TOWARDS UNIFYING NEURAL ARCHITECTURE SPACE EXPLORATION AND GENERALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we address a fundamental research question of significant practical interest: Can certain theoretical characteristics of CNN *architectures* indicate *a priori* (*i.e.*, without training) which models with highly different number of parameters and layers achieve a similar generalization performance? To answer this question, we model CNNs from a network science perspective and introduce a new, theoretically-grounded, architecture-level metric called *NN-Mass*. We also integrate, for the first time, the PAC-Bayes theory of generalization with small-world networks to discover new synergies among our proposed NN-Mass metric, architecture characteristics, and model generalization. With experiments on real datasets such as CIFAR-10/100, we provide extensive empirical evidence for our theoretical findings. Finally, we exploit these new insights for model compression and achieve up to $3\times$ fewer parameters and FLOPS, while losing minimal accuracy (*e.g.*, 96.82% *vs.* 97%) over large CNNs on the CIFAR-10 dataset.

1 INTRODUCTION

Are there any theoretical characteristics of CNN *architectures* that can indicate *a priori* (*i.e.*, without training) which models achieve a similar test accuracy, despite having a vastly different number of parameters and layers? Even though there has been significant progress in architecture design practices (both manual [He et al. (2016); Huang et al. (2017); Howard et al. (2017)] as well as automated via Neural Architecture Search (NAS) [Zoph et al. (2018); Liu et al. (2018); Real et al. (2017)]), the above question remains unanswered, thereby making it one of the most fundamental problems in modern deep learning research. Clearly, answering this question can help us *directly design* efficient CNN architectures with predictable performance figures. More precisely, the above question is related to three important areas of research:

Neural Architecture Search (NAS) techniques automatically search for highly accurate and efficient models [Zoph et al. (2018); Liu et al. (2018); Real et al. (2017)]. The models designed by NAS algorithms usually surpass the manually designed architectures [Howard et al. (2017)].

Model Compression methods reduce the computational costs of existing deep networks without losing significant test accuracy. The existing model compression techniques mainly focus on pruning, quantization, and knowledge distillation [Li et al. (2016); Yang et al. (2016); Hubara et al. (2017); Lai et al. (2017); Hinton et al. (2015)].

Generalization of deep networks aims to theoretically understand why deep networks work well in practice by exploring properties of weight-norms/initializations, stability of deep networks to noise, optimization characteristics such as sharpness of the minima, *etc.* [Zhang et al. (2016); Arora et al. (2018); Neyshabur et al. (2015; 2017b;a)]. Deep networks achieve low generalization error despite having a large number of parameters. In contrast, traditional wisdom suggests that models should overfit when the number of parameters is much larger than the dataset size (which is true in practice).

Although there has been extensive research in the above three areas separately, to the best of our knowledge, there is no research at the *intersection* of all three directions. Specifically, while NAS can generate efficient architectures for mobile applications [Tan et al. (2019); Cai et al. (2018); Wu et al. (2019)], existing NAS research does not theoretically explain *why* the newly discovered architectures perform better than other models containing a similar number of parameters. Conversely, NAS also does *not theoretically* explain why can architectures with significantly different number of parameters and layers sometimes achieve similar accuracy. Moreover, a few generalization studies (*e.g.*, Arora et al. (2018)) use compression to prove generalization error bounds. However, the generalization studies do *not* provide any theory that can explicitly guide *efficient* architecture design.

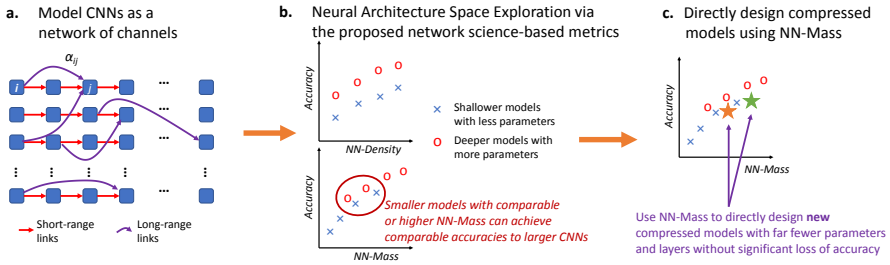

Figure 1: Approach Overview: (a) First, model a CNN as a network of channels. (b) Next, we propose NN-Mass and NN-Density, where NN-Mass is a theoretically-grounded metric that indicates generalization capability. (c) Exploit NN-Mass to directly design significantly compressed models.

