# Supplementary Information: How does topology influence gradient propagation and model performance of deep networks with DenseNet-type skip connections?

## A. DNNs/CNNs with random skip connections are Small-World Networks

In network theory, a small world network is formed by superimposing a random network $\mathcal{R}$ on top of a lattice network $\mathcal{G}$ (see Fig. 7) [20, 22]. As a result, these networks have both short-range and long-range links. Similarly, the DNNs/CNNs considered in our work have both short-range (due to layer-by-layer convolutions) and long-range links (due to random skip connections; see Fig. 1(a)). This is illustrated in Fig. 7.

## B. Derivation of Density of a Cell

Note that, the maximum number of neurons contributing skip connections at each layer in cell $c$ is given by $t_c$. Also, for a layer $i$, possible candidates for skip connections = 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 skip connections. For these layers, we use all available neurons for skip connections. Therefore, for a layer $i$, #skip connections ($l_i$) is given by:

$$l_i = \begin{cases} w_c(i-1) \times w_c & \text{if } t_c > w_c(i-1) \\ t_c \times w_c & \text{otherwise} \end{cases} \tag{4}$$

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

$$l_c = w_c \sum_{i=2}^{d_c-1} \min\{w_c(i-1), t_c\} \tag{5}$$

On the other hand, the total number of possible skip connections within a cell ($L$) is simply the sum of possible candidates at each layer:

$$\begin{aligned} L &= \sum_{i=2}^{d_c-1} w_c(i-1) \times w_c = w_c^2 \sum_{i=2}^{d_c-1} (i-1) \\ &= w_c^2[1+2+\ldots+(d_c-2)] \\ &= \frac{w_c^2(d_c-1)(d_c-2)}{2} \end{aligned} \tag{6}$$

Using Eq. 5 and Eq. 6, we can rewrite Eq. 1 as:

$$\rho_c = \frac{2\sum_{i=2}^{d_c-1} \min\{w_c(i-1), t_c\}}{w_c(d_c-1)(d_c-2)} \tag{7}$$

□

## C. Proof of Proposition 1

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

*Proof.* As shown in Fig. 7, deep networks with shortcut connections can be represented as small-world networks consisting of two parts: (*i*) lattice network containing only the layer-by-layer links, and (*ii*) random network superimposed on top of the lattice network to account for random skip connections. For sufficiently deep networks, the average degree for the lattice network will be just the width $w_c$ of the network. We consider the connections as undirected connections; hence each of the connection is counted only once. The average degree of the randomly added skip connections $\bar{k}_{\mathcal{R}|\mathcal{G}}$ is given by:

$$\begin{aligned} \bar{k}_{\mathcal{R}|\mathcal{G}} &= \frac{\textit{Number of skip connections added by } \mathcal{R}}{\textit{Number of nodes}} \\ &= \frac{w_c \sum_{i=2}^{d_c-1} \min\{w_c(i-1), t_c\}}{w_c d_c} \\ &= \frac{m(d_c-1)(d_c-2)}{2d_c^2} \quad \text{(Eq. 2 for one cell, } N_c = 1) \\ &\approx \frac{m}{2} \quad \text{(when } d_c >> 2, \textit{e.g.}, \text{ for deep networks)} \end{aligned} \tag{8}$$

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).** *Consider the case of deep linear networks with concatenation-type skip connections, where each layer is initialized using independently and identically distributed values with initialization variance $q$. For this setup, suppose we are given a small network $f_S$ (depth $d_S$) and a large network $f_L$ (depth $d_L$, $d_L >> d_S$), both with same initialization scheme, NN-Mass $m$, and width $w_c$. Then, the mean singular value of the initial layerwise Jacobian ($\mathbb{E}[\sigma]$) for both networks is bounded as follows:*

$$\sqrt{q(w_c+m/2)} - \sqrt{qw_c} \leq \mathbb{E}[\sigma] \leq \sqrt{q(w_c+m/2)} + \sqrt{qw_c}$$

*That is, the LDI for both models does not depend on the depth if the initialization variance ($q$) for each layer is depth-independent (which is the case for many initialization schemes). Hence, for such networks, models with similar width and NN-Mass result in similar gradient properties, even if their depths and number of parameters are different.*

*Proof.* We now formally prove the above result under the assumption of deep linear networks [29, 12, 1, 10]. As stated in the main text, layerwise Jacobians for linear networks follow a Gaussian distribution. We first prove the above result

