Discovering wiring patterns influencing neural network performance

Aleksandra I. Nowak$^{2[0000-0002-2830-6613]}$ ✉
and Romuald A. Janik$^{1[0000-0002-7782-2737]}$

^{1} Faculty of Mathematics and Computer Science, Jagiellonian University, Łojasiewicza 6, 30-348, Kraków, Poland, nowak.aleksandrairena@gmail.com
^{2} Institute of Theoretical Physics, Jagiellonian University, Łojasiewicza 11, 30-348, Kraków, Poland, romuald.janik@gmail.com

Abstract. The search for optimal neural network architecture is a well-known problem in deep learning. However, as many algorithms have been proposed in this domain, little attention is given to the analysis of wiring properties that are beneficial or detrimental to the network performance. We take a step at addressing this issue by performing a massive evaluation of artificial neural networks with various computational architectures, where the diversity of the studied constructions is obtained by basing the wiring topology of the networks on different types of random graphs. Our goal is to investigate the structural and numerical properties of the graphs and assess their relation to the test accuracy of the corresponding neural networks. We find that none of the classical numerical graph invariants by itself allows to single out the best networks. Consequently, we introduce a new numerical graph characteristic, called *quasi-1-dimensionality*, which is able to identify the majority of the best-performing graphs.

Keywords: Deep Learning · Artificial Neural Networks · Neural Architectures · Network Analysis · Image Classification

1 Introduction

Over the recent years many different neural architectures have been proposed, varying from hand-engineered solutions [23,11,13] to very complicated, automatically generated patterns produced by Neural Architecture Search (NAS) algorithms [31,15,19,6]. However, in this vast panorama of searching methods and benchmarking data, little focus is placed upon analyzing what specific structural properties of the architectures are related to the performance of the network. Apart from studies revolving around residual connections [25] and the impact of width or depth of the network [23,29], we still lack an understanding of why certain wiring topologies work better than others. We believe that addressing this issue would not only increase our knowledge about deep learning systems but also provide guidelines and principles for constructing new, better neural network architectures. Moreover, gathering empirical data linking the graph structure of the information flow with the performance could contribute nontrivial benchmark data for the, yet to be developed, theory.

The aim of this paper is to perform a wide-ranging study of neural network architectures for which the wiring pattern between the blocks of operations is based on a variety of random graphs. We focus on analyzing the interrelation of the structure of the graph with the performance of the corresponding neural network in an image recognition task. This allows us to address some fundamental questions for deep neural networks such as to what extent does the performance of the network depends on the pattern of information flow encoded in its global architecture. Is the performance basically independent of the structure or can we identify *quantitatively* structural patterns which typically yield enhanced performance? The goal of this work is to identify and analyze the discriminative features of neural network architectures. We do not aim at constructing, nor searching for, an optimal architecture.

Another motivation is the observation that artificial neural networks typically have a quite rigid connectivity structure, yet in recent years significant advances in performance have been made through novel global architectural changes like ResNets, [11] or DenseNets [12]. This has been further systematically exploited in the field of Neural Architecture Search (see [6] for a review). Hence there is a definite interest in exploring a wide variety of possible global network structures. On the other hand, biological neural networks in the brain do not have rigid structures and some randomness is an inherent feature of networks that evolved ontogenetically [4]. Contrarily, we also do not expect these networks to be uniformly random [17]. Therefore, it is very interesting to investigate the interrelations of structural randomness and global architectural properties with the network's performance.

To this end, we explore a wide variety of neural network architectures for an image recognition task, constructed accordingly to wiring topologies defined by random graphs. This approach can efficiently produce many qualitatively different connectivity patterns by alternating only the random graph generators [28]. The nodes in the graph correspond to a simple convolutional computational unit, whose internal structure is kept fixed. Apart from that, we do not impose any restrictions on the overall structure of the neural network. In particular, the employed constructions allow for modeling arbitrary global (as well as local) connectivity. We investigate a very diversified set of graph architectures, which range from the quintessential random, scale-free, and small-world families, through some edge-direction sensitive constructions, to graphs based on fMRI data. Altogether we conduct an analysis of more than 1000 neural networks, each corresponding to a different directed acyclic graph[3]. Such a wide variety of graphs is crucial for our goal of analyzing the properties of the network architecture by studying various characteristics of the corresponding graph and examining their impact on the performance of the model.

We find that among more than 50 graph-theoretical properties tested by us in this study, none is able to distinguish, by itself, the best performing graphs. We are able to identify one group of the top graphs by introducing a new numerical graph criterion, which we refer to as *quasi-1-dimensionality*. This

[3] The code is available at `https://github.com/rmldj/random-graph-nn-paper`

criterion captures graphs characterized by mostly local connections with a global elongated structure, providing guidelines for beneficial biases in the architectural design of neural networks.

