and sinks. This perturbation process is not applicable to all possible DAGs, and designing an algorithm that guarantees successful perturbation in all cases is challenging. Nonetheless, we have implemented a heuristic approach that accomplishes this task for the experiments conducted. The heuristic relies on four functions that can add or remove one source or sink at a time, and these functions can be repeatedly called. For brevity, we describe the function that transforms a randomly selected source into a hidden node in Algorithm 3. All other functions operate similarly. The algorithm takes as input a topologically sorted DAG, denoted as $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where nodes are identified by their topological indices (i.e., $u_{id}\ \forall\ u \in \mathcal{N}$). It is important to note that when transforming a source into a hidden node, the function may also convert another node into a source. Similar issues can arise with the other functions. Therefore, within the developed heuristic, we call these functions multiple times as needed until the target DAG is obtained or until a predetermined maximum number of iterations is reached.

Algorithm 3 Source Removal

```
**Input:** DAG $\mathcal{G}$ whose nodes are topologically sorted
**Output:** $\mathcal{G}'$
$v \sim \{u : \{(u', u) \in \mathcal{E}\} = \emptyset\}$
$s = \min\{u_{id} : (v, u) \in \mathcal{E}\}$
**for** $u \in \mathcal{N}$ **do**
  **if** $u_{id} > v_{id}$ **and** $u_{id} \leq s$ **then**
    $u_{id} \leftarrow u_{id} - 1$
  **end if**
**end for**
$v_{id} \leftarrow s$
$\mathcal{E}' \leftarrow \emptyset$
**for** $(u, v) \in \mathcal{E}$ **do**
  **if** $u_{id} < v_{id}$ **then**
    $\mathcal{E}' \cup \{(u, v)\}$
  **end if**
  **if** $u_{id} > v_{id}$ **then**
    $\mathcal{E}' \cup \{(v, u)\}$
  **end if**
**end for**
$\mathcal{G}' \leftarrow (\mathcal{N}, \mathcal{E}')$
```