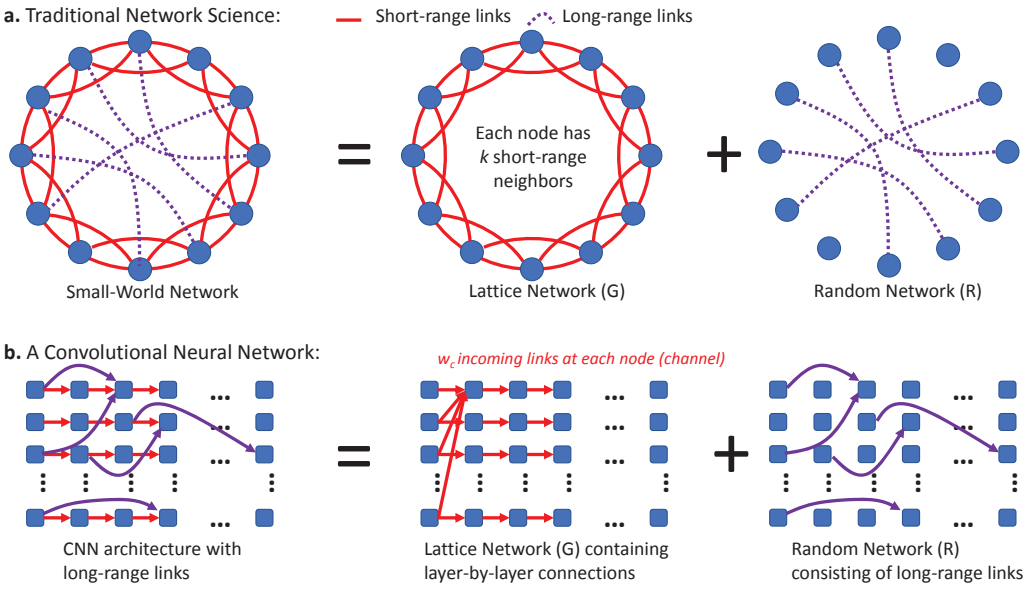



Figure 7: (a) Small-World Networks in traditional network science are modeled as a superposition of a lattice network ($\mathcal{G}$) and a random network $\mathcal{R}$ [32, 22, 20]. (b) A DNN/CNN with both short-range and long-range links can be similarly modeled as a random network superimposed on a lattice network. Not all links are shown for simplicity.

for a matrix $M \in \mathbb{R}^{H\times W}$ with $H$ rows and $W$ columns, where $H \geq W$, and all entries independently initialized with a Gaussian Distribution $\mathcal{N}(0, q)$. Towards the end of the proof, we will show how the results for matrix $M$ above apply to the correct layerwise Jacobians for a linear network ($\boldsymbol{J_{i,i-1}} = \boldsymbol{W_i}$, where $\boldsymbol{W_i}$ is initialized as Gaussian with variance $q$).

To calculate the mean singular value of $M$, we perform Singular Value Decomposition (SVD) for matrix $M$:

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

Given the $i^{th}$ row vector $\vec{u_i} \in \mathbb{R}^H$ in $U$, and the $i^{th}$ 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 $\sigma_i^2$ using the following product of SVD:

$$M^T M = U\Sigma V^T V \Sigma^T U^T = U(\Sigma\Sigma^T)U^T \tag{9}$$

Consequently, the square of singular value ($\sigma_i^2$) of $M$ are the eigenvalues ($\lambda_i^{'}$) of $M^T M$:

$$\sigma_i^2 = \lambda_i^{'} \tag{10}$$

Mathematically, $\frac{1}{\sqrt{q}}M$ is a standard Gassian Random Matrix, i.e., all the elements of $\frac{1}{\sqrt{q}}M$ follow an i.i.d. standard Guassian Distribution $\mathcal{N}(0, 1)$. From the theory of random matrix, the matrix $\frac{1}{q}M^T M$ is a Wishart ensemble [33]. Therefore, the distribution of the eigenvalues $(\lambda_1, \lambda_2, ..., \lambda_i, ...)$ of $\frac{1}{q}M^T M$ is Wishart Distribution. Then, for the Wishart Distribution, we know that the expectation of $(\lambda_1, \lambda_2, ..., \lambda_i, ...)$ [18]:

$$\mathbb{E}[(\lambda_1, \lambda_2, ..., \lambda_i, ...)] = (H, H, .., H, ...) \tag{11}$$

Clearly, $\lambda_i^{'} = q\lambda_i$. Combing Eq. 10 and Eq. 11, we get the following results:

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

Eq. 12 states that, by keeping the same initialization variance $q$, for a Gaussian $M \in \mathbb{R}^{H\times W}$ with $H \geq W$, $\mathbb{E}[\sigma_i^2]$ is dependent on number of rows $H$, and does *not* depend on W. To empirically verify this, we simulate several Gaussian matrices of widths $W \in \{20, 40, 80, 120, 160, 200\}$ and $H \in [200, 1200]$. We plot $\mathbb{E}[\sigma_i^2]$ *vs.* $H$ in Fig. 8. As evident, the means of square of singular values $\mathbb{E}[\sigma_i^2]$ are nearly coinciding for different $W$, thereby showing that mean singular value indeed depends only on $H$.

While Eq. 12 analytically shows the relationship between $\mathbb{E}[\sigma_i^2]$ and $H$, the LDI [13] depends on the mean singular value $\mathbb{E}[\sigma_i]$ (and not its square). Although, the analytical expression for $\mathbb{E}[\sigma_i]$ is still an unsolved mathematical problem, it is still possible to provides its bounds. Us-

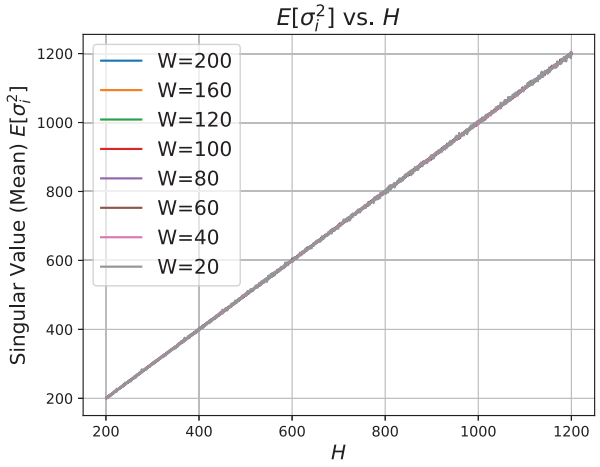



Figure 8: Mean of square of singular value $\mathbb{E}[\sigma_i^2]$ only increases with $H$ while varying $W$ when $H \geq W$.

ing the theory of Marchenko–Pastur distribution, all singular values for matrix $M$ asymptotically lie in the interval $[\sqrt{qH} - \sqrt{qW}, \sqrt{qH} + \sqrt{qW}]$ [19, 27]. Therefore, the mean singular value also asymptotically lies in the interval $[\sqrt{qH} - \sqrt{qW}, \sqrt{qH} + \sqrt{qW}]$.