2 Related Work

Neural Architecture Search. Studies undertaken over the recent years indicate a strong connection between the wiring of network layers and its generalization performance. For instance, ResNet introduced by [11], or DenseNet proposed in [12], enabled successful training of very large multi-layer networks, only by adding new connections between regular blocks of convolutional operations. The possible performance enhancement that can be gained by the change of network architecture has posed the question, whether the process of discovering the optimal neural network topology can be automatized. In consequence, many approaches to this Neural Architecture Search (NAS) problem were introduced over the recent years [6]. Among others, algorithms based on reinforcement learning [31,2], evolutionary techniques [19] or differentiable methods [15]. Large benchmarking datasets of the cell-operation blocks produced in NAS have been also proposed by [29] and extended by [5].

The key difference between NAS approaches an the present work is that we are not concentrating on directly optimizing the architecture of a neural network for performance, but rather on exploring a wide variety of random graph architectures in order to identify what features of a graph are related to good or bad performance of the associated neural network. Thus we need to study both strong and weak architectures in order to ascertain whether a given feature is, or is not *predictive* of good performance. We hope that our findings will help to develop new NAS search spaces.

Random Network Connectivity. There were already some prior approaches which focused on introducing randomness or irregularity into the network connectivity pattern. The work of [21] proposed stochastic connections between consecutive feed-forward layers, while in [13] entire blocks of layers were randomly dropped during training. However, the first paper which, to our knowledge, really investigated neural networks on random geometries was the pioneering work of [28]. This paper proposed a concrete construction of a neural network based on a set of underlying graphs (one for each resolution stage of the network). Several models based on classical random graph generators were evaluated on the ImageNet dataset, achieving competitive results to the models obtained by NAS or hand-engineered approaches. Using the same mapping, [20] investigated neural networks based on the connectomics of the mouse visual cortex and the biological neural network of *C.Elegans*, obtaining high accuracies on the MNIST and FashionMNIST datasets.

Although the works discussed above showed that deep learning models based on random or biologically inspired architectures can indeed be successfully trained without a loss in the predictive performance, they did not investigate what kind of graph properties characterize the best (and worst) performing topologies.

The idea of analyzing the architecture of the network by investigating its graph structure has been raised in [30]. However, this work focused on exploring the properties of the introduced relational graph, which defined the communication pattern of a network layer. Such a pattern was then repeated sequentially to form a deep model. In addition, [16] have also analyzed machine learning models with the tools of network science, but their research was devoted to Restricted Boltzmann Machines.

The main goal of our work is to perform a detailed study of numerical graph characteristics in relation to the associated neural network performance. Contrary to [30] we are not concentrating on exploring the fine-grained architecture of a layer in a sequential network. Instead, we keep the low-level operation pattern fixed and encapsulated in the elementary computational node. We focus on the high-level connectivity of the network, by analyzing the graph characteristics of neural network architectures based on arbitrary directed acyclic graphs.

3 From a Graph to a Neural Network

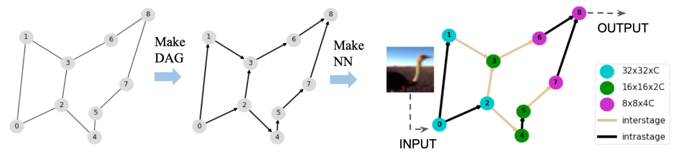
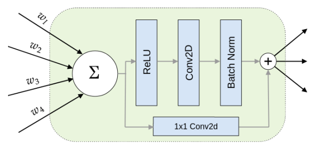


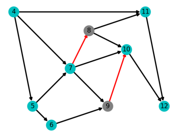
Fig. 1: The graph to neural network mapping. First, a graph is sampled from a predefined set of random graph generators. Next, the graph is transformed to a DAG by selecting a node ordering and enforcing the connections to be oriented accordingly to that ordering. Such DAG is treated as a blueprint for a neural network architecture. Nodes with different colors work on different resolutions of the feature maps. The beige (*interstage*) edges indicate the connections on which a resolution reduction is performed. The black edges (*intrastage*) link nodes that work within the same resolution. Best viewed in color.

In order to transform a graph into a neural network, we adopt the approach presented in [28]. In that paper, a graph is sampled from a predefined list of generators and transformed into a directed acyclic graph (DAG). Next, the DAG is mapped to a neural network architecture as follows:

The edges of the graph represent the flow of the information in the network and the nodes correspond to the operations performed on the data. The computation is performed accordingly to the topological order. In each node, the input from the ingoing edges is firstly aggregated using a weighted sum. Next, a ReLU – Conv2d – Batch-Norm block is applied. The result of this procedure is then propagated independently by each outgoing edge. When the computations leaves



(a) The node architecture



(b) *orphan* nodes

Fig. 2: **(a)**: The node is represented by the green-shaded area. The black arrows illustrate the graph edges labeled with the associated weights. The gray arrows indicate the ordering of the operations performed in the node as well as the residual connection. **(b)**: The gray nodes (orphan nodes) in the DAG either do not have an input from previous stages of processing or do not have an output. Hence we add the red edges from the immediately preceding node or to the immediately succeeding node.

the last node, a global average pooling is performed, followed by a dense layer with the number of output neurons equal to the target dimension.

