ON THE RELATIONSHIP BETWEEN TOPOLOGY AND GRADIENT PROPAGATION IN DEEP NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we address two fundamental research questions in neural architecture design: (*i*) How does the architecture topology impact the gradient flow during training? (*ii*) Can certain topological characteristics of deep networks indicate *a priori* (*i.e.*, without training) which models, with a different number of parameters/FLOPS/layers, achieve a similar accuracy? To this end, we formulate the problem of deep learning architecture design from a network science perspective and introduce a new metric called *NN-Mass* to quantify how effectively information flows through a given architecture. We establish a theoretical link between NN-Mass, a topological property of neural architectures, and gradient flow characteristics (*e.g.*, Layerwise Dynamical Isometry). As such, NN-Mass can identify models with similar accuracy, despite having significantly different size/compute requirements. Detailed experiments on both synthetic and real datasets (*e.g.*, MNIST, CIFAR-10, CIFAR-100, ImageNet) provide extensive evidence for our insights. Finally, we show that the closed-form equation of our theoretically grounded NN-Mass metric enables us to *design* efficient architectures directly without time-consuming training and searching.

1 INTRODUCTION

Recent research in neural architecture design has driven several breakthroughs in deep learning. Specifically, major contributions have been made in the following two directions: (*i*) Initialization of model weights [LeCun et al. (2012); Glorot & Bengio (2010); He et al. (2015)], and (*ii*) Network topology showing how different compute units (*e.g.*, neurons, channels, layers) should be connected to each other [He et al. (2016); Huang et al. (2017); Howard et al. (2017); Sandler & et al. (2018); Liu et al. (2018); Zoph et al. (2018)]. While many attempts have been made to study the impact of initialization on model accuracy [Poole et al. (2016); Tarnowski et al. (2018); Pennington et al. (2017)], good Deep Neural Network (DNN) topologies have been mainly developed either manually (*e.g.*, Resnets, Densenets, *etc.* [He et al. (2016); Huang et al. (2017); Howard et al. (2017); Sandler & et al. (2018)]) or automatically using Neural Architecture Search (NAS) techniques [Liu et al. (2018); Zoph et al. (2018); Tan et al. (2019); Cai et al. (2018)]. However, the impact of topological properties of DNNs on model performance has *not* been explored systematically. Hence, there is a significant gap in our understanding of how exactly various topological properties impact the gradient flow and accuracy of DNNs.

In general, the topology (or structure) of networks strongly influences the phenomena taking place over them [Newman et al. (2011)]. For instance, how closely the users of a social network are connected to each other directly affects how fast the information propagates through the network. Similarly, a DNN architecture can be seen as a network of different neurons connected together. Therefore, the topology of deep networks can influence how effectively the gradients propagate and, hence, how much information can be learned. Indeed, this means that even models with significantly different size/compute requirements, but similar topological properties can achieve similar accuracy.

Motivated by the need for understanding (*i*) the relationship between gradient flow and topology, and (*ii*) properties of efficient models, we address the following fundamental questions:

1. How does the DNN topology influence the gradient flow through the network?
2. Can topological properties of DNNs indicate *a priori* (*i.e.*, without training) which models achieve a similar accuracy, despite having vastly different #parameters/FLOPS/layers?

To answer these questions, we first model DNNs as complex networks (in order to exploit the network science – the study of networks) and quantify their topological properties. To this end, we propose

a new metric called *NN-Mass* that explains the relationship between the topological structure of DNNs and the Layerwise Dynamical Isometry (LDI) [Lee et al. (2020)], a property that indicates the faithful gradient propagation through the network [Saxe et al. (2013)]. Specifically, models with similar NN-Mass should have similar LDI, and thus a similar gradient flow that results in comparable accuracy. To support these theoretical insights, we conduct a thorough exploration on different types of networks (MLPs/CNNs/Depthwise Convolutions, etc.) and several datasets (MNIST, CIFAR-10, CIFAR-100, Imagenet). We show that models with the same width and NN-Mass indeed achieve a similar accuracy irrespective of their depth, number of parameters, and FLOPS. Finally, after extensive experiments linking topology and gradient flow, we show how the closed-form expression for NN-Mass can be used to *directly* design efficient deep networks without *any* training and searching. Overall, we propose a new theoretically grounded perspective that reveals how topology influences the gradient propagation in deep networks. Of note, our work does *not* belong to the family of NAS approaches since our goal is **not** to beat any existing NAS solutions (and we do not actually do NAS). Instead, our objective is to bring new theoretical insights into *why* different architectures result in similar accuracy.

The rest of the paper is organized as follows: Section 2 discusses the related work and some preliminaries. Then, Section 3 describes our proposed metrics and their theoretical analysis. Section 4 presents detailed experimental results. Finally, Section 5 summarizes our work and contributions.

2 Background and Related Work

NAS techniques [Liu et al. (2018); Zoph et al. (2018); Tan et al. (2019); Cai et al. (2018)] have resulted in state-of-the-art neural architectures. More recently, [Xie et al. (2019); Wortsman et al. (2019)] utilized standard network science ideas such as Barabasi-Albert (BA) [Barabási & Albert (1999)] or Watts-Strogatz (WS) [Watts & Strogatz (1998)] models for NAS. However, like the rest of the NAS research, [Xie et al. (2019); Wortsman et al. (2019)] do *not* address what characteristics of the topology make various models (with different #parameters/FLOPS/layers) achieve similar accuracy. In other words, unlike our work, existing NAS approaches do *not* connect the topology with the gradient flow.

On the other hand, the impact of initialization on model convergence and gradients has been studied [LeCun et al. (2012); Glorot & Bengio (2010); Saxe et al. (2013); Poole et al. (2016); Tarnowski et al. (2018); Pennington et al. (2017)]. Moreover, recent model compression literature attempts to connect pruning at initialization to gradient properties [Lee et al. (2020)]. Again, none of these studies address the impact of the architecture *topology* on gradient propagation. Hence, our work is orthogonal to prior art that explores the impact of initialization on gradients. Related work on important network science and gradient propagation concepts is discussed below.

Preliminaries. In our work, we use the following two well-established concepts:

Definition 1 (Average Degree Newman et al. (2011))**.** *Average degree ($\hat{k}$) of a network determines the average number of connections for all nodes, $\hat{k}$ = #edges/#nodes.*

Average degree and degree distribution (*i.e.*, distribution of nodes' degrees) are important topological characteristics which directly affect how information flows through a network. How fast a signal can propagate through a network heavily depends on the network topology.

Definition 2 (Layerwise Dynamical Isometry (LDI) Lee et al. (2020))**.** *A deep network satisfies LDI if the singular values of Jacobians at initialization are close to 1 for all layers. Specifically, for a multilayer feed-forward network, let $\boldsymbol{s_i}$ ($\boldsymbol{W_i}$) be the output (weights) of layer i such that $\boldsymbol{s_i} = \phi(\boldsymbol{h_i}), \boldsymbol{h_i} = \boldsymbol{W_i}\boldsymbol{s_{i-1}} + \boldsymbol{b_i}$; then, the Jacobian matrix at layer i is defined as: $\boldsymbol{J_{i,i-1}} = \frac{\partial \boldsymbol{s_i}}{\partial \boldsymbol{s_{i-1}}} = \boldsymbol{D_i}\boldsymbol{W_i}$. Here, $\boldsymbol{J_{i,i-1}} \in \mathbb{R}^{w_i, w_{i-1}}$, w_i is the number of neurons in layer i. $\boldsymbol{D_i^{jk}} = \phi'(\boldsymbol{h_i})\delta_{jk}$. ϕ' denotes the derivative of non-linearity ϕ and δ_{jk} is Kronecker delta. Then, if the singular values σ_j for all $\boldsymbol{J_{i,i-1}}$ are close to 1, then the network satisfies the LDI.*

LDI discourages the signal propagating through the DNN from getting attenuated or amplified too much; this ensures faithful propagation of gradients [Saxe et al. (2013)].

3 Topological Properties of Neural Architectures

We first model DNNs via network science to derive our proposed topological metrics. We then demonstrate the theoretical relationship between the topological metrics and gradient propagation.

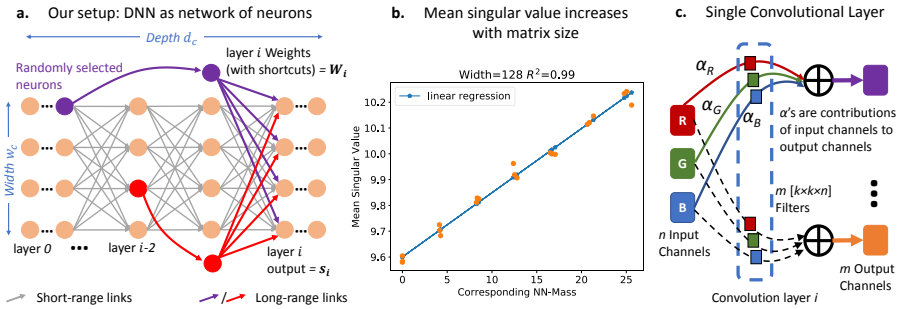


Figure 1: (a) DNN setup: The DNN (depth d_c, width w_c) has layer-by-layer short-range connections (gray) with additional long-range links (purple/red). (b) Simulation of Gaussian matrices: Mean singular values vs. size of a matrix $(w_c + m/2, w_c)$. Mean singular values increase as m increases (more simulations are given in Appendix D). (c) Convolutional layers form a similar topological structure as MLP layers: All input channels contribute to all output channels.

3.1 Modeling DNNs via Network Science

We start with a generic MLP setup with d_c layers containing w_c neurons each and assume shortcut connections (or *long-range links*) superimposed on top of a typical MLP structure (see Fig. 1(a)). Specifically, all neurons at layer i receive long-range links from a maximum of t_c neurons from previous layers. That is, we *randomly* select $\min\{w_c(i-1), t_c\}$ neurons from layers $0, 1, \ldots, (i-2)$, and concatenate them at layer $i-1$ (see Fig. 1(a))[1]; the concatenated neurons then pass through a fully-connected layer to generate the output of layer i ($\boldsymbol{s_i}$). As a result, the weight matrix $\boldsymbol{W_i}$ (which is used to generate $\boldsymbol{s_i}$) gets additional weights to account for the incoming long-range links. Similar to recent NAS research [Li & Talwalkar (2019)], we select random links because random architectures are often as competitive as the carefully designed models. Moreover, the random long-range links on top of fixed short-range links make our architectures a small-world network (Fig. 6, Appendix A) [Watts & Strogatz (1998)], and allows us to use network science to study their topology. Like standard CNNs (Resnets/Densenets), we can generalize this setup to contain multiple (N_c) cells of width w_c, depth d_c. Long-range links are present only within a cell and not across cells.

What about Resnets? The objective of our paper is to demonstrate that topology plays a fundamental role in gradient propagation; understanding these properties results in predictable behavior of deep networks. Therefore, we **purposely** chose to model topological properties on Densenet-like networks since they impose explicit topological (structural) constraints on gradient flow (due to *concat*); this allows us to exploit network science to study their gradient propagation properties. Clearly, Resnets have a different topological structure than Densenets (they are much more regular) and, therefore, will have different topological properties. In fact, [Veit et al. (2016)] showed that Resnets also exhibit small-world properties like short gradient paths due to the regular skip connections. However, [Veit et al. (2016)] is purely empirical and does not produce any theoretical insights. Hence, while we presently focus on Densenet-type networks, our future work will focus on the Resnet topology.

3.2 Proposed Metrics

Our key objectives are twofold: (*i*) Quantify what topological characteristics of DNN architectures affect their accuracy and gradient flow, and (*ii*) Exploit such properties to *directly* design efficient CNNs. To this end, we propose new metrics called *NN-Density* and *NN-Mass*, as defined below.

Definition 3 (Cell-Density)**.** *Density of a cell quantifies how densely its neurons are connected via long-range links. Formally, for a cell c, cell-density ρ_c is given by:*

$$\rho_c = \frac{\textit{Actual \#long-range links within cell } c}{\textit{Total possible \#long-range links within cell } c} = \frac{2\sum_{i=2}^{d_c-1} min\{w_c(i-1), t_c\}}{w_c(d_c-1)(d_c-2)} \tag{1}$$

For complete derivation, please refer to Appendix B. With the above definition for cell-density, NN-Density (ρ_{avg}) is simply defined as the average density across *all cells* in a DNN.