We now consider the initial layerwise Jacobian matrices ($\boldsymbol{J_{i,i-1}}$) for the deep linear network scenario (i.e., $\boldsymbol{J_{i,i-1}} = \boldsymbol{W_i}$, where $\boldsymbol{W_i}$ is initialized as Gaussian with variance $q$). 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, the above bounds for the mean singular value become:

$$\sqrt{q(w_c + m/2)} - \sqrt{qw_c} \leq \mathbb{E}[\sigma] \leq \sqrt{q(w_c + m/2)} + \sqrt{qw_c} \tag{13}$$

The above bound states that if the initialization variance $q$ is not dependent on depth (which is true for many initialization schemes), then, layerwise mean singular values for the given deep linear networks $f_S$ (depth $d_S$) and $f_L$ (depth $d_L, d_L >> d_S$) depends only on width $w_c$ and NN-Mass $m$. Therefore, for such deep networks, if $w_c$ and $m$ are the same, their layerwise dynamical isometry property (i.e., the mean singular values of initial layerwise Jacobians) has the same bounds. In other words, if two networks $f_S$ and $f_L$ have same width $w_c$ and NN-Mass $m$, they have similar gradient properties (i.e., LDI) even if they have significantly different depth and number of parameters. □

To empirically verify the Proposition 2 result, we plot the mean singular values as well as the bounds in (13) for Gaussian distributed matrices of size $(w_c + m/2, w_c)$ *vs.* NN-Mass ($m$) in Fig. 1(b) in the main paper. Clearly, the mean singular values for these simulated Jacobians fall within the above bounds. We will explicitly demonstrate in our experiments that NN-Mass is correlated with LDI for actual non-linear deep networks.

## 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 DenseNets [7], Resnets [5], *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 [30], 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 [37] and Mobilenets [6]). 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 skip connections is given by $t_c = [3, 4, 5]$. That is, the first cell can have a maximum of three skip connection *candidates* per layer (*i.e.*, previous channels that can supply skip connections), the second cell can have a maximum of four skip connection candidates per layer, and so on. Moreover, as mentioned before, we randomly choose $\min\{w_c(i-1), t_c\}$ channels for skip connections at each layer. The inset of Fig. 9 shows how skip connections 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 skip connections. Therefore, to perform experiments, we vary $\{d_c, w_c, t_c\}$ to obtain architectures with different

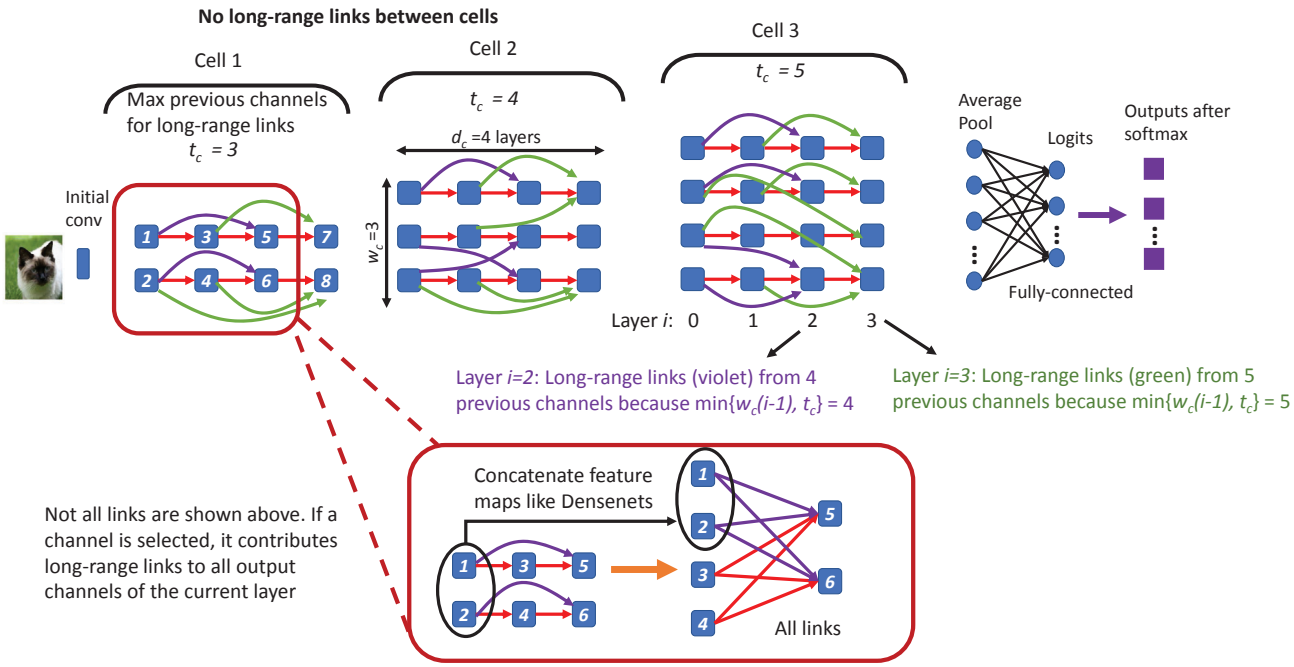



Figure 9: An example of CNN to calculate NN-Mass. Not all links are shown in the main figure for simplicity. The inset shows the contribution from all long-range and short-range links: The feature maps for randomly selected channels are concatenated at the current layer (similar to DenseNets [7]). At each layer in a given cell, the maximum number of channels that can contribute skip connections is given by $t_c$.