The network nodes are divided into three sets of equal size, referred to as *stages* (denoted by different colors in the figures). The first stage operates on the original input resolution, with the number of channels C being set in the first (input) node of the graph. The subsequent stages operate on a decreased input resolution and increased number of output channels by a factor of 2, with respect to the previous stage. In order to perform the downsampling, on every edge that crosses two stages the same block of operations as in a standard node is executed, but with the use of convolutions with stride 2 (when crossing subsequent stages) or stride 4 (when crossing from the first stage to the last). In the figures in the present paper, we represent such resolution-changing edges with beige color, and refer to them as *interstage*. See Fig. 1 for a visualization of the above described mapping. We introduce three modifications to this procedure:

Firstly, in [28] there were separate random graphs for each of the three stages of the neural network. This means that subsequent stages were connected only by one edge. In our case, we have a random graph for the *whole* network. Dimensionality reduction is performed on a graph edge when necessary, by a node with stride 2 or 4 convolution, as described above. In consequence, we do not bias the model to have a single bottleneck connection between the computations performed on different spatial resolutions. Moreover, we observe that the introduction of such a bottleneck generally deteriorates the network performance (we discuss this issue in Section 6.2).

Secondly, we introduce an additional residual connection from the aggregated signal to the output of the triplet block in the node. The residual connection always performs a projection (implemented by a 1×1-convolution, similar to ResNet C-type connections of [11] — see Figure 2a). The residual skip connection shifts the responsibility of taking care of the vanishing gradient problem from edges to the nodes, allowing the global connectivity structure to focus on the information flow, with the low-level benefits of the residual structure already built-in.

Thirdly, we improve the method of transforming a graph into a DAG so that it automatically takes into account the graph structure. This is achieved by ordering the nodes accordingly to a 2D Kamada-Kawai embedding [14] and setting the directionality of an edge from the lower to the higher node number. Any arising orphan nodes like the ones in Figure 2b are then fixed by adding a connection from the node with the preceding number or adding a connection to the node with the succeeding number. We observe that this approach leads to approximately 2x fewer orphan nodes than the random ordering, and circa 1.5x less than the original ordering returned by the generator, which was used in [28]. A detailed description of the DAG transformation process together with a comparison of various node orderings can be found in Appendix B and C.

4 The Space of Random Graphs and DAGs

We performed a massive empirical study of over 1000 neural network architectures based on 5 graph families and 2 auxiliary constructions. We summarize below their main characteristics.

- *Erdős-Rényi* (`er`) – In this model, given a parameter $p \in [0, 1]$, each possible (undirected) edge arises independently of all the other edges with probability p [7]. Small p usually results in sparse graphs, while increasing p increases also the chance of obtaining a graph with dense connections.
- *Barabási-Albert* (`ba`) – Given a set of m initially connected nodes, new nodes are added to the graph iteratively. In each step, a new node is connected with at most m other nodes with probability proportional to the nodes' degrees. The Barabási-Albert model favors the formation of hubs, as the few nodes with a high degree are more likely to get even more connections in each iteration. Therefore graphs produced by this model are associated with scale-free networks [3].
- *Watts-Strogatz* (`ws`) – The Watts-Strogatz model starts with a regular ring of nodes, where each of the nodes is connected to k of its nearest neighbors. Then, iteratively, every edge (u, v) which was initially present in the graph is replaced with probability p by an edge (u, w), where the node w is sampled uniformly at random from all the other nodes. The graphs obtained by this method tend to have the small-world property [26].
- *Random-DAG* (`rdag`) – The models mentioned so far produce undirected graphs, which need to be later transformed to DAGs. We choose to also study models produced by an algorithm that directly constructs a random DAG. An advantage of this algorithm over other existing DAG-generating methods is that it allows to easily model neural networks with mostly short-range or mostly long-range connections, which was the main reason for implementing this construction. This procedure and its parameters are thoroughly explained in Section 4.1.

- *fMRI based* (`fmri`) – In addition to the above algorithmic generators we also introduce a family of graphs that are based on resting-state functional MRI data. We use the network connectomes provided by the Human Connectome Project [24] obtained from the resting-state fMRI data of 1003 subjects [22]. As input for graph construction, we use the released (z-score transformed) partial correlation matrix for 50- and 100-component spatial group-ICA parcellation. We describe in detail the exact method of deriving DAGs from the fMRI partial correlation matrices in Appendix D. Apart from the number of nodes, this family has a single thresholding parameter.

Moreover, we considered two auxiliary types of graphs:

- *Bottleneck graphs* (`bottleneck`) – For some graphs from the above families, we introduced a bottleneck between the various resolution stages (see Section 6.2).
- *Composite graphs* (`composite`) – We obtained these graphs by maximizing in a Monte-Carlo simulation the expression $\left(\frac{log\_num\_paths}{num\_nodes}\right)^{\frac{1}{2}} - 2grc - avg\_clustering$ where grc is the global reaching centrality of the graph. This construction was motivated by a certain working hypothesis investigated at an early stage of this work which was later discontinued. Nevertheless, we kept the graphs for additional structural variety.