[1]Here, $w_c(i-1)$ is the total number of candidate neurons from layers $0, 1, \ldots, (i-2)$ that can supply long-range links; if the *maximum* number of neurons t_c that can supply long-range links to the current layer exceeds total number of possible candidates, then all neurons from layers $0, 1, \ldots, (i-2)$ are selected. Neurons are concatenated similar to how channels are concatenated in Densenets [Huang et al. (2017)].

Definition 4 (Mass of DNNs)**.** *NN-Mass quantifies how effectively information can flow through a given DNN topology. For a given width (w_c), models with similar NN-Mass, but different depths (d_c) and #parameters, should exhibit a similar gradient flow and, thus, achieve a similar accuracy.*

Mathematically, density is basically *mass/volume*. Let *volume* be the total number of neurons in a cell. Then, we can derive the NN-Mass (m) by multiplying the cell-density with total neurons in each cell:

$$m = \sum_{c=1}^{N_c} w_c d_c \rho_c = \sum_{c=1}^{N_c} \frac{2d_c \sum_{i=2}^{d_c-1} \min\{w_c(i-1), t_c\}}{(d_c-1)(d_c-2)} \tag{2}$$

Note that, NN-Mass is a function of network width, depth, and long-range links (*i.e.*, the topology of a network). For a fixed number of cells, an architecture can be completely specified by {*depth, width, maximum long-range link candidates*} per cell = $\{d_c, w_c, t_c\}$. Hence, to create different architectures, we vary $\{d_c, w_c, t_c\}$ to create architectures with random #parameters/FLOPS/layers, and NN-Mass. We then train these architectures and characterize their accuracy, topology, and gradient propagation to understand the relationships among them. But first, we provide our theoretical analysis.

3.3 Relationships among topology, NN-Mass and gradient propagation

Without loss of generality, we assume that the DNN has only one cell of width w_c and depth d_c.

Proposition 1 (**NN-Mass and average degree**)**.** *The average degree of a deep network with NN-Mass m is given by $\hat{k} = w_c + m/2$.*

The proof of the above result is given in Appendix C.

Intuition. Proposition 1 states that the average degree of a deep network is $w_c + m/2$, which, given the NN-Mass m, is independent of depth d_c. The average degree indicates how well-connected the network is. Hence, it controls how effectively the information can flow through a given topology. Therefore, for a given width and NN-Mass, the average amount of information that can flow through various architectures (with different #parameters/layers) should be similar (due to the same average degree). Thus, we hypothesize that these topological characteristics might constrain the amount of information being learned by DNNs. Next, we show the impact of topology on gradient propagation.

Proposition 2 (**NN-Mass and LDI**)**.** *Given a small deep network f_S (depth d_S) and a large deep network f_L (depth d_L, $d_L >> d_S$), both with same NN-Mass m and width w_c, the LDI for both models is equivalent. Specifically, if $\mathbf{\Sigma_S^i}$ ($\mathbf{\Sigma_L^i}$) denotes the singular values of the initial layerwise Jacobians $\boldsymbol{J_{i,i-1}}$ for the small (large) model, then, the mean singular values in both models are similar; that is, $\mathbb{E}[\mathbf{\Sigma_S^i}] \approx \mathbb{E}[\mathbf{\Sigma_L^i}]$.*

Proof. To prove this, it suffices to show that the initial Jacobians $\boldsymbol{J_{i,i-1}}$ have similar properties for both models (and thus their singular values will be similar). For our setup, the output of layer i, $\boldsymbol{s_i} = \phi(\boldsymbol{W_i x_{i-1}} + \boldsymbol{b_i})$, where $\boldsymbol{x_{i-1}} = \boldsymbol{s_{i-1}} \cup \boldsymbol{y_{0:i-2}}$ concatenates output of layer $i-1$ ($\boldsymbol{s_{i-1}}$) with the neurons $\boldsymbol{y_{0:i-2}}$ supplying the long-range links (random $\min\{w_c(i-1), t_c\}$ neurons selected uniformly from layers 0 to $i-2$). Hence, $\boldsymbol{J_{i,i-1}} = \partial \boldsymbol{s_i} / \partial \boldsymbol{x_{i-1}} = \boldsymbol{D_i W_i}$. Compared to a typical MLP (see Definition 2), the sizes of $\boldsymbol{D_i}$ and $\boldsymbol{W_i}$ increase to account for incoming long-range links.

For two models f_S and f_L, the layerwise Jacobian ($\boldsymbol{J_{i,i-1}}$) can have two kinds of properties: (*i*) The values inside Jacobian matrix for f_S and f_L can be different, and/or (*ii*) The sizes of layerwise Jacobian matrices for f_S and f_L can be different. Hence, our objective is to show that when the width and NN-Mass are similar, irrespective of the depth of the model (and thus irrespective of number of parameters/FLOPS), both the values and the size of initial layerwise Jacobians are similar.

Let us start by considering a linear network: in this case, $\boldsymbol{J_{i,i-1}} = \boldsymbol{W_i}$. Since the LDI looks at the properties of layerwise Jacobians *at initialization*, and because all models are initialized the same way (*e.g.*, Gaussians with variance scaling[2]), the values inside $\boldsymbol{J_{i,i-1}}$ for both f_S and f_L have same distribution (point (*i*) above is satisfied). We next show that even the sizes of layerwise Jacobians for both models are similar if the width and NN-Mass are similar.

How is topology related to the layerwise Jacobians? Since the average degree is same for both models (see Proposition 1), on average, the number of incoming shortcuts at a typical layer is $w_c \times m/2$.

[2]Variance scaling methods also take into account the number of input/output units. Hence, if the width is the same between models of different depths, the distribution at initialization is still similar.

In other words, since the degree distribution for the random long-range links is Poisson [Barabasi (2016)] with average degree $k_{\mathcal{R}|\mathcal{G}} \approx m/2$ (see Eq. 7, Appendix C), an average $m/2$ neurons supply long-range links to each layer[3]. Therefore, the Jacobians will theoretically have the same dimensions $(w_c + m/2, w_c)$ irrespective of the depth of the neural network (*i.e.*, point (*ii*) is also satisfied).

So far, the discussion has considered only a linear network. For a non-linear network, the Jacobian is given as $\boldsymbol{J_{i,i-1}} = \boldsymbol{D_i W_i}$. As explained in [Lee et al. (2020)], $\boldsymbol{D_i}$ depends on pre-activations $\boldsymbol{h_i} = \boldsymbol{W_i x_{i-1}} + \boldsymbol{b_i}$. As established in several deep network mean field theory studies [Poole et al. (2016); Tarnowski et al. (2018)], the distribution of pre-activations at layer i ($\boldsymbol{h_i}$) is a Gaussian $\mathcal{N}(0, q_i)$ due to the central limit theorem. Similar to [Lee et al. (2020); Pennington et al. (2017)], if the input $\boldsymbol{h_0}$ is chosen to satisfy a fixed point $q_i = q^*$, the distribution of D_i becomes independent of the depth ($\mathcal{N}(0, q^*)$). Therefore, the distribution of both $\boldsymbol{D_i}$ and $\boldsymbol{W_i}$ is similar for different models irrespective of the depth, even for non-linear networks. Moreover, the sizes of the matrices will again be similar due to similar average degree in f_S and f_L.

Hence, the size and distribution of values in the Jacobian matrix are similar for both the large and the small model (provided the width and NN-Mass are similar); that is, the distribution and mean singular values will also be similar: $\mathbb{E}[\boldsymbol{\Sigma_S^i}] \approx \mathbb{E}[\boldsymbol{\Sigma_L^i}]$. In other words, LDI is equivalent between models of different depths if their width and NN-Mass are similar. □

We note that the mean singular values increase with NN-Mass. To illustrate this effect, we numerically simulate several Gaussian-distributed matrices of sizes $(w_c + m/2, w_c)$ and compute their mean singular values. Specifically, we vary m for widths w_c and see the impact of this size variation on mean singular values. Fig. 1(b) shows that as NN-Mass varies, the mean singular values linearly increase with NN-Mass. In our experiments, we show that this linear trend between mean singular values and NN-Mass holds true for actual non-linear deep networks. A formal proof of this observation and more simulations are given in Appendix D. Note that, our results should *not* be interpreted as bigger models yield larger mean singular values. We show in the next section that the relationship between the #parameters and mean singular values is significantly worse than that for NN-Mass. Hence, it is the topological properties that enable LDI in different deep networks and not #parameters.

Remark 1 (NN-Mass formulation is same for CNNs). Fig. 1(c) shows a typical convolutional layer. Since all channel-wise convolutions are added together, each output channel is some function of all input channels. This makes the topology of CNNs similar to that of our MLP setup. The key difference is that the nodes in the network (see Fig. 1(a)) are now channels and not individual neurons. Of note, for our CNN setup, we use three cells (similar to Densenets). More details on CNN setup (including a concrete example for NN-Mass calculations) are given in Appendices E and F.

We next provide extensive empirical evidence for our theoretical insights on topology, gradient propagation, LDI, and model accuracy (Proposition 2).

4 Experimental Setup and Results

4.1 Experimental Setup

For experiments on MLPs and CNNs, we generate random architectures with different NN-Mass and number of parameters (#Params) by varying $\{d_c, w_c, t_c\}$. For random MLPs with different $\{d_c, t_c\}$ and $w_c = 8$ (#cells = 1), we conduct the following experiments on the MNIST dataset: (*i*) We explore the impact of varying #Params and NN-Mass on the test accuracy; (*ii*) We demonstrate how LDI depends on NN-Mass and #Params; (*iii*) We further show that models with similar NN-Mass (and width) result in similar training convergence, despite having different depths and #Params.

After the extensive empirical evidence for our theoretical insights (*i.e.*, the connection between gradient propagation and topology), we next move on to random CNN architectures with three cells. We conduct the following experiments on the CIFAR-10 and CIFAR-100 datasets: (*i*) We show that NN-Mass can further identify CNNs that achieve similar test accuracy, despite having highly different #Params/FLOPS/layers; (*ii*) We show that NN-Mass is a significantly more effective indicator of model performance than parameter counts; (*iii*) We also show that our findings hold for CIFAR-100 and ImageNet; (*iv*) We further verify our findings for ops like depthwise separable convolutions.

Finally, we exploit NN-Mass to directly design efficient CNNs (for CIFAR-10) which achieve accuracy comparable to significantly larger models. For these experiments, the models are trained

[3]Theoretically, a Poisson process assumes a constant rate of arrival of links.

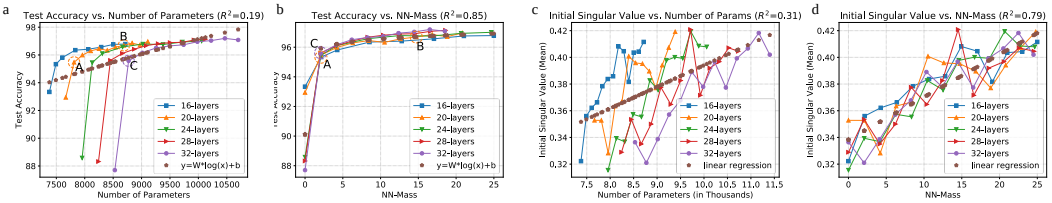


Figure 2: MNIST results: (a) Models with different number of parameters (#Params) achieve similar test accuracy. (b) Test accuracy curves of models with different depths/#Params concentrate when plotted against NN-Mass (test accuracy std. dev. $\sim 0.05 - 0.34\%$). (c,d) Mean singular values of $\boldsymbol{J_{i,i-1}}$ are much better correlated with NN-Mass ($R^2 = 0.79$) than with #Params ($R^2 = 0.31$).

for 600 epochs. Overall, we train hundreds of different MLP and CNN architectures with each MLP (CNN) repeated five (three) times with different random seeds, to obtain our results. More setup details (*e.g.*, architecture details, learning rates, *etc.*) are given in Appendix G (see Tables 1, 2, and 3).