In view of the above, in this paper, we explicitly unify the architecture space exploration and generalization. Towards this, we first model CNNs as complex networks [Newman et al. (2011)] since CNNs essentially consist of channels connected via filters at each convolutional layer (see Fig. 1(a)). Then, we define a new *NN-Density* metric to quantify how densely the channels of a CNN are connected to each other. We next use NN-Density to define a new, theoretically-grounded, architecture-level metric called *NN-Mass* which establishes a link between the structure of CNN architectures and their generalization error. Our objective is to use these architectural metrics for *Neural Architecture Space Exploration (NASE)*; throughout this paper, NASE refers to the process of studying the design space of deep networks via theoretically-grounded metrics such as NN-Mass.

To this end, we combine for the very first time, the Probably Approximately Correct (PAC)-Bayes theory of generalization [McAllester (1999a;b)] with network science [Newman & Watts (1999); Monasson (1999); Watts & Strogatz (1998a)] to theoretically prove that: (*i*) Architectures with higher NN-Mass achieve lower generalization error, and (*ii*) Models with similar NN-Mass lead to similar test accuracy, despite having different numbers of parameters and layers (see Fig. 1(b)). Then, we provide extensive empirical evidence to support our theory. Finally, given a large, highly accurate CNN, we show how NN-Mass can be used to *directly* design efficient models without compromising their accuracy. Of note, our proposed NN-Mass metric has a *closed-form equation* in terms of structure of the CNN architecture. Hence, it can be used to directly discover new, efficient models without training individual models (see Fig. 1(c)). Our approach is illustrated in Fig. 1.

Overall, we make the following **key contributions** to both theory and practice:

1. To the best of our knowledge, we are the first to exploit network science to *theoretically* study the generalization properties of CNN architectures. Towards this, we propose a new, architecture-level metric called NN-Mass which can indicate the generalization capabilities of CNNs with *long-range links*; we call concatenation-type skip connections (see Densenet [Huang et al. (2017)]) as long-range links or shortcut connections in this paper.
2. We are also the first to integrate the PAC-Bayes theory with network science to offer a new, principled method for studying properties of large deep networks. We discover a theoretical link between NN-Mass, a property of the CNN architectures, and generalization. We also show that models with similar NN-Mass achieve similar generalization errors.
3. To validate our new findings, we conduct extensive experiments with CNNs of different depths, parameters, and long-range links for CIFAR-10/100 datasets. We quantify the relationship between NN-Mass and generalization by demonstrating that the goodness-of-fit parameter (R^2) for a linear fit achieves high values (*e.g.*, between 0.74-0.90). We also show that NN-Mass can be used to predict the test accuracy of unknown architectures.
4. Finally, we demonstrate practical implications of our work by exploiting NN-Mass for model compression. Specifically, given a large, highly accurate CNN (*e.g.*, $\sim 97\%$ on CIFAR-10), we directly use our proposed NN-Mass metric to design new architectures that achieve accuracy close to that of the large model (*e.g.*, 96.82% on CIFAR-10 test set), while reducing the total parameters and FLOPS by more than $3\times$.

The rest of the paper is organized as follows: Section 2 covers related work on NAS and generalization, while Section 3 describes our proposed approach and the theoretical relationship between CNN architectures and their generalization. Next, Section 4 presents extensive experiments to support our theoretical results. Section 5 concludes the paper with final remarks on future work.

2 RELATED WORK

We now discuss related work on NAS, model compression and generalization. Prior art on long-range links for CNNs and network science is discussed in Appendix A.

Network Science-based NAS and Model Compression. Recently, standard network-generation techniques such as Barabasi-Albert (BA) [Barabási & Albert (1999)] or Watts-Strogatz (WS) [Watts & Strogatz (1998a)] models were used for NAS [Xie et al. (2019); Wortsman et al. (2019)]. However, like the rest of the NAS research, [Xie et al. (2019); Wortsman et al. (2019)] did *not* explore what theoretical characteristics of the architecture make models (with different number of parameters and layers) achieve similar generalization performance. Hence, to our knowledge, no theoretical attempt has been made to understand the link between architecture design and generalization.

Another prior work analyzes the impact of initialization on pruned networks via a lottery ticket hypothesis [Frankle & Carbin (2018)]. However, this work (and the rest of the model compression literature) does *not* explore the characteristics of the *architectures* that can indicate generalization.