For each of the above families, we fix a set of representative parameters[4]. Then for every family-parameters pair, we sample 5 versions of the model by passing different random seeds to the generator. Using this procedure we create 475 networks with 30 nodes and 545 networks with 60 nodes. We train all networks for 100 epochs with the same settings on the CIFAR-10 dataset[5]. For each network, we set the number of initial channels C in order to obtain approximately the same number of parameters as in ResNet-56 (853k).

4.1 Direct Construction of Random DAGs

In order to study some specific questions, like the role of long-range versus short-range connectivity, we implement a procedure for directly constructing random DAGs that allows for more fine-grained control than the standard random graph generators and is flexible enough to generate various qualitatively different kinds of graph behaviors. As an additional benefit, we do not need to pass through the slightly artificial process of transforming an arbitrary undirected graph to DAG.

We present the method in Algorithm 1. We start with N nodes, with a prescribed ordering given by integers $0, \ldots, N-1$. For each node i, we fix the number of outgoing edges n_i^{out} (clearly $n_i^{out} < N - i$). Here we have various choices leading to qualitatively different graphs. For example, sampling n_i^{out} from a Gaussian and rounding to a positive integer (or setting n_i^{out} to a constant) would yield approximately homogeneous graphs. Taking a long-tailed distribution

[4] Refer to Appendix I for a full list.

[5] We provide a full description of the training procedure in Appendix A.

Algorithm 1 Random DAG

```
1: Input: nodes $i = 0, \dots, N-1$,
          number of outgoing edges $n_i^{out}$,
          size of a local neighbourhood $B$,
          real $\alpha$, function $f(x)$
2: for $i = 0$ to $N-2$ do
3:     if node $i+1$ does not have an ingoing connection then
4:         make an edge $i \to i+1$
5:     end if
6:     while not all $n_i^{out}$ outgoing edges chosen do
7:         Make randomly the edge $i \to j$ with probability $p_j = \frac{w_{ij}}{\sum_{k>i} w_{ik}}$
8:         where the weight $w_{ij}$ is given by
9:         $w_{ij} = (n_j^{out})^\alpha f\left(\lfloor \frac{j-i}{B} \rfloor\right)$
10:        provided $j > i$ and $i \to j$ does not exist so far
11:    end while
12: end for
```

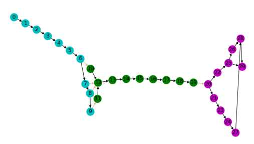
would yield some outgoing hubs. One could also select the large outgoing hubs by hand and place them in a background of constant and small n_i^{out}.

For each node i we then randomly choose (with weight w_{ij} given in Algorithm 1) nodes $j > i$ to saturate the required n_i^{out} connections. The freedom in the choice of weight w_{ij} gives us the flexibility of preferential attachment through the parameter α, and the possibility of imposing local or semi-local structure through the choice of the weighting function $f\left(\lfloor \frac{j-i}{B} \rfloor\right)$.

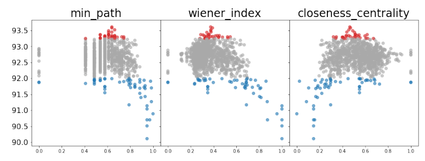
Different choices of f lead to different connectivity structures of the DAG. An exponential $f(x) = \exp(-Cx)$ results in short-range connections and local connectivity. The power law scaling $f(x) = 1/x$ produces occasionally longer range connections, while $f(x) = 1$ does not imply any nontrivial spatial structure at all. In this work, we investigated all three of the above possibilities. Since we do not want the integer node labels i or j to be effectively a 1d coordinate, we define a local neighborhood size B so that differences of node labels of order B would not matter. This motivates the form of the argument of the weighting function $f(x) \equiv f\left(\lfloor \frac{j-i}{B} \rfloor\right)$, where $\lfloor a \rfloor$ denotes the floor of a. In the simulations we set $B = 5$ or $B = 10$.

Through the choice of the function $f(.)$, we can model graphs with varying proportions of short- to long-range connections with the parameter B defining the size of the local neighborhood. The choice of multiplicity distribution of n_i^{out} allows to model, within the same framework, a uniform graph, a graph with power law outgoing degree scaling, or a graph with a few hubs with very high multiplicity. Finally, the parameter α enables to control preferential attachment of the connections. Consequently, the algorithm allows to produce DAGs with diverse architectural characteristics well suited for neural network analysis.