4.2 MLP Results (MNIST/Synthetic Data): Topology vs. Gradient Propagation

Test Accuracy. Fig. 2(a) shows test accuracy *vs.* #Params of DNNs with different depths on the MNIST dataset. As evident, even though many models have different #Params, they achieve a similar test accuracy. On the other hand, when the same set of models are plotted against NN-Mass, their test accuracy curves cluster together tightly, as shown in Fig. 2(b). To further quantify the above observation, we generate a linear fit between test accuracy *vs.* log(#Params) and log(NN-Mass) (see brown markers in Fig. 2(a,b)). For NN-Mass, we achieve a significantly higher goodness-of-fit $R^2 = 0.85$ than that for #Params ($R^2 = 0.19$). This demonstrates that NN-Mass can identify DNNs that achieve similar accuracy, even if they have a highly different number of parameters/FLOPS[4]/layers. We next investigate the gradient propagation properties to explain the test accuracy results.

Layerwise Dynamical Isometry (LDI). We calculate the mean singular values of initial layerwise Jacobians, and plot them against #Params (see Fig. 2(c)) and NN-Mass (see Fig. 2(d)). Clearly, NN-Mass ($R^2 = 0.79$) is far better correlated with the mean singular values than #Params ($R^2 = 0.31$). More importantly, just as Proposition 2 predicts, these results show that models with similar NN-Mass and width have equivalent LDI properties, irrespective of the total depth (and, thus #Params) of the network. For example, even though the 32-layer models have more parameters, they have similar mean singular values as the 16-layer DNNs. This clearly suggests that the gradient propagation properties are heavily influenced by the topological characteristics like NN-Mass, and not just by DNN depth and #Params. Of note, the linear trend in Fig. 2(d) is similar to that seen in Fig. 1(b).

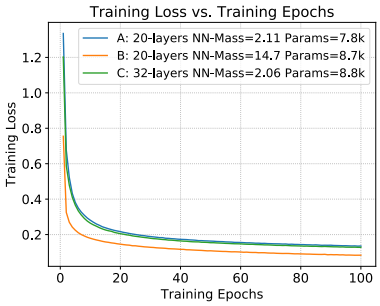


Figure 3: Models A and C have the same NN-Mass and achieve very similar training convergence, even though they have highly different #Params and depth. Model B has significantly fewer layers than C but the same #Params, yet achieves a faster training convergence than C (B has higher NN-Mass than C).

Training Convergence. The above results suggest the following hypotheses: (*i*) If the gradient flow between DNNs (with similar NN-Mass and width) is similar, their training convergence should be similar, even if they have highly different #Params and depths; (*ii*) If two models have same #Params (and width), but different depths and NN-Mass, then the DNN with higher NN-Mass should have faster training convergence (since its mean singular value will be higher – see the trend in Fig. 2(d)).

To demonstrate that both hypotheses above hold true, we pick three models – A, B, and C – from Fig. 2(a,b) and plot their training loss *vs.* #epochs. Models A and C have similar NN-Mass, but C has more #Params and depth than A. Model B has far fewer layers and nearly the same #Params as C, but has a higher NN-Mass. Fig. 3 shows the training convergence results for all three models. As evident, the training convergence of model A (7.8K Params, 20-layers) nearly coincides with that of model C (8.8K Params, 32-layers). Moreover, even though model B (8.7K Params, 20-layers) is

[4]For our setup, more parameters lead to more FLOPS. FLOPS results are given for CNNs in Appendix H.8.

shallower than the 32-layer model C, the training convergence of B is significantly faster than that of C (due to higher NN-Mass and, therefore, better LDI). Training convergence results for several other models in Fig. 2(a,b) show similar observations (see Fig. 10 in Appendix H.1). These results clearly validate the theoretical insights in Proposition 2, and emphasize the importance of topological properties of neural architectures in characterizing the gradient propagation and model performance. Other similar experiments for synthetic datasets are given in Appendix H.2.

4.3 CNN Results on CIFAR-10, CIFAR-100, and ImageNet Datasets

Since we have now established a concrete relationship between gradient propagation and topological properties, in the rest of the paper, we show that NN-Mass can be used to identify and design efficient CNNs that achieve similar accuracy as models with significantly higher #Params/FLOPS/layers.

Model Performance. Fig. 4(a) shows the test accuracy of various CNNs *vs.* total #Params. As evident, models with highly different number of parameters (*e.g.*, see models A-E in box W), achieve a similar test accuracy. Note that, there is a large gap in the model size: CNNs in box W range from 5M parameters (model A) to 9M parameters (models D,E). Again, as shown in Fig. 4(b), when plotted against NN-Mass, the test accuracy curves of CNNs with different depths cluster together (*e.g.*, models A-E in box W cluster into A'-E' within bucket Z). Hence, NN-Mass identifies CNNs with similar accuracy, despite having highly different #Params/layers. The same holds true for models within X and Y. More results with different width multipliers are given in Appendix H.4. Again, the observations are similar and for higher width, the models tend to cluster even more tightly for NN-Mass.

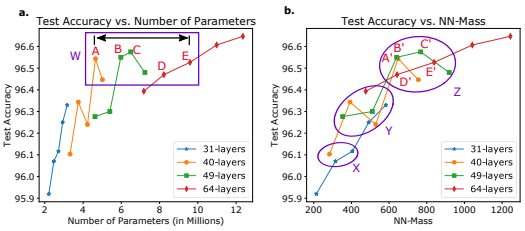


Figure 4: CIFAR-10 Width Multiplier $wm = 2$: (a) Models with very different #Params (box W) achieve similar test accuracies. (b) Models with similar accuracy often have similar NN-Mass: Models in W cluster into Z. Results are reported as the mean of three runs (std. dev. $\sim 0.1\%$).

NN-Mass vs. Parameter Counting. As shown in Fig. 17 in Appendix H.6, for $wm = 2$, #Params yield an $R^2 = 0.76$ which is lower than that for NN-Mass ($R^2 = 0.84$, see Fig. 17(a, b)). However, for higher widths ($wm = 3$), the parameter count completely fails to predict model performance ($R^2 = 0.14$ in Fig. 17(c)). On the other hand, NN-Mass achieves a significantly higher $R^2 = 0.90$ (see Fig. 17(d)).

Results for CIFAR-100 Dataset. We now corroborate our main findings on CIFAR-100 dataset which is significantly more complex than CIFAR-10. To this end, we train the models in Fig. 4 on CIFAR-100. Fig. 18 (see Appendix H.7) once again shows that several models with highly different number of parameters achieve similar accuracy. Moreover, Fig. 18(b) demonstrates that these models get clustered when plotted against NN-Mass. Further, a high $R^2 = 0.84$ is achieved for a linear fit on the accuracy vs. log(NN-Mass) plot (see Appendix H.7 and Fig. 18).

Results for #FLOPS. So far, we have shown results for the number of parameters. However, the results for #FLOPS follow a very similar pattern (see Fig. 19 in Appendix H.8). In summary, we show that NN-Mass can identify models that yield similar test accuracy, despite having very different #parameters/FLOPS/layers. We next use this observation to directly design efficient architectures.

NN-Mass scales to ImageNet. For ImageNet, we create several CNNs containing four cells and total depth $\in \{48, 56, 60, 64, 68\}$ layers, and width multiplier $wm \in \{1.5, 2\}$. Due to lack of resources, we minimally trained these models on ImageNet dataset for 60 epochs. Fig. 5(a) shows the test accuracy of these CNNs vs. total #Params, while Fig. 5(b) shows the test accuracy vs. NN-Mass. As evident, although the model sizes are very different (e.g., model X is 3M parameters bigger than model W; see other arrows also), the accuracy is quite similar. Once again, the models cluster together when plotted against NN-Mass (e.g., see clusters for models {W,X}, {Y,Z}, and {P,Q} in Fig. 5(b)). Note that, the accuracies do *not* saturate (similar to other CIFAR-10 and CIFAR-100 results in Fig. 4, 15, 18 and Fig. 5(c,d) in next section): Cluster {Y,Z} achieves 4% lower Top-1 accuracy (red points in Fig. 5(a,b)) than cluster {W,X}, whereas within each cluster, the models are merely 0.2% and 0.7% away from each other. Same observation holds for Top-5 accuracy (blue points in Fig. 5(a,b)). Finally, models {P,Q} cannot be compared in accuracy against {Y,Z} since they have different width (recall that Proposition 2 requires the models within the *same* cluster to

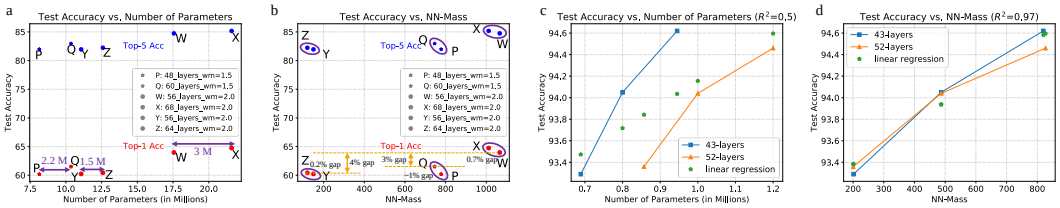


Figure 5: (a,b) ImageNet: (a) Models {P,Q}, {W,X}, and {Y,Z} have very different #Params but similar test accuracy. (b) When plotted against NN-Mass, the models with similar NN-Mass and accuracy cluster together. (c,d) CIFAR-10 with DSConv: Again, models with similar NN-Mass achieve similar accuracy but have quite different #Params/layers.

have both same width and NN-Mass). We have provided the $wm = 1.5$ points to show that NN-Mass works for ImageNet across multiple widths. Hence, our ideas scale to ImageNet dataset.

Generalizability to other operations. DARTS is heavily influenced by depthwise separable convolutions (DSConv). To demonstrate that NN-Mass works with DSConv, we take our current setup (layer-by-layer convolutions with channels connected via random long-range links) and replace all convolutions with MobilenetV2 Expansion Blocks (1x1 conv $\rightarrow$ 3x3 DSConv $\rightarrow$ 1x1 conv) [Sandler & et al. (2018)]. Random long-range links connect input channels across various Expansion Blocks. Fig. 5(c) shows test accuracy vs. #Params of CNNs with DSConv on CIFAR-10. Again, even though many models have different #Params, they achieve a similar test accuracy. On the other hand, when the same set of models are plotted against NN-Mass, their test accuracy curves cluster together tightly, as shown in Fig. 5(d), with a significantly higher goodness-of-fit ($R^2 = 0.97$) than that for #Params ($R^2 = 0.5$). This demonstrates that NN-Mass can be used to quantify topological properties of diverse/heterogeneous CNNs with regular convolutions, DSConv, pointwise conv, etc.

Extension to existing models. Our work exploits SotA Densenet-like models which are more recent than Resnets. For existing Densenets [Huang et al. (2017)], cell-density (Eq. 1) = 1 (all-to-all connections); thus, NN-Mass for Densenet = $\sum_{all\ cells}$[(#channels per layer for this cell) $\times$ (#layers per cell)]. For VGG-like models, there are no shortcuts, so NN-Mass = 0. Our theory works for NN-Mass = 0: Fig. 2(b) shows two clusters for [low-depth (16,20) NN-Mass 0] models and [high-depth (24,28,32) NN-Mass 0] models. This is *not* surprising: without shortcuts, the gradient diminishes as depth increases (e.g., see Resnets [He et al. (2016)]). Same holds for our NN-Mass=0 CNNs on ImageNet and CIFAR-10. Hence, our ideas apply to a variety of existing models. The Resnet skip connection topology is not the same as that for Densenets and, hence, this is left for future work.

Directly designing efficient CNNs with NN-Mass. In Appendix H.9, we design regular CNNs (no DSConv) with long-range links for CIFAR-10 dataset and show how such models can be compressed directly using NN-Mass equation (2) without searching for an efficient network. Overall, as shown in Table 5 (Appendix H.9), our models reach a test accuracy of 96.82%-97.00%, while reducing the number of parameters and FLOPS by up to $3\times$ over large CNNs (e.g., 3.82M *vs.* 11.89M parameters). Moreover, DARTS [Liu et al. (2018)], a competitive NAS baseline, achieves a comparable (97%) accuracy with slightly lower 3.3M parameters. As mentioned earlier, our objective is *not* to beat DARTS or any other NAS, but rather to provide theoretical insights into the behavior of neural architectures. Our efficient, high-accuracy, theoretically grounded CNNs (that do not use specialized search spaces like NAS) clearly demonstrate that we bridge this gap between theory and practice.