Generalization of deep networks. The field of generalization has recently gained attention to understand why deep networks generalize without overfitting [Saxe et al. (2013); Nye & Saxe (2018); Li & Liang (2018); Brutzkus et al. (2017); Arora et al. (2018); Neyshabur et al. (2015; 2017b;a); Bartlett et al. (2017)]. However, these generalization studies either explore the properties of model weights (*e.g.*, weight-norms, noise stability, and other spectral properties), or attempt to understand the role of the optimization algorithm (*e.g.*, sharpness of minima, *etc.*); hence, generalization does *not* explicitly study what characteristics make good deep network *architectures*.

In contrast, our objective is twofold: (*i*) *Theoretically* understand architectural aspects of generalization, and (*ii*) *Practically* design new, efficient architectures without searching for them (*e.g.*, by directly exploiting our proposed metrics). Moreover, by integrating the PAC-Bayes theory with small-world networks, we present an effective way of modeling generalization of CNN architectures containing long-range links. We next describe our proposed approach that unifies these areas.

3 PROPOSED APPROACH

We first explain how CNNs can be modeled via network science. Next, we mathematically derive the proposed NN-Density and NN-Mass metrics that are needed for NASE. We then demonstrate the theoretical relationship between NN-Mass and generalization.

3.1 MODELING CNNS VIA NETWORK SCIENCE

We assume a generic CNN consisting of multiple cells, each containing a fixed number of convolutional layers, similar to existing works such as Densenets [Huang et al. (2017)], Resnets [He et al. (2016)], *etc*. As shown in Fig. 2(a), each cell can have a different width, *i.e.*, number of channels per layer. Following the standard practice [Simonyan & Zisserman (2014)], the width is increased by a factor of two at each cell as the feature map is reduced by half (see Fig. 2(a)).

We now illustrate a single convolution layer in Fig. 2(b) for our setup. In a standard CNN, a convolutional layer with n input channels and m output channels consists of m filters, each with $[k \times k \times n]$ dimensions. That is, the red kernel in Fig. 2(b) convolves with red input channel, green kernel convolves with green input channel, and so on. The outputs of all such channel-wise convolutions are added together to obtain a single output channel (violet output channel in Fig. 2(b)). In our setup, we explicitly assign *different* contributions from each input channel i to each output channel j as probabilities α_{ij}, which are fixed to random values[1]. These α_{ij} probabilities are directly used to define an adjacency matrix of any cell c: $\mathcal{A}^c_{ij} = \alpha_{ij}$, where $i, j \in \{0, 1, 2, \ldots, w_c \cdot d_c - 1\}$.

Next, we define the structure of our cells. Fig. 2(c) shows a cell with d_c convolution layers, and w_c channels per layer. Output channels at each layer i receive contributions from all output channels of layer $i-1$; we call these contributions *short-range links* since they connect consecutive layers (see red links in Fig. 2(c))[2]. In addition to short-range links, the output channels at layer i can also receive *long-range* contributions from *maximum* t_c channels present at layers $l \in \{0, 1, \ldots, i-2\}$ within the

[1]See Appendix B for more details on how these α_{ij} probabilities are assigned to channel connections.

[2]Not all links are shown in Fig. 2(c): All output channels at layer i will receive short-range links from the last layer, and additional long-range links from selected previous channels (see Fig. 8 (inset) in Appendix D).