Let us note that the presented procedure is somewhat similar to the *latent position random graph* model [1] with graph features $X_i = (i, n_i^{out})$ and kernel



(a)



(b)

Fig. 3: **(a)**: One of the worst networks with 30 nodes. The worst networks are typically characterized by sparse connections and long chains of operations. For more examples of the worst networks see Appendix J. **(b)**: The test accuracy versus selected network features. We indicate the best (equal or above 93.25%) models as red, the worst (below 92%) as blue, and the rest as gray. The first presented feature is the length of the shortest path between the input and the output node. The second one is the Wiener index [27], and the third is closeness centrality [8]. All features are rescaled using min-max scaling (For more details on data processing refer to Appendix F).

$\kappa(X_i, X_j) = (n_j^{out})^\alpha f\left(\lfloor \frac{j-i}{B} \rfloor\right)$. However, in such a case, the kernel would still need to be normalized by the weights of all nodes smaller than j, as in line 7 of Algorithm 1. Moreover, such formulation could produce DAGs with disconnected components, whereas in our setting we ensure that every node $n > 0$ has an incoming connection (see lines 3-4).

5 Results

In this Section, we first exhibit the inadequacy of classical graph invariants to select the best performing networks and describe the generic features of the worst networks. Then we introduce a class of well-performing networks (which we call *quasi-1-dimensional* or Q1D) and provide their characterization in terms of a novel numerical graph invariant.

5.1 The Inadequacy of Classical Graph Characteristics

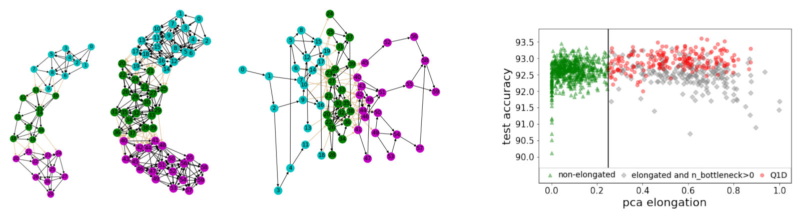
The key motivation for this work was to understand what features of the underlying graph are correlated with the test performance of the corresponding neural network. To this end, for the analysis, we use 54 graph features, mostly provided by the `networkx` library [10] as well as some simple natural ones, like the logarithm of the total number of paths between the input and output or the relative number of connections between stages with various resolutions. For a full list of the features see Appendix F.

It turns out that none of the classical features by itself is enough to isolate the best-performing networks. However, the worst networks form outliers for several of the tested graph properties and thus can be more or less identified (see Figure 3b for a representative example and more plots in the Appendix H).

5.2 The Worst Networks

As mentioned before, several investigated network features seem to be able to discriminate the worst networks. In Figure 3b we present a selected set of such features: the minimal length of a path between the input and output node, the Wiener index [27], and the closeness centrality [8] of the output node. The Wiener index is the sum of the lengths of all-pair shortest paths and the closeness centrality of a node is the reciprocal of the average shortest path distance to that node.

The above properties show that the worst networks are usually characterized by long distances between any two nodes in the graph, resulting in long chains of operations and sparse connections. An example of such a graph is presented in Figure 3a. In addition, we verified that purely sequential 1d chain graphs (node i is connected only to node $i+1$) gave indeed the worst performance.



(a) (b)

Fig. 4: **(a)**: The best network with 30 nodes (left), with 60 nodes (center) and an example of a highly ranked fMRI based network. For more examples of the best networks see Appendix J. **(b)**: The visualization of the Q1D criterion. The green triangles indicate graphs without a global elongated structure and the gray diamonds are used to represented the elongated graphs with bottlenecks. Networks with Q1D property are drawn as red dots. The black vertical line illustrates the threshold $\tau = 0.25$. The Q1D criterion for this threshold successfully selects the best networks from the elongated group.

5.3 The Best Networks

Crucial to our results is the observation that the best networks belonged predominantly to the Random DAG category with short-range connections (i.e. exponential $f(x)$). One generic visual feature of these graphs is that they have a definite global ordering in the feed-forward processing sequence defining the 1d structure, yet locally there are lots of interconnections that most probably implement rich expressiveness of intermediate feature representations (see the first two graphs in Figure 4a). We call such graphs *quasi-1-dimensional* (quasi-1d for short). These models have a very large number of paths between the input and the output. This is, however, not the feature responsible for good performance, as maximally connected DAGs that have the maximal possible number of paths do not fall into this category and give worse results (see Figure 8 in Appendix). In contrast, filament-like, almost sequential models such as some Watts-Strogatz networks (recall Figure 3a) have in fact significantly worse performance, so sequentiality by itself also does not ensure good generalization.

We would like to formally characterize these graphs purely in terms of some numerical graph features without recourse to their method of construction. This is not *a priori* a trivial task. This is because one has to be sensitive to the globally elongated structure. However, the filament-like, almost sequential graphs are quite similar in this respect, yet they yield very bad performance. So numerical graph properties which are positively correlated with a stretched topology tend to have similar or even larger values for the very bad graphs. A condition that can eliminate the filament-like graphs is $n_{bottlenecks} = 0$, where a bottleneck edge is defined by the property that cutting that edge would split the graph into two separate components.