5 Conclusion

We have proposed a new, network science-based metric called *NN-Mass* which quantifies how effectively information flows through a given architecture. We have also established concrete theoretical relationships among NN-Mass, topological structure of networks, and layerwise dynamical isometry that ensures faithful propagation of gradients through DNNs. Our experiments have demonstrated that NN-Mass is significantly more effective than the number of parameters to characterize the gradient flow properties, and to identify models with similar accuracy, despite having a highly different number of parameters/FLOPS/layers. We have performed experiments on both synthetic and real datasets (*e.g.*, MNIST, CIFAR-10/100, and ImageNet). Finally, to show the practical implications of our work, we have exploited the closed-form equation of our NN-Mass metric to directly *design* efficient CNNs.

Since topology is deeply intertwined with the gradient propagation, such topological metrics deserve major attention in the future for other architectures like Resnets/Mobilenets. Research at the intersection of initialization and topology will also be important in future work.

REFERENCES

Albert-Laszlo Barabasi. *Network Science (Chapter 3: Random Networks)*. Cambridge University Press, 2016. URL https://bit.ly/2ONAUqQ.

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286 (5439):509–512, 1999.

Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv:1704.04861*, 2017.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 2012.

Namhoon Lee, Thalaiyasingam Ajanthan, Stephen Gould, and Philip H. S. Torr. A signal propagation perspective for pruning neural networks at initialization. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJeTo2VFwH.

Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. *arXiv preprint arXiv:1902.07638*, 2019.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

Remi Monasson. Diffusion, localization and dispersion relations on "small-world" lattices. *The European Physical Journal B-Condensed Matter and Complex Systems*, 12(4):555–567, 1999.

Mark Newman, Albert-Laszlo Barabasi, and Duncan J Watts. *The structure and dynamics of networks*, volume 19. Princeton University Press, 2011.

Mark EJ Newman and Duncan J Watts. Renormalization group analysis of the small-world network model. *Physics Letters A*, 263(4-6):341–346, 1999.

Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In *Advances in neural information processing systems*, pp. 4785–4795, 2017.

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Advances in neural information processing systems*, pp. 3360–3368, 2016.

Mark Sandler and et al. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *arXiv:1801.04381*, 2018.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2820–2828, 2019.

Wojciech Tarnowski, Piotr Warchoł, Stanisław Jastrzębski, Jacek Tabor, and Maciej A Nowak. Dynamical isometry is achieved in residual networks in a universal way for any activation function. *arXiv preprint arXiv:1809.08848*, 2018.

Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Advances in neural information processing systems*, pp. 550–558, 2016.

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

Mitchell Wortsman, Ali Farhadi, and Mohammad Rastegari. Discovering neural wirings. *arXiv preprint arXiv:1906.00586*, 2019.

Saining Xie, Alexander Kirillov, Ross Girshick, and Kaiming He. Exploring randomly wired neural networks for image recognition. *arXiv preprint arXiv:1904.01569*, 2019.

Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.

A DNNs/CNNs with long-range links are Small-World Networks

Note that, the DNNs/CNNs considered in our work have both short-range and long-range links (see Fig. 1(a)). This kind of topology typically falls into the category of small-world networks which can be represented as a lattice network $\mathcal{G}$ (containing short-range links) superimposed with a random network $\mathcal{R}$ (to account for long-range links) [Monasson (1999); Newman & Watts (1999)]. This is illustrated in Fig. 6.

B Derivation of Density of a Cell

Note that, the maximum number of neurons contributing long-range links at each layer in cell c is given by t_c. Also, for a layer i, possible candidates for long-range links = all neurons up to layer $(i-2)$ are $w_c(i-1)$ (see Fig. 1(a)). Indeed, if t_c is sufficiently large, initial few layers may not have t_c neurons that can supply long-range links. For these layers, we use all available neurons for long-range links. Therefore, for a given layer i, number of long-range links (l_i) is given by:

$$l_i = \begin{cases} w_c(i-1) \times w_c & \text{if } t_c > w_c(i-1) \\ t_c \times w_c & \text{otherwise} \end{cases} \tag{3}$$

where, both cases have been multiplied by w_c because once the neurons are randomly selected, they supply long-range links to all w_c neurons at the current layer i (see Fig. 1(a)). Hence, for an entire cell, total number of neurons contributing long-range links (l_c) is as follows:

$$l_c = w_c \sum_{i=2}^{d_c-1} \min\{w_c(i-1), t_c\} \tag{4}$$

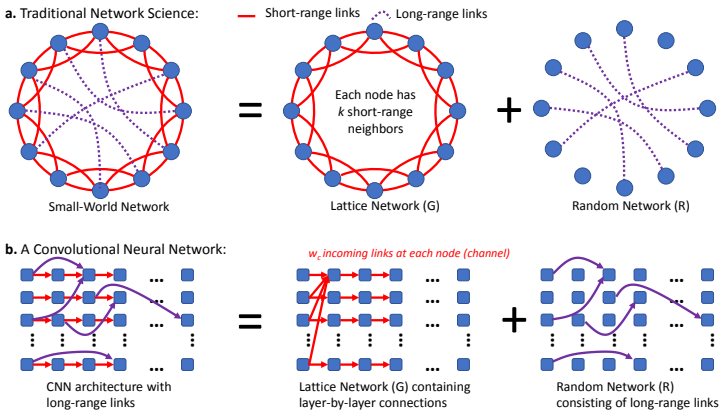


Figure 6: (a) Small-World Networks in traditional network science are modeled as a superposition of a lattice network ($\mathcal{G}$) and a random network $\mathcal{R}$ [Watts & Strogatz (1998); Newman & Watts (1999); Monasson (1999)]. (b) A DNN/CNN with both short-range and long-range links can be similarly modeled as a random network superimposed on a lattice network. Not all links are shown for simplicity.

On the other hand, the total number of possible long-range links within a cell (L) is simply the sum of possible candidates at each layer:

$$\begin{aligned} L &= \sum_{i=2}^{d_c-1} w_c(i-1) \times w_c = w_c^2 \sum_{i=2}^{d_c-1} (i-1) \\ &= w_c^2[1+2+\ldots+(d_c-2)] \\ &= \frac{w_c^2(d_c-1)(d_c-2)}{2} \end{aligned} \tag{5}$$

Using Eq. 4 and Eq. 5, we can rewrite Eq. 1 as:

$$\rho_c = \frac{2\sum_{i=2}^{d_c-1} \min\{w_c(i-1), t_c\}}{w_c(d_c-1)(d_c-2)} \tag{6}$$

□

C Proof of Proposition 1

Proposition 1 (**NN-Mass and average degree of the network (a topological property)**). *The average degree of a deep network with NN-Mass m is given by $\hat{k} = w_c + m/2$.*

Proof. As shown in Fig. 6, deep networks with shortcut connections can be represented as small-world networks consisting of two parts: (*i*) lattice network containing only the short-range links, and (*ii*) random network superimposed on top of the lattice network to account for long-range links. For sufficiently deep networks, the average degree for the lattice network will be just the width w_c of the network. The average degree of the randomly added long-range links $\bar{k}_{\mathcal{R}|\mathcal{G}}$ is given by:

$$\begin{aligned} \bar{k}_{\mathcal{R}|\mathcal{G}} &= \frac{\textit{Number of long-range links added by } \mathcal{R}}{\textit{Number of nodes}} = \frac{w_c \sum_{i=2}^{d_c-1} \min\{w_c(i-1), t_c\}}{w_c d_c} \\ &= \frac{m(d_c-1)(d_c-2)}{2d_c^2} \quad \text{(using 2 for one cell)} \\ &\approx \frac{m}{2} \quad \text{(when } d_c >> 2\text{, } \textit{e.g.}\text{, for deep networks)} \end{aligned} \tag{7}$$

Therefore, average degree of the complete model is given by $w_c + m/2$. □

D PROOF OF PROPOSITION 2

Proposition 2 (NN-Mass and LDI). *Given a small deep network f_S (depth d_S) and a large deep network f_L (depth d_L, $d_L >> d_S$), both with same NN-Mass m and width w_c, the LDI for both models is equivalent. Specifically, if $\mathbf{\Sigma_S^i}$ ($\mathbf{\Sigma_L^i}$) denotes the singular values of the initial layerwise Jacobians $\mathbf{J_{i,i-1}}$ for the small (large) model, then, the mean singular values in both models are similar; that is, $\mathbb{E}[\mathbf{\Sigma_S^i}] \approx \mathbb{E}[\mathbf{\Sigma_L^i}]$.*

Proof. Consider a matrix $M \in \mathbb{R}^{H \times W}$ with H rows and W columns, and all entries independently initialized with a Gaussian Distribution $\mathcal{N}(0, q)$, we calculate its mean singular value. We first perform Singular Value Decomposition (*SVD*) on the given matrix M:

$$U \in \mathbb{R}^{H \times H}, \Sigma \in \mathbb{R}^{H \times W}, V \in \mathbb{R}^{W \times W} = SVD(M)$$
$$\Sigma \in \mathbb{R}^{\mathrm{H} \times \mathrm{W}} = Diag(\sigma_0, \sigma_1, ..., \sigma_K)$$

Given a row vector $\vec{u_i} \in \mathbb{R}^H$ in U, and a row vector $\vec{v_i} \in \mathbb{R}^W$ in V, we use the following relations of SVD in our proof:

$$\sigma_i = \vec{u_i}^T M \vec{v_i}$$
$$\vec{u_i}^T \vec{u_i} = 1$$
$$\vec{v_i}^T \vec{v_i} = 1$$

It is hard to directly compute the mean singular value $\mathbb{E}[\sigma_i]$. To simplify the problem, consider σ_i^2:

$$\begin{aligned} \sigma_i^2 &= \sigma_i \times \sigma_i^T \\ &= (\vec{u_i}^T M \vec{v_i})(\vec{u_i}^T M \vec{v_i})^T \\ &= \vec{u_i}^T M \vec{v_i} \vec{v_i}^T M^T \vec{u_i} \\ &= \vec{u_i}^T M M^T \vec{u_i} \end{aligned} \tag{8}$$

Substituting $B = MM^T$ (where, $B \in \mathbb{R}^{H \times H}$), and using m_{ij} to represent the ij^{th} entry of the given matrix M, the entry b_{ij} in B is given by:

$$b_{ij} = \begin{cases} \sum_{k=1}^{H} m_{ik}^2, \text{ when } i = j \\ \sum_{k=1}^{H} m_{ik} m_{kj}, \text{ when } i \neq j \end{cases}$$

Since m_{ij} follows an independent and identical Gaussian Distribution $\mathcal{N}(0, q)$, the diagonal entries of B (b_{ii}) follow a chi-square distribution with H degrees of freedom:

$$b_{ii} \sim \chi^2(H)$$

For the non-diagonal entries of B, i.e. $i \neq j$, suppose $z_k = xy,\ x = m_{ik}, and\ y = m_{kj}$; then the probability density function (*PDF*) of z is as follows:

$$PDF_Z(z_k) = \int_{-\infty}^{\infty} \frac{PDF_X(t)}{|t|} PDF_Y(\frac{z_k}{t}) dt \quad = \int_{-\infty}^{\infty} \frac{1}{2\pi|t|} e^{-\frac{t^4 + z_k^2}{2t^2}} dt \tag{9}$$

Based on probability density function of z_k, the expectation of z_k is given by:

$$\mathbb{E}[z_k] = \int_{-\infty}^{\infty} PDF_Z(z_k) z_k dz_k$$

As shown in Eq. 9, $PDF_Z(z_k)$ is an even function, then $PDF_Z(z_k)z_k$ is an odd function; therefore, $\mathbb{E}[z_k] = 0$ and, thus, $\mathbb{E}[b_{ij}] = \sum_{k=1}^{H} \mathbb{E}[z_k] = 0,\ when\ i \neq j$.