NN-Mass 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 skip connections 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. 3 and standard deviation, which is typically low, is also given in Fig. 3 caption. Overall, this setup results in many MLPs with different #Params/FLOPS/layers.

### G.2. CNNs with DenseNet-type Skip Connections

Much of the setup for creating concatenation-type skip connections 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 $\{28, 43, 52, 58\}$.

Again, we conduct several experiments for different $\{d_c, w_c, t_c\}$ values which yield many random CNN architectures. The random skip connection 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 3, 4, and 5. 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.

Table 3: CNN architecture details (width multiplier = 2)

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

Table 4: CNN architecture details (width multiplier = 1)

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

Table 5: CNN architecture details (width multiplier = 3)

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

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.10 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.

### G.3. MobileNet-v2 Setup

We create random MobileNet-v2-like architectures with different NN-Mass, #Params, and MACs by varying $N_c$ and width-multiplier. The standard MobileNet-v2 has $\{1, 2, 3, 4, 3, 3, 1\}$ inverted residual blocks with width (number of channels) $\{16, 24, 32, 64, 96, 160, 320\}$. Correspondingly, our searched compressed MobileNet-v2 has $\{2, 3, 4, 5, 4, 3, 3, 1\}$ inverted residual blocks with the same width as standard MobileNet-v2. Furthermore, we sampled the width-multiplier as $\{0.15, 0.35, 0.6, 0.75, 0.9, 1.0\}$.

As for the training process, we use the same training hyperparameters for all networks. We use SGD with $momentum = 0.9$ and $weight - decay = 4 * 10^{-5}$ as the optimizer. Moreover, we set the training epochs as 150, batch size as 256, and initial learning rate $lr_0 = 0.05$. For epoch $e$, the corresponding learning rate $lr_e = \frac{lr_0}{2}(1 + cos(\frac{e-1}{150}\pi))$. All models are trained in Pytorch on NVIDIA 1080-Ti, TitanXp, 2080-Ti, V100, and 3090 GPUs. This completes the experimental setup.

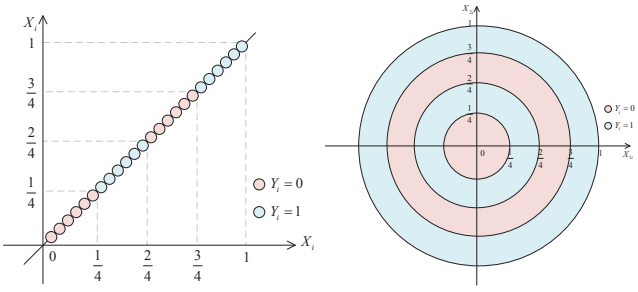
## H. Additional Results for DenseNet-type CNNs/MLPs

All results below are for DenseNet-type CNNs/MLPs.

### H.1. 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. 10(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. 10(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 6. Of note, we have used the ReLU activation function for these experiments (unlike ELU used for MNIST).




(a) Seg4




(b) Circle4

Figure 10: 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.

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. 11(a, b, c) and Fig. 11(d, e, f), NN-Mass is a much better metric to characterize the model performance of DNNs than the number of parameters. Again, 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.

**Layerwise Dynamical Isometry** Fig. 12 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.

### H.2. Impact of Varying NN-Density

NN-Density ($\rho_{avg}$) is defined as the average cell-density ($\rho_c$, see Definition 3) across all cells in a DNN. 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 3 models in Appendix G). Fig. 13 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. 13). Therefore, NN-Density alone is not sufficient.