In order to numerically encode the elongated character of a network, we perform PCA on the set of node coordinates returned by the Kamada-Kawai embedding and require a sufficiently anisotropic explained variance ratio. Note that despite appearances this is a quite complex invariant of the original abstract graph, as the Kamada-Kawai embedding depends on the whole global adjacency structure through the spring energy minimization. Hence the nature of the embedding encodes nontrivial relevant information about the structure of the graph. We define then the elongation of the network as

$$pca\_elongation = 2 \cdot (variance\_ratio - 0.5), \tag{1}$$

where $variance\_ratio$ is the percentage of the variance explained by the component corresponding to the largest eigenvalue computed during the PCA decomposition. Networks with very large $pca\_elongation$ tend to have only one main computational path, while small $pca\_elongation$ is associated with graphs with many global inter-connections. For instance, almost purely sequential graphs have $pca\_elongation$ close to 1.0, while for fully connected DAGs this property is equal to 0. In order to use this continuous feature to define a discrete class of graphs with a visible hierarchical structure of the Kamada-Kawai embedding we need to specify a threshold τ and consider only graphs for which $pca\_elongation > \tau$. Accordingly, we formally define the quasi-1d graphs (Q1D) as satisfying the condition:

$$pca\_elongation > \tau \quad \text{and} \quad n_{bottlenecks} = 0, \tag{2}$$

This condition is visualized in Figure 4b. The first term of the Q1D definition accounts for networks that have a global one-dimensional order (like the two first networks in Figure 4a). The second condition eliminates graphs containing bottlenecks which form the bulk of badly performing elongated graphs (denoted by gray dots in Figure 4b). In our analysis, we set $\tau = 0.25$, which is a visual estimate motivated by Figure 4b. This value is of course not set in stone and could just as well be a bit higher or lower. The rough choice of τ is also corroborated

Table 1: For each graph family we report in percentage the number of all graphs having Q1D property, the number of graphs in top-50, the share of the given family in top-50 and the number of Q1D graphs within the ones present in top-50, followed by analogous statistics for the graphs in bottom-50. The Q1D criterion selects almost every best performing `rdag` and more than half `fmri` graphs (fourth column), which are the majority in top-50 (second column). None of the worst performing graph satisfies the Q1D criterion (last column).

MODEL	WITH Q1D PROPERTY	IN TOP-50	SHARE IN TOP-50	Q1D TOP-50	IN BOT-50	SHARE IN BOT-50	Q1D BOT-50
`ba`	0.00	4.00	4.00	0.00	0.00	0.00	0.00
`bottleneck`	0.00	0.67	2.00	0.00	6.67	20.00	0.00
`composite`	0.00	0.00	0.00	0.00	0.00	0.00	0.00
`er`	1.33	2.67	4.00	0.00	2.67	4.00	0.00
`fmri`	32.86	12.86	18.00	55.56	1.43	2.00	0.00
`rdag`	66.98	13.95	60.00	93.33	0.93	4.00	0.00
`ws`	7.05	1.36	12.00	16.67	7.95	70.00	0.00
`all`	19.50	4.90	–	**68.00**	4.90	–	**0.00**

by the fact that the Q1D criterion for this threshold is strongly correlated with performance, as we discuss below.

We find that among the top 50 networks, 68% have the Q1D property. Moreover, out of the remaining 970 graphs, only 17% are Q1D. A breakdown of the top-50 and bottom-50 by specific graph families and the Q1D property is presented in Table 1. One may observe that Q1D successfully selects almost every of the best performing `rdags` and half of the `fmri` graphs (fourth column). Those two families are also the most representative among the top-50. Furthermore, *none* of the graphs in the bottom-50 has the Q1D property (last column).

The Q1D criterion is able to single out *one type* of the best performing networks, being at the same time *agnostic* about the details of the graph generation procedure. This is especially important considering the failure of classical graph features in this regard.

Finally, let us also mention that there are some qualitatively different networks (see for example the `fmri` network in Figure 4a) in the `fmri` class as well as in the `ba` class, which achieve good performance. Those networks are often not elongated (as indicated by several green points with high test accuracy in Figure 4b) and therefore do not satisfy the Q1D criterion. It seems, however, quite difficult to identify a numerical characterization that would pick out the best networks from this category.

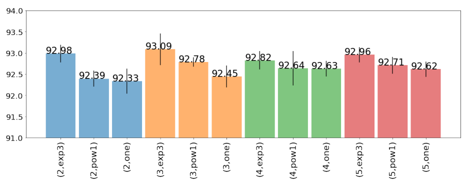


Fig. 5: The CIFAR-10 test accuracy averaged over different versions (random seeds) of random DAG models with 30 nodes and constant number (2-5) of output edges n_i^{out}. The symbol *exp3* stands for exponential weighting function $f(x)$, *pow1* for a power law and *one* for a constant. It may be observed that the networks with primarily local connections (*exp3* - the first bar in each set) have the best performance.