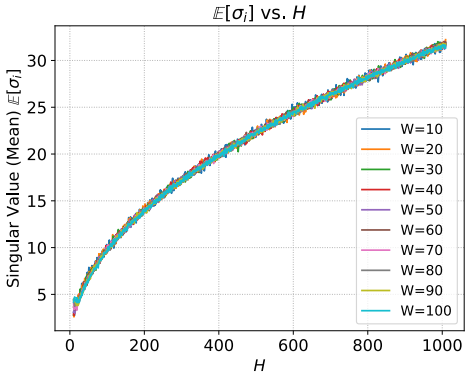


Figure 7: Mean Singular Value $\mathbb{E}[\sigma_i]$ only increases with H while varying W. For small-enough ranges, the $\mathbb{E}[\sigma_i]$ *vs.* H relationship can be approximated by a linear trend.

Hence, we can now get the expectation for each entry in the Matrix B: $\mathbb{E}[b_{ij}] = \begin{cases} H, & i = j \\ 0, & i \neq j \end{cases}$; that is:

$$\mathbb{E}[B] = Diag(b_{ii}) = H\mathbb{I} \tag{10}$$

where, $\mathbb{I} \in \mathbb{R}^{H \times H}$ is an identity matrix. Combining Eq. 8 and Eq. 10, we get the following results:

$$\begin{aligned} \mathbb{E}[\sigma_i^2] &= \mathbb{E}[\vec{u_i}^T M M^T \vec{u_i}] \\ &= \mathbb{E}[\vec{u_i}^T]\mathbb{E}[MM^T]\mathbb{E}[\vec{u_i}] \\ &= \mathbb{E}[\vec{u_i}^T]\mathbb{E}[B]\mathbb{E}[\vec{u_i}] \\ &= \mathbb{E}[\vec{u_i}^T]H\mathbb{I}\mathbb{E}[\vec{u_i}] \\ &= H\mathbb{E}[\vec{u_i}^T\vec{u_i}] \\ &= H \end{aligned} \tag{11}$$

Therefore, we have:

$$\mathbb{E}[\sigma_i^2] = H \tag{12}$$

Eq. 12 states that, for a Gaussian $M \in \mathbb{R}^{H \times W}$, $\mathbb{E}[\sigma_i^2]$ is dependent on number of rows H, and does *not* depend on W. To empirically verify this, we simulate several Gaussian matrices of widths $W \in \{10, 20, ..., 100\}$ and $H \in (0, 1000)$. We plot $\mathbb{E}[\sigma_i]$ *vs.* H in Fig. 7. As evident, for different W, the mean singular values are nearly coinciding, thereby showing that mean singular value indeed depends on H. Also, for small-enough ranges of H, the relationship between $\mathbb{E}[\sigma_i]$ and H can be approximated with a linear trend.

To see the above linear trend between the mean singular values ($\mathbb{E}[\sigma_i]$) and H, we now simulate a more realistic scenario that will happen in the case of initial layerwise Jacobian matrices ($\boldsymbol{J_{i,i-1}}$). As explained in the main paper, the layerwise Jacobians will theoretically have $(w_c + m/2, w_c)$ dimensions, where w_c is the width of DNN and m is the NN-Mass. That is, now $M = \boldsymbol{J_{i,i-1}}$, $W = w_c$, and $H = w_c + m/2$. Hence, in Fig. 8, we plot mean singular values for Gaussian distributed matrices of size $(w_c + m/2, w_c)$ *vs.* NN-Mass (m). As evident, for w_c ranging from 8 to 256, mean singular values increase linearly with NN-Mass. We will explicitly demonstrate in our experiments that this linear trend holds true for actual non-linear deep networks.

Finally, since the Jacobians have a size of $(w_c + m/2, w_c)$, Eq. 12 suggests that its mean singular values should depend on $H = w_c + m/2$. Hence, when two DNNs have same NN-Mass and width, their mean singular values should be similar, *i.e.*, $\mathbb{E}[\boldsymbol{\Sigma_S^i}] \approx \mathbb{E}[\boldsymbol{\Sigma_L^i}]$ (irrespective of their depths). □

E CNN Details

In contrast to our MLP setup which contains only a single cell of width w_c and depth d_c, our CNN setup contains three cells, each containing a fixed number of layers, similar to prior works such as

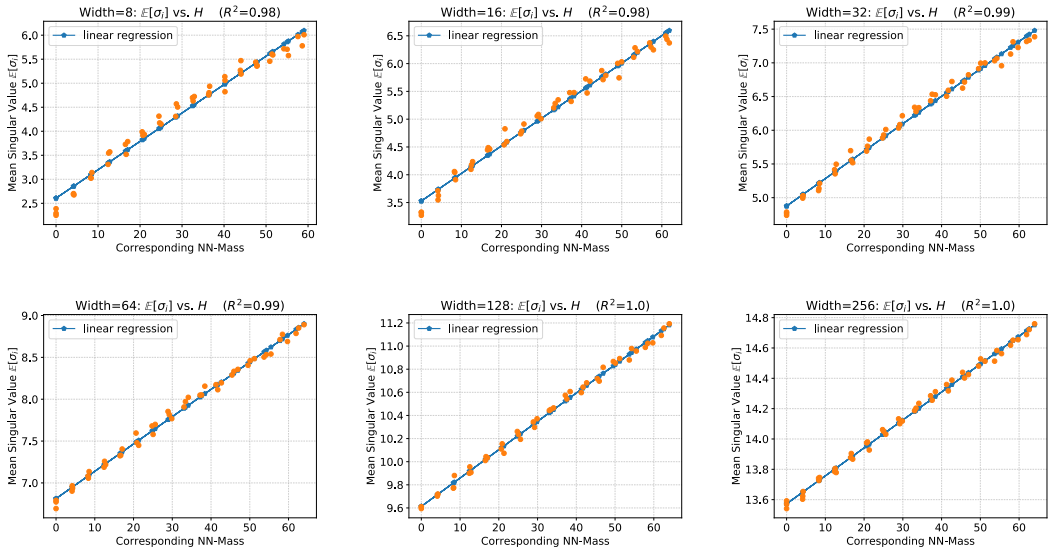


Figure 8: To simulate more realistic Jacobian matrices, we calculate the mean singular value of matrix M with size $[w_c + m/2, w_c]$ (w_c is given by the *Width* in the title of each sub-figure). Clearly, $\mathbb{E}[\sigma_i]$ varies linearly with corresponding NN-Mass for all w_c values. Moreover, as w_c increases, the mean singular values ($\mathbb{E}[\sigma_i]$) increase. Both observations show that $\mathbb{E}[\sigma_i]$ increases with $\hat{k} = w_c + m/2$ (since the height of the Jacobian matrix $H = \hat{k}$ depends on both w_c and m).

Densenets [Huang et al. (2017)], Resnets [He et al. (2016), *etc*]. However, topologically, a CNN is very similar to MLP. Since in a regular convolutional layer, channel-wise convolutions are added to get the final output channel (see Fig. 1(c)), each input channel contributes to each output channel at all layers. This is true for both long-range and short-range links; this makes the topological structure of CNNs similar to our MLP setup shown in Fig. 1(a) in the main paper (the only difference is that now each channel is a node in the network and not each neuron).

In the case of CNNs, following the standard practice [Simonyan & Zisserman (2014)], the width (*i.e.*, the number of channels per layer) is increased by a factor of two at each cell as the feature map height and width are reduced by half. After the convolutions, the final feature map is average-pooled and passed through a fully-connected layer to generate logits. The width (*i.e.*, the number of channels at each layer) of CNNs is controlled using a width multiplier, wm (like in Wide Resnets [Zagoruyko & Komodakis (2016)] and Mobilenets [Howard et al. (2017)]). Base #channels in each cell is [16,32,64]. For $wm = 2$, cells will have [32,64,128] channels per layer.

F Example: Computing NN-Mass for a CNN

Given a CNN architecture shown in Fig. 9, we now calculate its NN-Mass. This CNN consists of three cells, each containing $d_c = 4$ convolutional layers. The three cells have a width, (*i.e.*, the number of channels per layer) of 2, 3, and 4, respectively. We denote the network width as $w_c = [2, 3, 4]$. Finally, the maximum number of channels that can supply long-range links is given by $t_c = [3, 4, 5]$. That is, the first cell can have a maximum of three long-range link *candidates* per layer (*i.e.*, previous channels that can supply long-range links), the second cell can have a maximum of four long-range link candidates per layer, and so on. Moreover, as mentioned before, we randomly choose $\min\{w_c(i-1), t_c\}$ channels for long-range links at each layer. The inset of Fig. 9 shows how long-range links are created by concatenating the feature maps from previous layers.

Hence, using $d_c = 4$, $w_c = [2, 3, 4]$, and $t_c = [3, 4, 5]$ for each cell c, we can directly use Eq. 2 to compute the NN-Mass value. Putting the values in the equations, we obtain $m = 28$. Consequently, the set $\{d_c, w_c, t_c\}$ can be used to specify the architecture of any CNN with concatenation-type

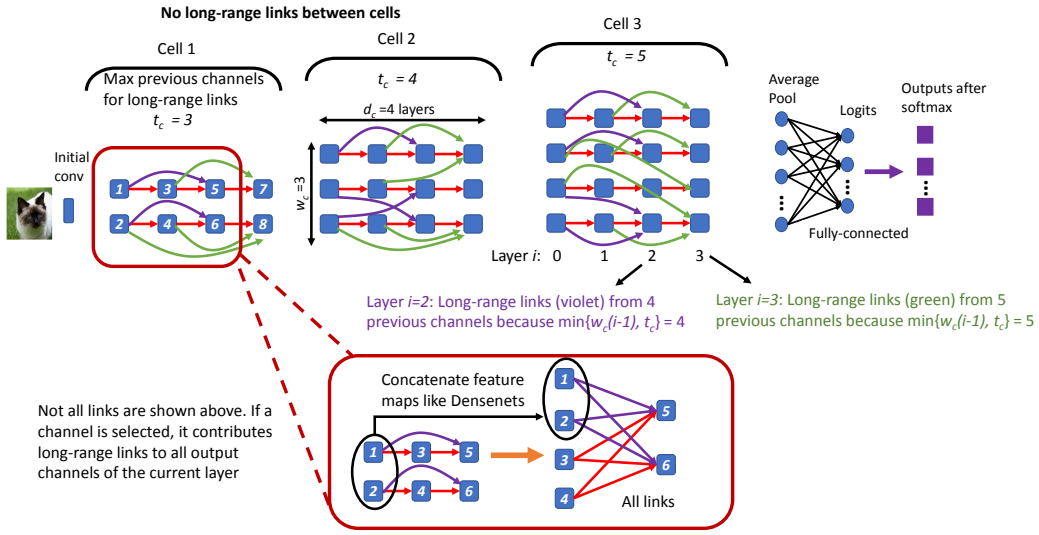


Figure 9: An example of CNN to calculate NN-Density and NN-Mass. Not all links are shown in the main figure for simplicity. The inset shows the contribution from all long-range and short-range links: The feature maps for randomly selected channels are concatenated at the current layer (similar to Densenets [Huang et al. (2017)]). At each layer in a given cell, the maximum number of channels that can contribute long-range links is given by t_c.

long-range links. Therefore, to perform experiments, we vary $\{d_c, w_c, t_c\}$ to obtain architectures with different NN-Mass and NN-Density values.

G Complete Details of the Experimental Setup

G.1 MLP Setup

We now explain more details on our MLP setup for the MNIST dataset. We create random architectures with different NN-Mass and #Params by varying t_c and d_c. Moreover, we just use a single cell for all MLP experiments. We fix $w_c = 8$ and vary $d_c \in \{16, 20, 24, 28, 32\}$. For each depth d_c, we vary $t_c \in \{0, 1, 2, \ldots, 14\}$. Specifically, for a given $\{d_c, w_c, t_c\}$ configuration, we create random long-range links at layer i by uniformly sampling $\min\{w_c(i-1), t_c\}$ neurons out of $w_c(i-1)$ activation outputs from previous $\{0, 1, \ldots, i-2\}$ layers.

We train these random architectures on the MNIST dataset for 60 epochs with Exponential Linear Unit (ELU) as the activation function. Further, each $\{d_c, w_c, t_c\}$ configuration is trained five times with different random seeds. In other words, during each of the five runs of a specific $\{d_c, w_c, t_c\}$ configuration, the shortcuts are initialized randomly so these five models are not the same. This kind of setup is used to validate that NN-Mass is indeed a topological property of deep networks, and that the specific connections inside the random architectures do *not* affect our conclusions. The results are then averaged over all runs: Mean is plotted in Fig. 2 and standard deviation, which is typically low, is also given in Fig. 2 caption. Overall, this setup results in many MLPs with different #Params/FLOPS/layers.