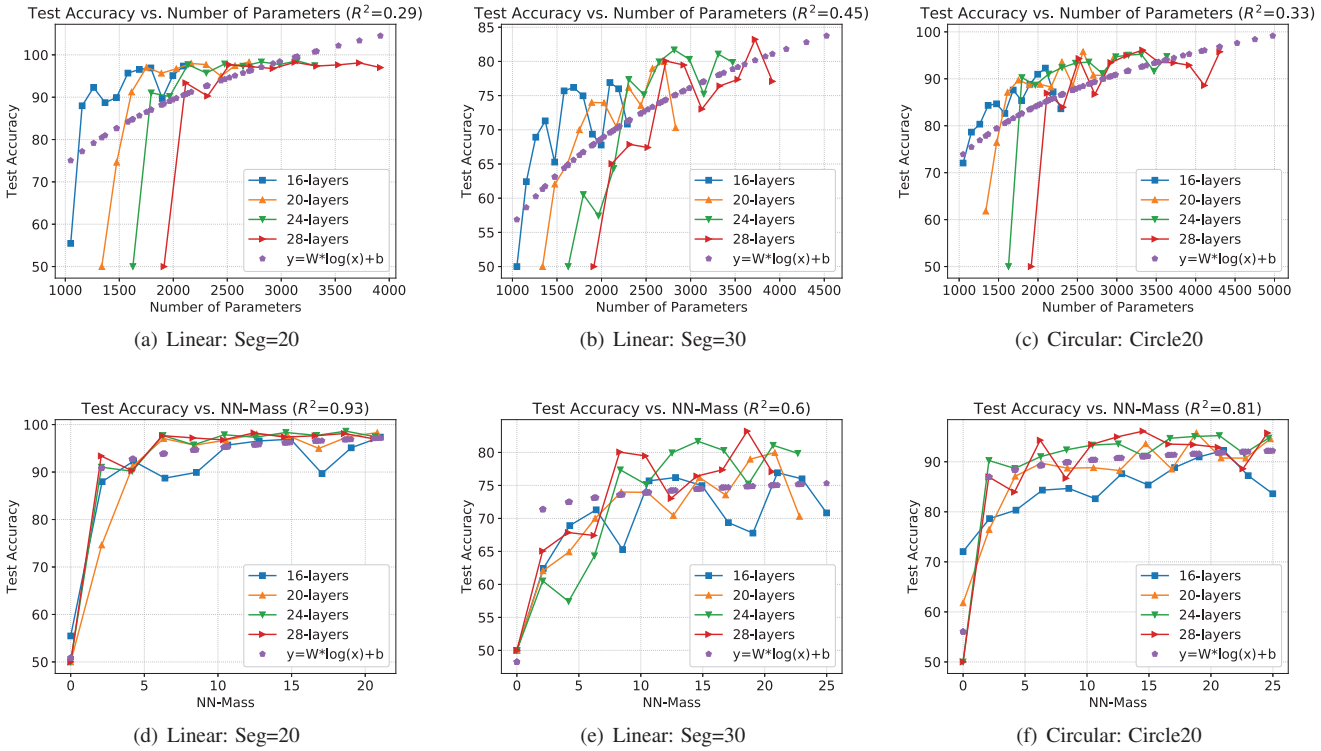
### H.3. Varying width multiplier on CIFAR-10

We now explore the impact of varying model width. In our DenseNet setup, we control the width of the models using *width multipliers* ($wm$)[8] [37, 6]. The above results are for $wm = 2$. For lower width CNNs ($wm = 1$), Fig. 14(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. 14(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. 14(c) shows the results for $wm = 3$. As evident, models with 6M-7M parameters achieve comparable test accuracy as models with up to 16M parameters (*e.g.*, bucket Y in Fig. 14(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

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

Table 6: 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: $Y_i$, $X_{1i} = L_i * cos(rand\_num)$, $X_{2i} = L_i * sin(rand\_num)$, $L_i = sample(\frac{1}{20}[\lfloor \frac{i}{20} \rfloor$, $Y_i = \lfloor \frac{i}{20} \rfloor mod 2$ |




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

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

Figure 11: 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.

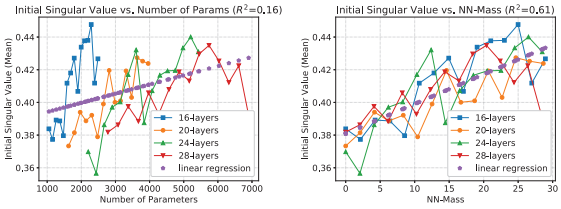



Figure 12: Synthetic results (Circle20 datasets): Mean singular value of $\boldsymbol{J_{i,i-1}}$ is much better correlated with NN-Mass than with #Params.

log(NN-Mass) achieves high values (0.74-0.90 as shown in Fig. 15 in Appendix H.4).

### H.4. $R^2$ of CIFAR-10 Accuracy vs. NN-Mass

Fig. 15 shows the impact of increasing model widths on $R^2$ of linear fit between test accuracy and log(NN-Mass).

### H.5. 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. 16(a), we fit a linear model between test accuracy and log(#Params) and found

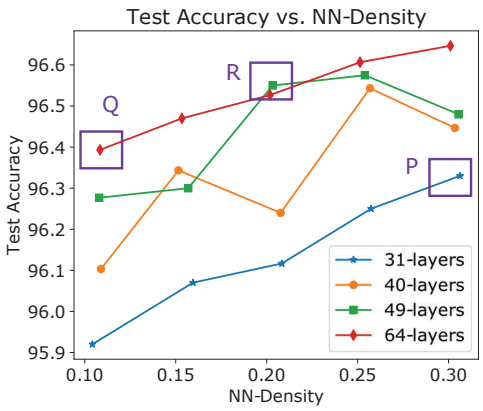



Figure 13: CIFAR-10 Width Multiplier $wm = 2$: Shallower models with higher density can reach comparable accuracy to deeper models with lower density. This does not help since models with different depths achieve comparable accuracies at different densities.

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. 16(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.6. Results for CIFAR-100

Results for CIFAR-100 dataset are shown in Fig. 17. As evident, several models achieve similar accuracy despite having highly different number of parameters (*e.g.*, see models within box W in Fig. 17(a)). Again, these models get clustered together when plotted against NN-Mass. Specifically, models within box W in Fig. 17(a) fall into buckets Y and Z in Fig. 17(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. 17(c)). Therefore, our observations hold true across multiple image classification datasets.