6 Impact on Architecture Design

The key components of the Q1D graphs are elongated structure and lack of bottlenecks. In this section we further analyze the importance of those characteristics as guidelines in the design of neural network connectivity. We start with a study of the effect of short- vs. long-range connections in the `rdag` graphs and follow with a commentary about the role of many resolution-changing pathways. Finally, we also perform a comparison of the CIFAR-10 results with results on CIFAR-100 in order to ascertain the consistency of the identification of the best and worst-performing network families.

6.1 Long- vs. Short-range Connections

The algorithm for directly generating random DAGs allows for modifying, in a controllable way, the pattern of long- versus short-range connectivity. This is achieved by changing the function $f(x)$ from an exponential, leading to local connections, through a power law, which allows for occasional long-range connections, to a constant function, which does not impose any spatial order and allows connections at all scales. The results are presented in Figure 5. We observe that within this class of networks the best performance comes from networks with primarily short-range connections and deteriorates with their increasing length.

This may at first glance seem counter-intuitive, as skip connections are typically considered beneficial. However, the effect of long-range connections which is associated with easier gradient propagation is already taken care of by the residual structure of each node in our neural networks (recall Section 3). One can understand the deterioration of the network performance with the introduction of long-term connections as coming from an inconsistency

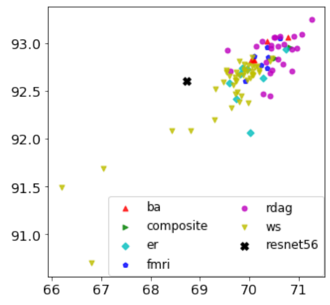


Fig. 6: The CIFAR-10 (y-axis) and CIFAR-100 (x-axis) test accuracies. Each datapoint contains results averaged over the random versions of the models. The results are strongly correlated, yielding Pearson correlation coefficient equal to 0.868.

of the network with the natural hierarchical semantic structure of images. This result leads also to some caution in relation to physical intuition from critical systems where all kinds of power law properties abound. The dominance of short-range over long-range connections is also consistent with the good performance of *quasi-1-dimensional* networks as discussed in Section 5.3.

6.2 Influence of Bottlenecks

As noted in Section 3, one difference between the networks of [28] and our construction was that in the former case, there were separate random graphs for each processing stage of a specific resolution, which were connected with a single gateway. In our case, we have a single graph, which encompasses all resolutions. Thus generally there are many independent resolution-reducing edges in the network instead of a single one. In order to verify whether such a single gateway between different resolutions is beneficial or not, for a selected set of graphs, we artificially introduced such a bottleneck by first erasing all inter-resolution edges. Next, we create a single edge from the last node in the preceding stage to the first node in the consequent stage and then fixing possible orphans as in Figure 2b[6]. We found that, systematically, the introduction of a bottleneck deteriorates performance (see Figure 8. in Appendix). Hence multiple resolution reduction pathways are beneficial. Let us note that this result is coherent with our findings from Section 5.3, where bottleneck edges (also within a single resolution stage) typically appear in badly performing networks.

6.3 CIFAR-10 versus CIFAR-100 Consistency

In addition to the CIFAR-10 task, we trained all networks with 60 nodes (except for the bottleneck ablations) on the CIFAR-100 dataset. We used the same training procedure as the one for CIFAR-10. The motivation for this experiment was to verify whether the graph families which performed best in the first problem achieve also high results in the second. Indeed, we observe a significant correlation 0.868 (see Figure 6) between the respective test accuracies (averaged over the 5 random realizations of each graph type). Especially noteworthy is the consistency between the groups of best and worst graphs for the two datasets.

7 Conclusions and Outlook

We have performed an extensive study of the performance of artificial neural networks based on random graphs of various types, keeping the training protocol

[6] See a visualization of a bottleneck graph in Appendix I

fixed. One class of networks which had the best performance in our simulations were networks, which could be characterized as quasi-1-dimensional, having mostly local connections with a definite 1-dimensional hierarchy in data processing (one can dub this structure as *local chaos and global order*). These were predominantly networks in the `rdag` family. We also introduced a very compact numerical characterization of such graphs. It is worth noting, that some of the fMRI-based graphs were also among the best-performing ones (together with some `ws` and `ba` ones). We lack, however, a clear-cut numerical characterization of these "good" graphs as there exist graphs with apparently similar structure and numerical invariants but much worse performance.

Among other structural observations made in this project, we noted that long-range connections were predominantly negatively impacting network performance. Similarly, artificially imposing a bottleneck between the processing stages of various resolutions also caused the results to deteriorate. Thus, a general guideline in devising neural network architectures which can be formed in consequence of our study is to prefer networks with rich local connections composed into an overall hierarchical computational flow, with multiple resolution-reducing pathways and no bottleneck edges. These characteristics seem to consistently lead to good performance among the vast panorama of connectivity patterns investigated in the present paper.