G.2 CNN Setup

Much of the setup for creating long-range links in CNNs is the same as that for MLPs, except we have three cells instead of just one. As explained in Appendix E, the width of the three cells is given as $wm \times [16, 32, 64]$, where wm is the width multiplier. Note that, since we have three cells of different widths (w_c), t_c also has a different value for each cell. The depth per cell d_c is the same for all cells; hence, the total depth is given by $3d_c + 4$. For instance, for 31-layer model, our $d_c = 9$. For most of our experiments, we set the total depth of the CNN as $\{31, 40, 49, 64\}$. Some of the experiments also use a total depth of $\{28, 43, 52, 58\}$.

Table 1: CNN architecture details (width multiplier = 2)

Number of Cells	Max. Long-Range Link Candidates (t_c)	Depth	Width Multiplier
3	[10,35,50] [20,45,75] [30,50,100] [40,60,120] [50,70,145]	31	2
3	[20,40,70] [30,50,100] [40,80,125] [50,105,150] [60,130,170]	40	2
3	[25,50,90] [35,80,125] [50,105,150] [70,130,170] [90,150,210]	49	2
3	[30,80,117] [50,110,150] [70,140,200] [90,175,250] [110,215,300]	64	2

Table 2: CNN architecture details (width multiplier = 1)

Number of Cells	Max. Long-Range Link Candidates (t_c)	Depth	Width Multiplier
3	[5,8,12] [10,30,50] [30,40,70] [41,61,91] [50,90,110]	31	1
3	[5,9,12] [11,31,51] [31,41,71] [41,62,92] [50,90,109]	40	1
3	[5,10,11] [11,31,52] [31,41,73] [42,62,93] [50,90,109]	49	1
3	[5,10,12] [11,32,53] [31,42,74] [42,62,94] [49,90,110]	64	1

Again, we conduct several experiments for different $\{d_c, w_c, t_c\}$ values which yield many random CNN architectures. The random long-range link creation process is the same as that in MLPs and, for CNN experiments, we have repeated all experiments three times with different random seeds. Specific numbers used for $\{d_c, w_c, t_c\}$ are given in Tables 1, 2, and 3. Each row in all tables represents a different $\{d_c, w_c, t_c\}$ configuration. Of note, all CNNs use ReLU activation function and Batch Norm layers.

For CNNs, we verify our findings on CIFAR-10 and CIFAR-100 image classification datasets. The learning rate for all models is initialized to 0.05 and follows a cosine-annealing schedule at each epoch. The minimum learning rate is 0.0 (see the end of Section H.9 for details on how we fixed these hyper-parameter values). Similar to the setup in NAS prior works, the cutout is used for data augmentation. All models are trained in Pytorch on NVIDIA 1080-Ti, Titan Xp, and 2080-Ti GPUs. This completes the experimental setup.

Table 3: CNN architecture details (width multiplier = 3)

Number of Cells	Max. Long-Range Link Candidates (t_c)	Depth	Width Multiplier
3	[10,30,50] [40,60,90] [70,90,130] [100,120,170] [130,150,210]	31	3
3	[11,31,51] [42,62,92] [72,93,133] [103,123,173] [133,153,212]	40	3
3	[11,31,52] [43,63,93] [73,95,135] [104,124,176] [134,154,214]	49	3
3	[12,32,52] [44,64,95] [76,96,136] [106,126,178] [135,156,216]	64	3

H Additional Results

H.1 More MNIST training convergence results

We pick two groups of three models each – (X, Y, and Z) – and – (D, E, and F) and plot their training accuracy *vs.* epochs. Models X and Y have similar NN-Mass but Y has more #Params and depth than X. Model Z has far fewer layers and nearly the same #Params as X, but has higher NN-Mass. Fig. 10(c) shows the training convergence results for all three models. As is evident, the training convergence of model X (8.3K Params, 24-layers) nearly coincides with that of model Y (9.0K Params, 32-layers). Moreover, even though model Z (8.3K Params, 16-layers) is shallower than the 32-layer model Y (and has far fewer #Params), training convergence of Z is significantly faster than that of Y (due to higher NN-Mass and, therefore, better LDI). These results clearly show the evidence towards theoretical insights in Proposition 2, and emphasize the importance of topological properties of neural architectures in characterizing gradient propagation and model performance. Similar observations are found among models D, E, and F.

H.2 Results on synthetic data

In this section, we design a few synthetic experiments for MLP experiments to verify that our observations in Section 4.2 hold for diverse datasets. Specifically, we design three datasets – Seg20, Seg30, and Circle20 (or just Circle). Fig. 11(a) illustrates the Seg4 dataset where the range $[0.\ 1]$ is broken into 4 segments. Similarly, Seg20 (Seg30) breaks down the linear line into 20 (30) segments. The classification problem has two classes (each alternate segment is a single class).

Fig. 11(b) shows the circle dataset where a unit circle is broken down into concentric circles (regions between circles make a class and we have two total classes). The details of these datasets are given in Table 4. Of note, we have used the ReLU activation function for these experiments (unlike ELU used for MNIST).

For the above synthetic experiments, we once again conduct the following experiments: (*i*) We explore the impact of varying #Params and NN-Mass on the test accuracy. (*ii*) We demonstrate how LDI depends on NN-Mass and #Params.

Test Accuracy As shown in Fig. 12(a, b, c) and Fig. 12(d, e, f), NN-Mass is a much better metric to characterize the model performance of DNNs than the number of parameters. Again,

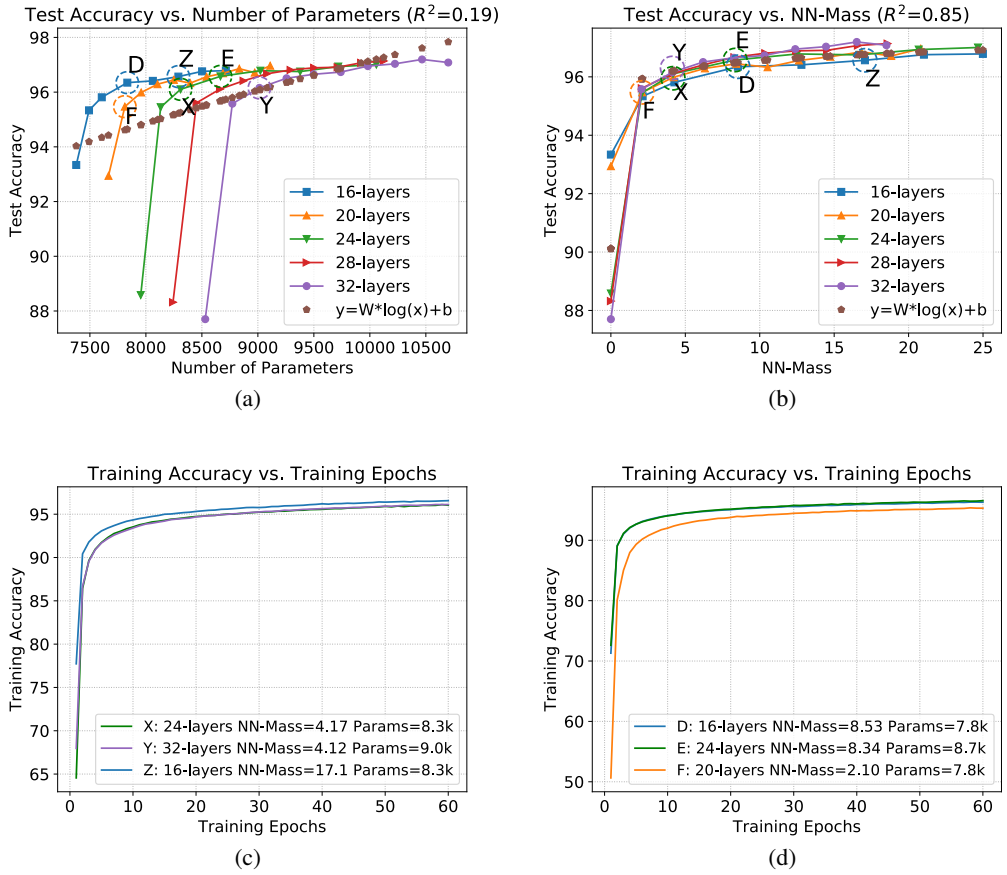
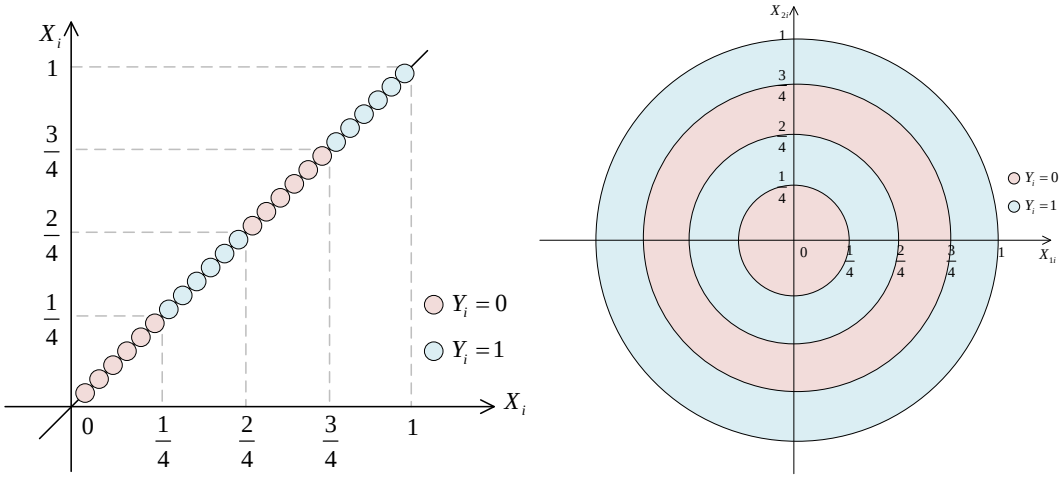


Figure 10: More MNIST training convergence results (a, b are repeated from Fig. 2 but have been annotated with different models (X, Y, Z, D, E, F)): (a) Models with different #Params achieve similar test accuracy. (b) Test accuracy curves of models with different depths/#Params concentrate when plotted against NN-Mass (test accuracy std. dev.$\sim$ 0.05-0.34%). (c,d) Models X and Y have the same NN-Mass and achieve very similar training convergence, even though they have highly different #Params and depth. Model Z has significantly fewer layers than Y but the same #Params and yet achieves faster training convergence than Y (Z has higher NN-Mass than Y). The above conclusions hold true for models D, E, and F. Note that, the training convergence curves for similar NN-Mass models are coinciding.