### H.7. Results for ImageNet (DenseNet setup)

For ImageNet, we create several DenseNet-type 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 trained these models on ImageNet dataset for 60 epochs. Fig. 18(a) shows the test accuracy of these CNNs vs. total #Params, while Fig. 18(b) shows the test accuracy vs. NN-Mass. As evident, although the model sizes are very different (e.g., model X is 3M parameters larger 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. 18(b)). Note that, the accuracies do *not* saturate (similar to other CIFAR-10 and CIFAR-100 results in Fig. 5, 14, 17 and Fig. 18(c,d) in next section): Cluster {Y,Z} achieves $4\%$ lower Top-1 accuracy (red points in Fig. 18(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. 18(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 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 the ImageNet dataset.

### H.8. Results for depthwise separable convolutions

Recent works are heavily influenced by depthwise separable convolutions (DSConv). To demonstrate that NN-Mass works with DSConv, we take our current DenseNet setup (*i.e.*, layer-by-layer convolutions with channels connected via random skip connections) and replace all convolutions with MobilenetV2 Expansion Blocks (1x1 conv $\rightarrow$ 3x3 DSConv $\rightarrow$ 1x1 conv) [28]. Random skip connections connect input channels across various Expansion Blocks. Fig. 18(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. 18(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.

### H.9. Results for Floating Point Operations (FLOPS)

All results for FLOPS (of CNN architectures in Tables 3, 4, and 5) 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.10. 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

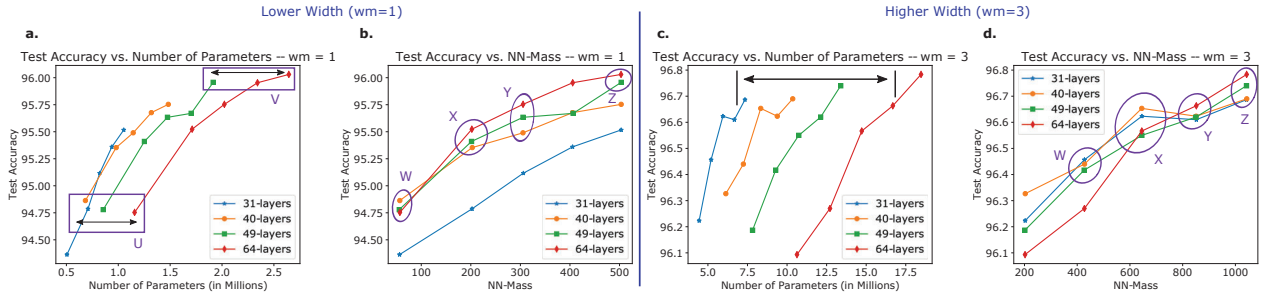



Figure 14: DenseNet-type CNNs for low- ($wm = 1$) and high-width ($wm = 3$) models: (a, b) Many models with very different #Params (boxes U and V) cluster into buckets W and Z (see also other buckets). (c, d) For high-width, we observe a significantly tighter clustering compared to the low-width case. Results are reported as the mean of three runs (std. dev. $\sim 0.1\%$).

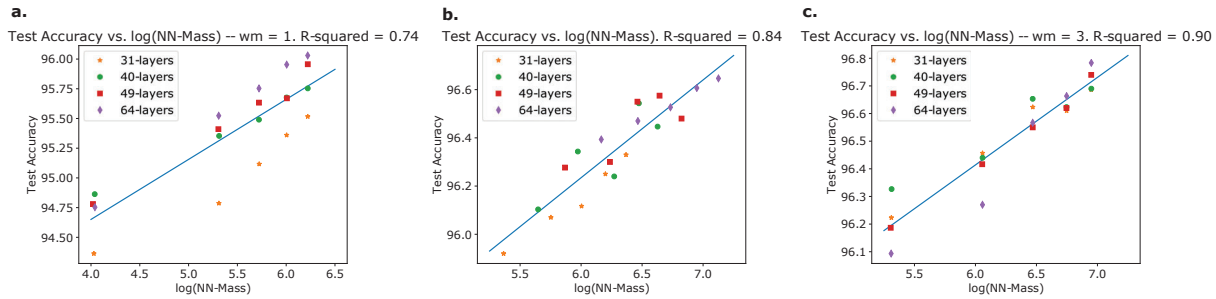



Figure 15: Impact of varying width of DenseNet-type CNNs: (a) Width multiplier, $wm = 1$, (b) $wm = 2$, and (c) $wm = 3$. As width increases, the capacity of small (shallower) models increases and, therefore, the accuracy-gap between models of different depths reduces. Hence, the $R^2$ for linear fit increases as width increases.

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 $m_S \approx m_L$, we only need to find how many skip connections to add in each cell. Since, NN-Mass has a closed form equation (*i.e.*, Eq. 2), a simple search over the number of skip connections 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 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.

**Directly designing compressed DenseNet-type CNNs for CIFAR-10.** We train our models for 600 epochs on the CIFAR-10 dataset (similar to the setup in DARTS [17]). Table 1 (main paper) 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 [40]. 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)!