Acknowledgments Work carried out within the research project *Bio-inspired artificial neural network* (grant no. POIR.04.04.00-00-14DE/18-00) within the Team-Net program of the Foundation for Polish Science co-financed by the European Union under the European Regional Development Fund. The fMRI data were provided by the Human Connectome Project, WU-Minn Consortium (PIs: David Van Essen and Kamil Ugurbil; 1U54MH091657) funded by 16 NIH Institutes and Centers that support the NIH Blueprint for Neuroscience Research; and by the McDonnell Center for Systems Neuroscience at Washington University.

References

1. Athreya, A., Fishkind, D.E., Tang, M., Priebe, C.E., Park, Y., Vogelstein, J.T., Levin, K., Lyzinski, V., Qin, Y.: Statistical inference on random dot product graphs: a survey. The Journal of Machine Learning Research **18**(1), 8393–8484 (2017)
2. Baker, B., Gupta, O., Naik, N., Raskar, R.: Designing neural network architectures using reinforcement learning. In: International Conference on Learning Representations (2016)
3. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science **286**(5439), 509–512 (1999)
4. Bullmore, E.T., Bassett, D.S.: Brain graphs: graphical models of the human brain connectome. Annual review of clinical psychology **7**, 113–140 (2011)
5. Dong, X., Yang, Y.: Nas-bench-201: Extending the scope of reproducible neural architecture search. In: International Conference on Learning Representations (2019)
6. Elsken, T., Metzen, J.H., Hutter, F.: Neural architecture search: A survey. Journal of Machine Learning Research **20**(55), 1–21 (2019)

7. Erdős, P., Rényi, A.: On the evolution of random graphs. Publ. Math. Inst. Hungar. Acad. Sci **5**, 17–61 (1960)
8. Freeman, L.C.: Centrality in social networks conceptual clarification. Social networks **1**(3), 215–239 (1978)
9. Fruchterman, T.M., Reingold, E.M.: Graph drawing by force-directed placement. Software: Practice and experience **21**(11), 1129–1164 (1991)
10. Hagberg, A., Swart, P., S Chult, D.: Exploring network structure, dynamics, and function using networkx. Tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States) (2008)
11. 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. pp. 770–778 (2016)
12. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)
13. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: European conference on computer vision. pp. 646–661. Springer (2016)
14. Kamada, T., Kawai, S.: An algorithm for drawing general undirected graphs. Information Processing Letters **31**(1), 7–15 (1989)
15. Liu, H., Simonyan, K., Yang, Y.: Darts: Differentiable architecture search. In: International Conference on Learning Representations (2019)
16. Mocanu, D.C., Mocanu, E., Nguyen, P.H., Gibescu, M., Liotta, A.: A topological insight into restricted boltzmann machines. Machine Learning **104**(2), 243–270 (2016)
17. Orsini, C., Dankulov, M.M., Colomer-de Simón, P., Jamakovic, A., Mahadevan, P., Vahdat, A., Bassler, K.E., Toroczkai, Z., Boguná, M., Caldarelli, G., et al.: Quantifying randomness in real networks. Nature communications **6**(1), 1–10 (2015)
18. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. arXiv preprint arXiv:1912.01703 (2019)
19. Real, E., Aggarwal, A., Huang, Y., Le, Q.V.: Regularized evolution for image classifier architecture search. In: Proceedings of the AAAI conference on artificial intelligence. vol. 33, pp. 4780–4789 (2019)
20. Roberts, N., Yap, D.A., Prabhu, V.U.: Deep connectomics networks: Neural network architectures inspired by neuronal networks. arXiv preprint arXiv:1912.08986 (2019)
21. Shafiee, M.J., Siva, P., Wong, A.: Stochasticnet: Forming deep neural networks via stochastic connectivity. IEEE Access **4**, 1915–1924 (2016)
22. Smith, S.M., Beckmann, C.F., Andersson, J., Auerbach, E.J., et al.: Resting-state fmri in the human connectome project. NeuroImage **80**, 144–168 (2013)
23. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1–9 (2015)
24. Van Essen, D.C., Smith, S.M., Barch, D.M., Behrens, T.E., Yacoub, E., Ugurbil, K.: The wu-minn human connectome project: An overview. NeuroImage **80**, 62–79 (2013)
25. Veit, A., Wilber, M., Belongie, S.: Residual networks behave like ensembles of relatively shallow networks. arXiv preprint arXiv:1605.06431 (2016)
26. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world'networks. Nature **393**(6684), 440 (1998)

27. Wiener, H.: Structural determination of paraffin boiling points. Journal of the American chemical society **69**(1), 17–20 (1947)
28. Xie, S., Kirillov, A., Girshick, R., He, K.: Exploring randomly wired neural networks for image recognition. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1284–1293 (2019)
29. Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K., Hutter, F.: Nas-bench-101: Towards reproducible neural architecture search. In: International Conference on Machine Learning. pp. 7105–7114 (2019)
30. You, J., Leskovec, J., He, K., Xie, S.: Graph structure of neural networks. In: Proceedings of the International Conference on Machine Learning (2020)
31. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. In: International Conference on Learning Representations (2017)