Table 4: Description of our generated Synthetic Datasets

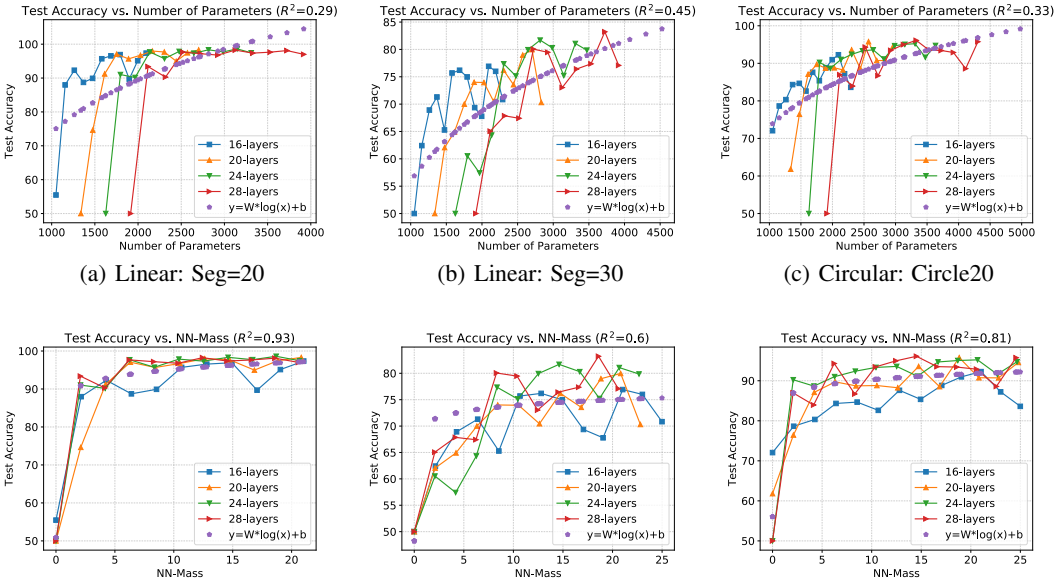
Dataset name	Description: Training Set, $i \in [1, 60000]$; Test Set, $i \in [1, 12000]$
Seg20	Feature: $[X_i, X_i]$, Label: Y_i, $X_i = sample(\frac{1}{20}[\lfloor\frac{i}{20}\rfloor, \lfloor\frac{i}{20}\rfloor + 1])$, $Y_i = \lfloor\frac{i}{20}\rfloor mod 2$
Seg30	Feature: $[X_i, X_i]$, Label: Y_i, $X_i = sample(\frac{1}{30}[\lfloor\frac{i}{30}\rfloor, \lfloor\frac{i}{30}\rfloor + 1])$, $Y_i = \lfloor\frac{i}{30}\rfloor mod 2$
Circle (Circle20)	Feature: $[X_{1i}, X_{2i}]$, Label: Y_i, $X_{1i} = L_i * cos(rand\_num), X_{2i} = L_i * sin(rand\_num), L_i = sample(\frac{1}{20}[\lfloor\frac{i}{20}\rfloor, Y_i = \lfloor\frac{i}{20}\rfloor mod 2$

we quantitatively analyze the above results by generating a linear fit between test accuracy vs. log(#Params) and log(NN-Mass). Similar to the MNIST case, our results show that R^2 of test accuracy vs. NN-Mass is much higher than that for #Params.



(a) Seg4 (b) Circle4

Figure 11: Illustration of synthetic datasets Seg4 and Circle4: (a). Seg20 (Seg30) dataset is similar to Seg4, but divides the $[0,\ 1]$ range into 20 (30) segments. (b). Circle (or Circle20) dataset is similar to Circle4, but divides a unit circle into 20 concentric circles.



(a) Linear: Seg=20 (b) Linear: Seg=30 (c) Circular: Circle20

(d) Linear: Seg=20 (e) Linear: Seg=30 (f) Circular: Circle20

Figure 12: Synthetic results: (a, b, c) Models with different #Params achieve similar test accuracy across all synthetic datasets. (d, e, f) Test accuracy curves for the same set of models come closer together when plotted against NN-Mass.

Layerwise Dynamical Isometry Fig. 13 shows the LDI results for the Circle20 dataset. Again, higher NN-Mass leads to higher initial singular value. Moreover, NN-Mass is better correlated with LDI than #Params. Hence, this further emphasizes why networks with similar NN-Mass (instead of #Params) result in a more similar model performance.

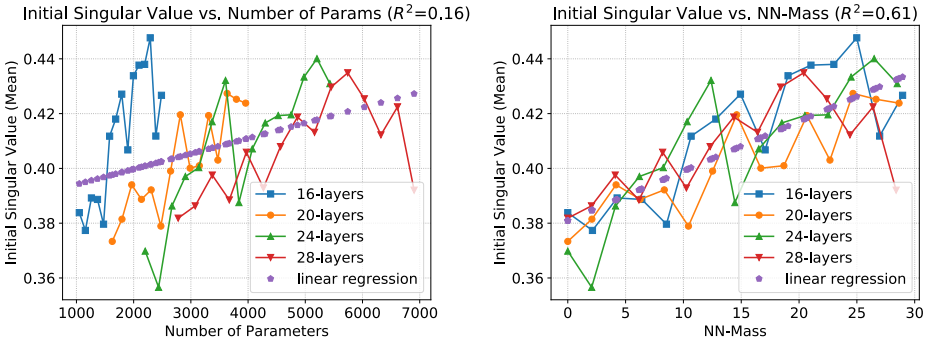


Figure 13: Synthetic results (Circle20 datasets): Mean singular value of $\boldsymbol{J_{i,i-1}}$ is much better correlated with NN-Mass than with #Params.

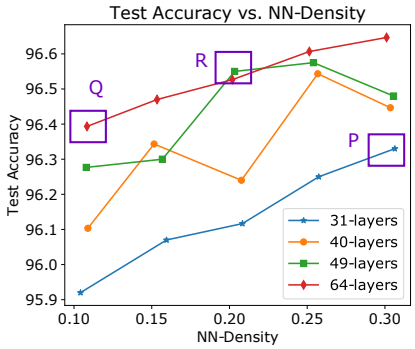


Figure 14: CIFAR-10 Width Multiplier $wm = 2$: Shallower models with higher density can reach comparable accuracy to deeper models with lower density. This does not help since models with different depths achieve comparable accuracies at different densities.

H.3 Impact of Varying NN-Density

As a baseline, we show that NN-Density cannot predict the accuracy of models with different depths. We train different deep networks with varying NN-Density (see Table 1 models in Appendix G). Fig. 14 shows that shallower models with higher density can reach accuracy comparable to deeper models with lower density (which is quite reasonable since the shallower models are more densely connected compared to deeper networks, thereby promoting more effective information flow in shallower CNNs despite having significantly fewer parameters). However, NN-Density alone does not identify models (with different sizes/compute) that achieve similar accuracy: CNNs with different depths achieve comparable test accuracies at different NN-Density values (*e.g.*, although a 31-layer model with $\rho_{avg} = 0.3$ performs close to 64-layer model with $\rho_{avg} = 0.1$, a 49-layer model with $\rho_{avg} = 0.2$ already outperforms the test accuracy of the above 64-layer model; see models P, Q, R in Fig. 14). Therefore, NN-Density alone is not sufficient.

H.4 Impact of Varying Width Multiplier on CIFAR-10

We now explore the impact of varying model width. In our CNN setup, we control the width of the models using *width multipliers* (wm)[5] [Zagoruyko & Komodakis (2016); Howard et al. (2017)]. The above results are for $wm = 2$. For lower width CNNs ($wm = 1$), Fig. 15(a) shows that models in boxes U and V concentrate into the buckets W and Z, respectively (see also other buckets). Note that, the 31-layer models do not fall within the buckets (see blue line in Fig. 15(b)). We hypothesize that this could be because the capacity of these models is too small to reach high accuracy. This does not happen for CNNs with higher width. Specifically, Fig. 15(c) shows the results for $wm = 3$. As evident, models with 6M-7M parameters achieve comparable test accuracy as models with up

[5]Base #channels in each cell is [16,32,64]. For $wm = 2$, cells will have [32,64,128] channels per layer.

to 16M parameters (*e.g.*, bucket Y in Fig. 15(d) contains models ranging from {31 layers, 6.7M parameters}, all the way to {64 layers, 16.7M parameters}). Again, for all widths, the goodness-of-fit (R^2) for linear fit between test accuracy and log(NN-Mass) achieves high values (0.74-0.90 as shown in Fig. 16 in Appendix H.5).

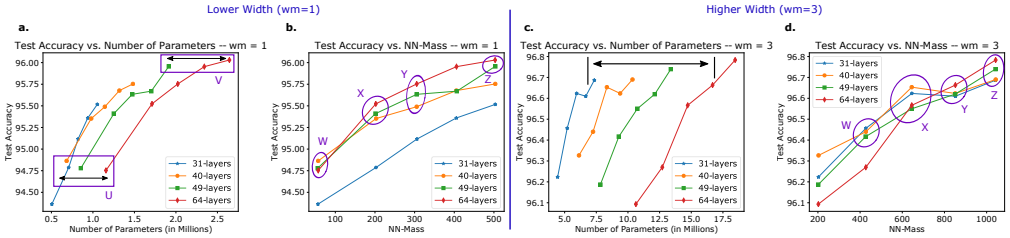


Figure 15: Similar observations hold for low- ($wm = 1$) and high-width ($wm = 3$) models: (a, b) Many models with very different #Params (boxes U and V) cluster into buckets W and Z (see also other buckets). (c, d) For high-width, we observe a significantly tighter clustering compared to the low-width case. Results are reported as the mean of three runs (std. dev. $\sim 0.1\%$).

H.5 R-Squared of CIFAR-10 Accuracy vs. NN-Mass

Fig. 16 shows the impact of increasing model widths on R^2 of linear fit between test accuracy and log(NN-Mass).

H.6 Comparison between NN-Mass and Parameter Counting for CNNs

For MLPs, we have shown that NN-Mass significantly outperforms #Params for predicting model performance. For CNNs, we quantitatively demonstrate that while parameter counting can be a useful indicator of test accuracy for models with low width (but still not as good as NN-Mass), as the width increases, parameter counting completely fails to predict test accuracy. Specifically, in Fig. 17(a), we fit a linear model between test accuracy and log(#parameters) and found that the R^2 for this model is 0.76 which is slightly lower than that obtained for NN-Mass ($R^2 = 0.84$, see Fig. 17(b)). When the width multiplier of CNNs increases to three, parameter counting completely fails to fit the test accuracies of the models ($R^2 = 0.14$). In contrast, NN-Mass significantly outperforms parameter counting for $wm = 3$ as it achieves an $R^2 = 0.90$. This demonstrates that NN-Mass is indeed a significantly stronger indicator of model performance than parameter counting.

H.7 Results for CIFAR-100

Results for CIFAR-100 dataset are shown in Fig. 18. As evident, several models achieve similar accuracy despite having highly different number of parameters (*e.g.*, see models within box W in Fig. 18(a)). Again, these models get clustered together when plotted against NN-Mass. Specifically, models within box W in Fig. 18(a) fall into buckets Y and Z in Fig. 18(b). Hence, models that got clustered together for CIFAR-10 dataset, also get clustered for CIFAR-100. To quantify the above results, we fit a linear model between test accuracy and log(NN-Mass) and, again, obtain a high $R^2 = 0.84$ (see Fig. 18(c)). Therefore, our observations hold true across multiple image classification datasets.

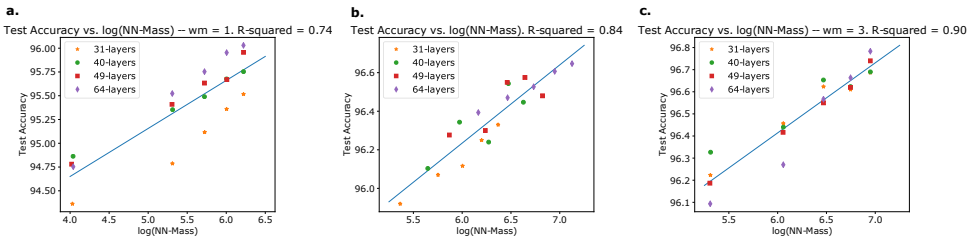


Figure 16: Impact of varying width: (a) Width multiplier, $wm = 1$, (b) $wm = 2$, and (c) $wm = 3$. As width increases, the capacity of small (shallower) models increases and, therefore, the accuracy-gap between models of different depths reduces. Hence, the R^2 for linear fit increases as width increases.

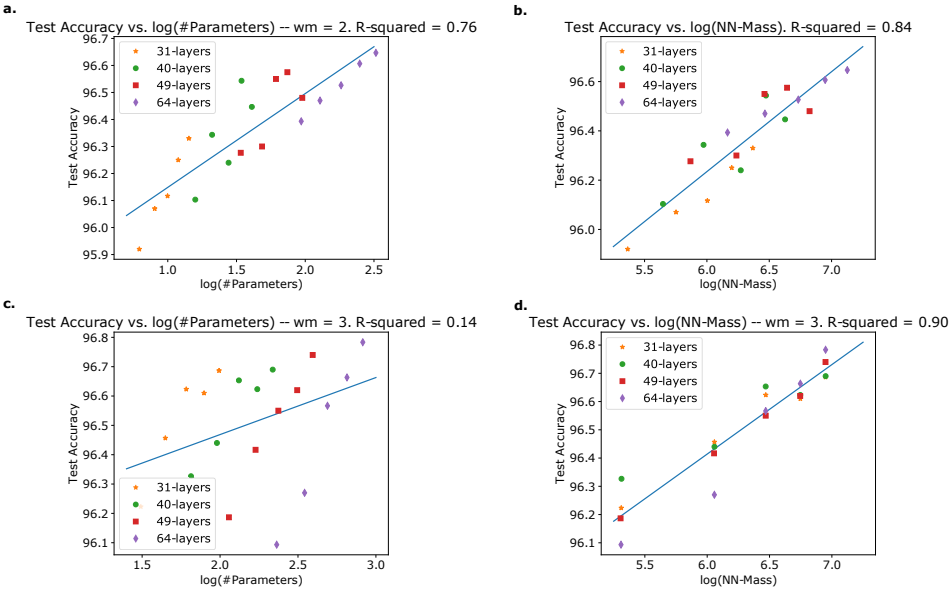


Figure 17: NN-Mass as an indicator of model performance compared to parameter counting. (a) For $wm = 2$, log(#parameters) fits the test accuracy with an $R^2 = 0.76$. (b) For the same $wm = 2$ case, log(NN-Mass) fits the test accuracy with a higher $R^2 = 0.84$. (c) For higher width ($wm = 3$), parameter counting completely fails to fit the test accuracy of various models ($R^2 = 0.14$). (d) In contrast, NN-Mass still fits the accuracies with a high $R^2 = 0.9$.