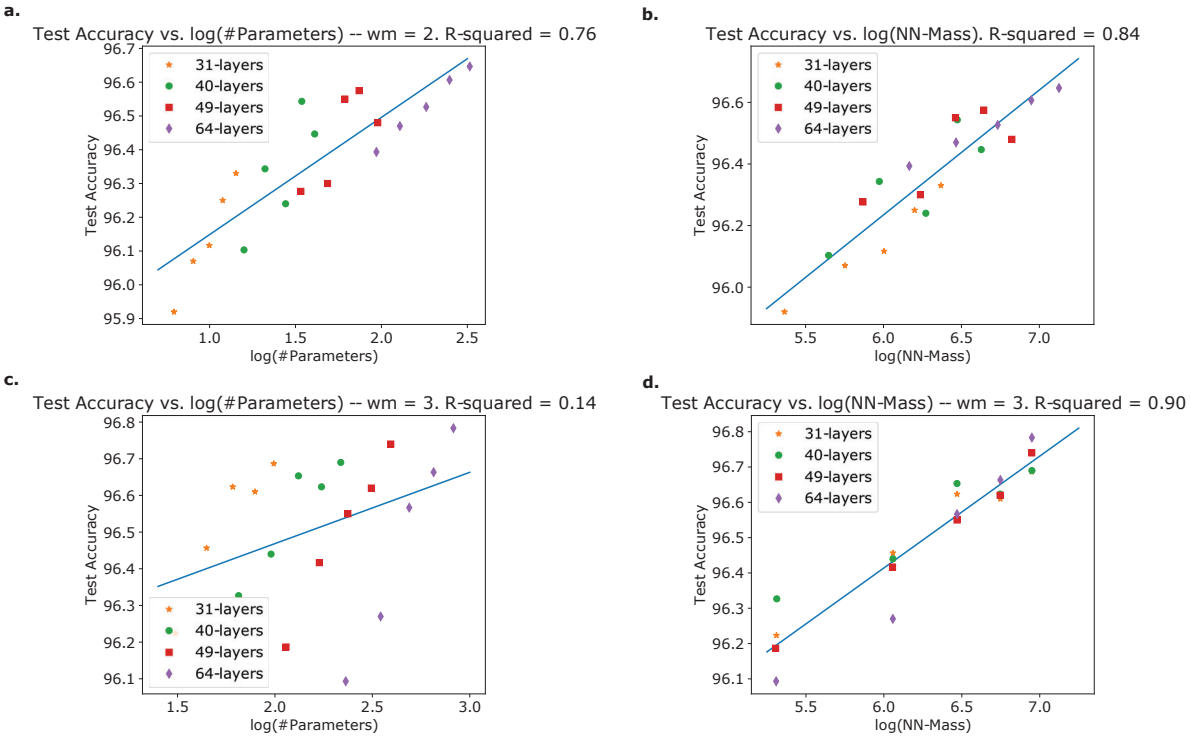



Figure 16: NN-Mass vs. parameter counting for DenseNet-type CNNs. (a) For $wm = 2$, log(#Params) fits the test accuracy with an $R^2 = 0.76$. (b) For the same $wm = 2$ case, log(NN-Mass) fits the test accuracy with a higher $R^2 = 0.84$. (c) For higher width ($wm = 3$), parameter counting completely 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 $R^2 = 0.9$.

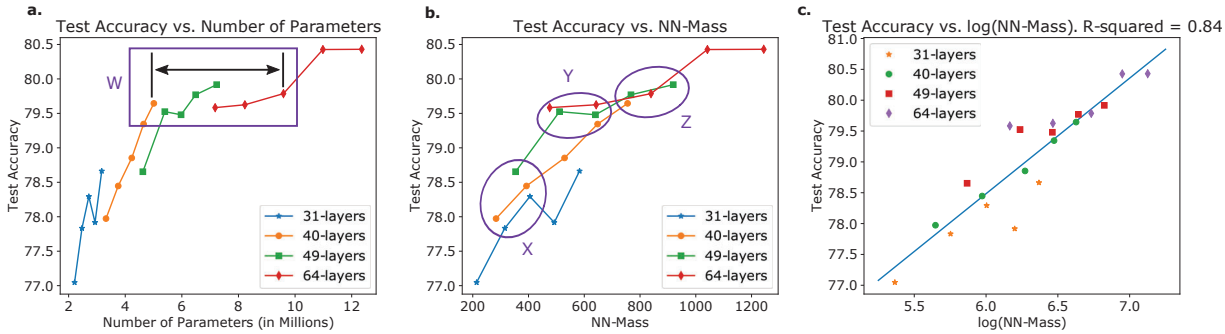



Figure 17: DenseNet-type CNNs: Similar results are obtained for CIFAR-100 ($wm = 2$). (a) Models in box W have highly different #Params but achieve similar accuracy. (b) These models get clustered into buckets Y and Z. (c) The $R^2$ value for fitting a linear regression model is $0.84$ which shows that NN-Mass is a good predictor of test accuracy. Results are reported as the mean of three runs (std. dev. $\sim 0.2\%$).

As evident from Table 1 (main paper), 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 [17], 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 [6], dilated convolutions [36], *etc*. On the contrary, we use regular convolutions with only concatenation-type skip connections in our work and present a theoretically grounded approach. Indeed, our current objective is not to beat DARTS (or any other technique), but rather underscore the topological properties that should guide the