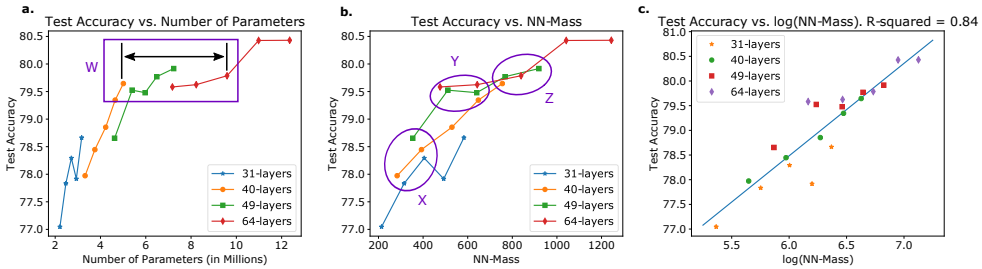


Figure 18: Similar results are obtained for CIFAR-100 ($wm = 2$). (a) Models in box W have highly different #parameters but achieve similar accuracy. (b) These models get clustered into buckets Y and Z. (c) The R^2 value for fitting a linear regression model is 0.84 which shows that NN-Mass is a good predictor of test accuracy. Results are reported as the mean of three runs (std. dev. $\sim 0.2\%$).

H.8 Results for Floating Point Operations (FLOPS)

All results for FLOPS (of CNN architectures in Tables 1, 2, and 3) are shown in Fig. 19. As evident, models with highly different number of FLOPS often achieve similar test accuracy. As shown earlier, many of these CNN architectures cluster together when plotted against NN-Mass.

H.9 NN-Mass for directly designing compressed architectures

Our theoretical and empirical evidence shows that NN-Mass is a reliable indicator for models which achieve a similar accuracy despite having different number of layers and parameters. Therefore, this observation can be used for directly designing efficient CNNs as follows:

- First, train a reference big CNN (with a large number of parameters and layers) which achieves very high accuracy on the target dataset. Calculate its NN-Mass (denoted m_L).
- Next, create a *completely new and significantly smaller model* using far fewer parameters and layers, but with a NN-Mass (m_S) comparable to or higher than the large CNN. This process is very fast as the new model is created without any *a priori* training. For instance, to design an efficient CNN of width w_c and depth per cell d_c and NN-Mass $m_S \approx m_L$,

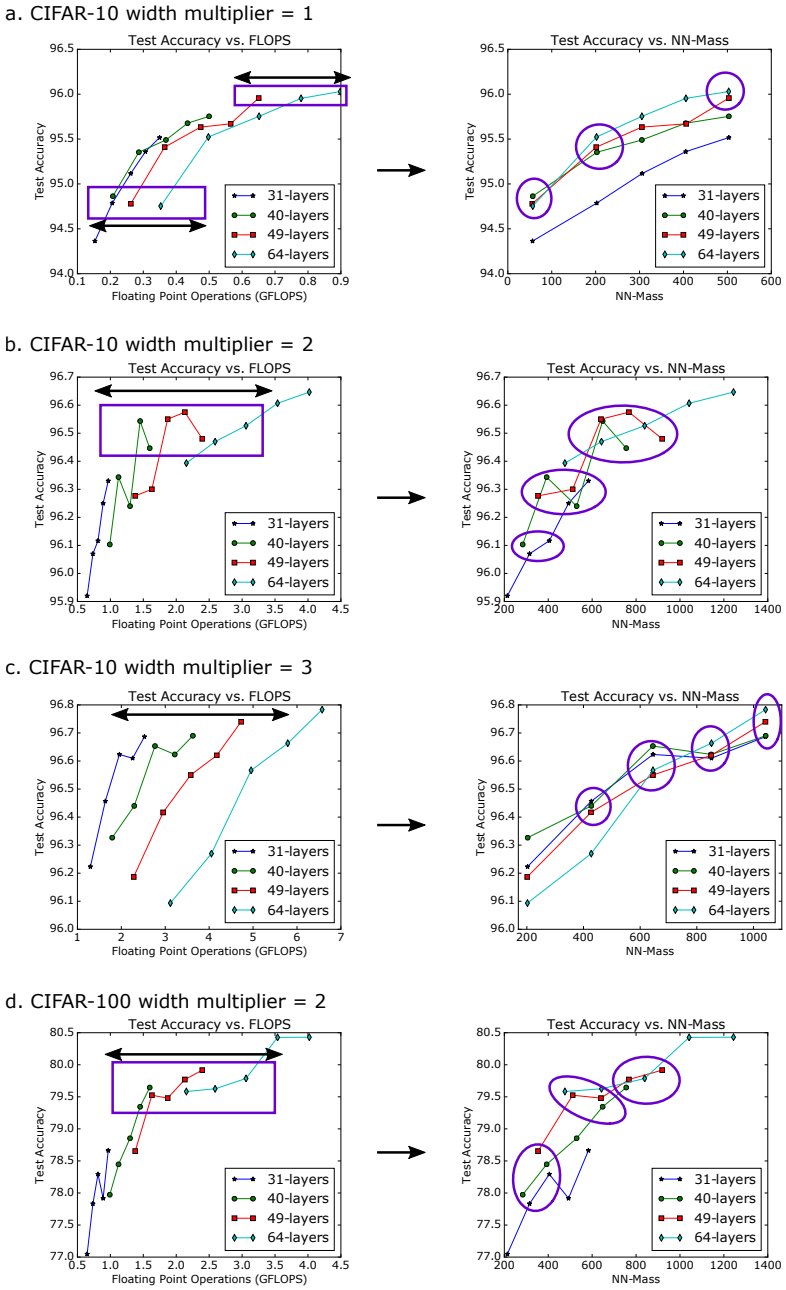


Figure 19: Models with highly different number of FLOPS achieve similar test accuracies. The CNN architectures are the same as those used in Figures 4, 15, and 18. The pattern for FLOPS is very similar to that for the number of parameters. Hence, these results show that models with both highly different number of parameters and FLOPS can achieve similar test accuracy. Again, these models cluster together when plotted against NN-Mass.

we only need to find how many long-range links to add in each cell. Since, NN-Mass has a closed form equation (*i.e.*, Eq. 2), a simple search over the number of long-range links can directly determine NN-Mass of various architectures. Then, we select the architecture with the NN-Mass close to that of the reference CNN. Unlike current manual or NAS-based methods, our approach does not require training of individual architectures during the search.

- Since NN-Mass of the smaller model is similar to that of the reference CNN, our theoretical as well as empirical results suggest that the newly generated model will lose only a small

Table 5: Exploiting NN-Mass for Model Compression on CIFAR-10 Dataset. All our experiments are reported as mean $\pm$ standard deviation of three runs. DARTS results are from [Liu et al. (2018)].

Model	Architecture design method	#Parameters/ #FLOPS	#layers	Specialized search space?	NN-Mass	Test Accuracy
DARTS (first order)	NAS [Liu et al. (2018)]	3.3M/–	–	Yes	–	$97.00 \pm 0.14\%$
DARTS (second order)	NAS [Liu et al. (2018)]	3.3M/–	–	Yes	–	$\mathbf{97.24 \pm 0.09\%}$
Train large models to be compressed	Manual	11.89M/3.63G	64	No	1126	$97.02 \pm 0.06\%$
	Manual	8.15M/2.54G	64	No	622	$96.99 \pm 0.07\%$
Proposed	Directly via NN-Mass	**5.02M/1.59G**	40	No	755	$\mathbf{97.00 \pm 0.06\%}$
Proposed	Directly via NN-Mass	**4.69M/1.51G**	37	No	813	$\mathbf{96.93 \pm 0.10\%}$
Proposed	Directly via NN-Mass	**3.82M/1.2G**	31	No	856	$\mathbf{96.82 \pm 0.05\%}$

amount of accuracy, while significantly reducing the model size. To validate this, we train the new, significantly smaller model and compare its test accuracy against that of the original large CNN.

We train our models for 600 epochs on the CIFAR-10 dataset (similar to the setup in DARTS [Liu et al. (2018)]). Table 5 summarizes the number of parameters, FLOPS, and test accuracy of various CNNs. We first train two large CNN models of about 8M and 12M parameters with NN-Mass of 622 and 1126, respectively; both of these models achieve around 97% accuracy. Next, we train three significantly smaller models: (*i*) A 5M parameter model with 40 layers and a NN-Mass of 755, (*ii*) A 4.6M parameter model with 37 layers and a NN-Mass of 813, and (*iii*) A 31-layer, 3.82M parameter model with a NN-Mass of 856.

We set the NN-Mass of our smaller models between 750-850 (*i.e.*, within the 600-1100 range of the manually-designed CNNs). Interestingly, *we do not need to train any intermediate architectures* to arrive at the above efficient CNNs. Indeed, classical NAS involves an initial "search-phase" over a space of operations to find the architectures [Zoph et al. (2018)]. In contrast, our efficient models can be directly designed using the closed form Eq. 2 of NN-Mass (as explained in the beginning of this section), which does not involve any intermediate training or even an initial search-phase like prior NAS methods. As explained earlier, this is possible because NN-Mass can identify models with similar performance *a priori* (*i.e.*, without any training)!

As evident from Table 5, our 5M parameter model reaches a test accuracy of 97.00%, while the 4.6M (3.82M) parameter model obtains 96.93% (96.82%) accuracy on the CIFAR-10 test set. Clearly, all these accuracies are either comparable to, or slightly lower ($\sim 0.2\%$) than the large CNNs, while reducing #Params/FLOPS by up to $3\times$ compared to the 11.89M-parameter/3.63G-FLOPS model. Moreover, DARTS [Liu et al. (2018)], a competitive NAS baseline, achieves a comparable (97%) accuracy with slightly lower 3.3M parameters. However, the search space of DARTS (like all other NAS techniques) is very specialized and utilizes many state-of-the-art innovations such as depth-wise separable convolutions [Howard et al. (2017)], dilated convolutions [Yu & Koltun (2015)], *etc*. On the contrary, we use regular convolutions with only concatenation-type long-range links in our work and present a theoretically grounded approach. Indeed, our current objective is not to beat DARTS (or any other NAS technique), but rather underscore the topological properties that should guide the efficient architecture design process. Ultimately, this theoretical knowledge (and its extensions to other kinds of networks) can help us drastically reduce the search space of NAS by directly removing architectures that are unlikely to improve accuracy.

A note on hyper-parameter (*e.g.*, initial learning rate) optimization. Note that, throughout this work, we optimized the hyper-parameters such as initial learning rate for the largest models and then used the same initial learning rate for the smaller models. Hence, if these hyper-parameters were further optimized for the smaller models, the gap between the accuracy curves in Figures 15, 18, 19, *etc.*, would reduce further (*i.e.*, the clustering on NN-Mass plots would further improve). Similarly, the accuracy gap between compressed models and the large CNNs would reduce even more in Table 5 if the hyper-parameters were optimized for the smaller models as well. We did not optimize the initial learning rates, *etc.*, for the smaller models as it would have resulted in an explosion in terms of number of experiments. Hence, since our focus is on topological properties of CNN architectures, we fixed the other hyper-parameters as described above.