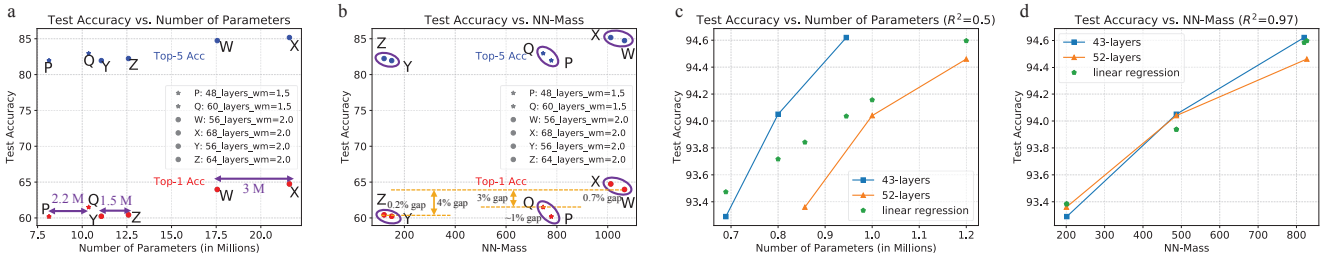



Figure 18: More results for DenseNet-type CNNs. (a,b) ImageNet: (a) Models {P,Q}, {W,X}, and {Y,Z} have very different #Params but similar test accuracy. (b) When plotted against NN-Mass, the models with similar NN-Mass and accuracy cluster together. (c,d) CIFAR-10 with DSConv: Again, models with similar NN-Mass achieve similar accuracy but have quite different #Params/layers.

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 14, 17, 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 1 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 CNNs, we fixed the other hyper-parameters as described above.

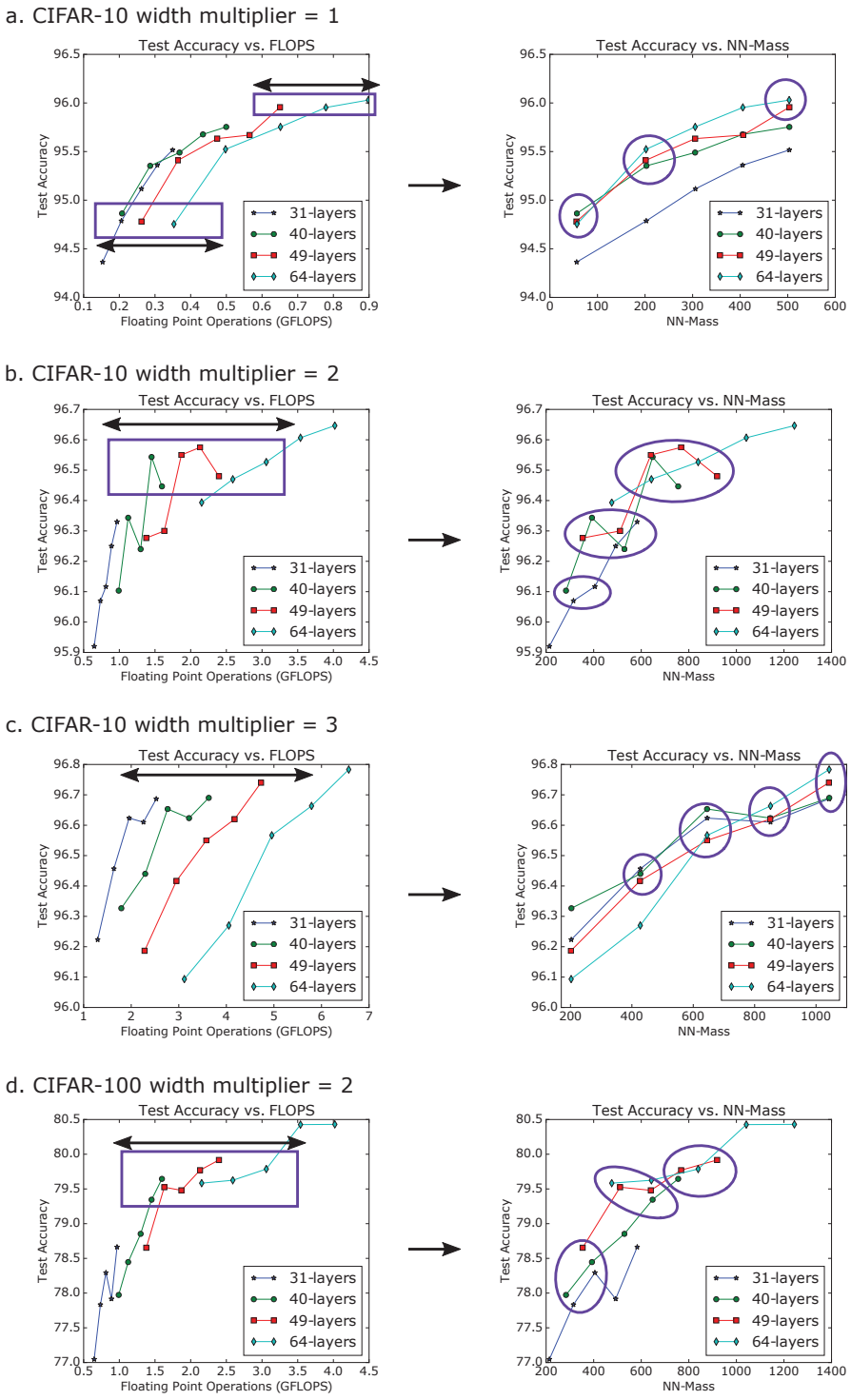



Figure 19: Models with highly different number of FLOPS achieve similar test accuracies. The CNN architectures are the same as those used in Figures 5, 14, and 17. The pattern for FLOPS is very similar to that for the number of parameters. Hence, these results show that models with both highly different number of parameters and FLOPS can achieve similar test accuracy. Again, these models cluster together when plotted against NN-Mass.