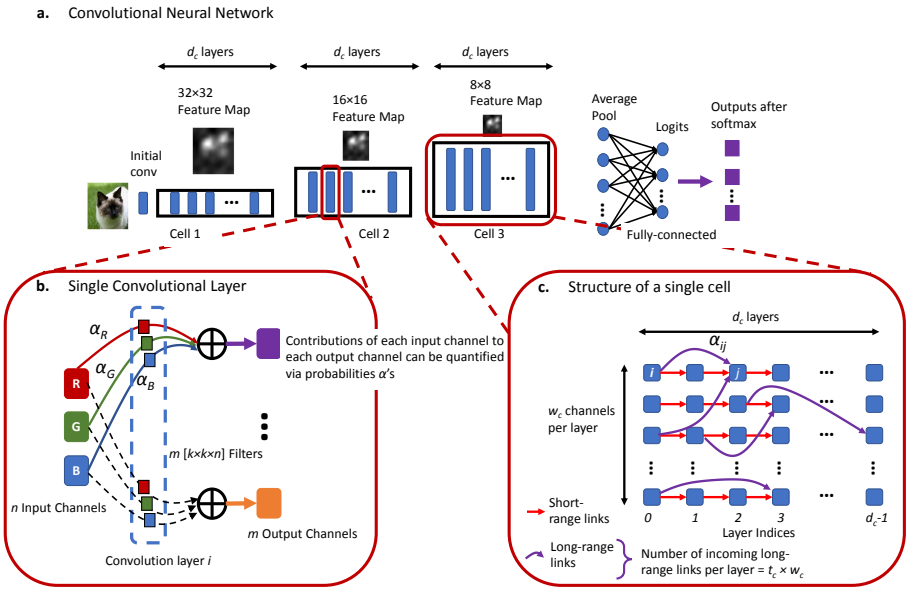

Figure 2: (a) CNN consisting of three cells of d_c layers each. (b) An input channel i contributes to a given output channel j with some fixed probability α_{ij}. (c) Each cell contains d_c layers with w_c channels per layer. Output channels at each layer get contributions from output channels of last layer and additional long-range links from previous layers (via concatenation). Not all links are shown. The contribution probabilities (α_{ij}'s) are used to define the weighted adjacency matrix of the CNN.

given cell (see purple links in Fig. 2(c)). That is, t_c determines maximum number of channels that can supply long-range links. By definition, a link is long-range if it connects channels across two or more layers. Hence, each layer receives long-range contributions from $\min\{w_c(i-1), t_c\}$ channels[3]. In practice, to create long-range links, the feature maps from previous layers are concatenated at the current convolution layer (like in Densenets [Huang et al. (2017)]). Of note, in our current setup, long-range links are confined only within the cell and do not extend across multiple cells.

To create long-range links at each layer i, we *randomly* select $\min\{w_c(i-1), t_c\}$ previous channels (out of all channels until layer $i-2$). Similar to recent NAS research [Li & Talwalkar (2019)], our rationale behind selecting random links (and random probabilities α_{ij}'s) is that random architectures are often as competitive as the carefully designed/searched models. Hence, throughout the paper, we look only at architectures with randomly chosen long-range links (after fixing the random seed).

3.2 Proposed Metrics for Neural Architecture Space Exploration

The above network formulation can be used to systematically study the architecture space for deep networks. Specifically, our problem has following objectives: (*i*) Theoretically quantify the architectural aspects of the generalization problem, and (*ii*) Exploit the above theory to *directly* design efficient CNN architectures in practice. To address the above goals, we propose two new network science-based metrics called NN-Mass and NN-Density, as defined below.

Definition 1 (Cell-Density)**.** *Given a CNN, density of a cell quantifies how densely its channels are connected via long-range links. Formally, for a given cell c, cell-density ρ_c is expressed as:*

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

For complete derivation, please refer to Appendix C. Next, we define NN-Density as the average density across all cells in a CNN. If a CNN has N_c cells, then the NN-Density ρ_{avg} is expressed as:

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

[3]When $t_c > w_c(i-1)$, total possible channels that can supply long-range links is limited to $w_c(i-1)$, *i.e.*, the total number of channels until layer $i-2$

Definition 2 (Mass of Deep Networks)**.** *We define NN-Mass to quantify the generalization capability of a CNN. Intuitively, for a given width (w_c), models with similar NN-Mass but different depth (d_c), long-range links (t_c), and number of parameters should achieve a similar test accuracy.*

Note that, density is basically mass divided by volume. Let volume be the total number of channels in a cell. Then, we can analogously derive the NN-Mass metric by multiplying the cell-density with total number of channels in each cell. Hence, the *NN-Mass* (m) is given as:

$$m = \sum_{c=1}^{N_c} w_c d_c \rho_c = \sum_{c=1}^{N_c} w_c d_c \frac{2\sum_{i=2}^{d_c-1} \min\{w_c(i-1), t_c\}}{w_c(d_c-1)(d_c-2)} = \sum_{c=1}^{N_c} \frac{2d_c \sum_{i=2}^{d_c-1} \min\{w_c(i-1), t_c\}}{(d_c-1)(d_c-2)} \tag{3}$$

Below, we explain the use of above metrics for Neural Architecture Space Exploration.

Neural Architecture Space Exploration (NASE). We define NASE as the process of studying the design space of CNNs via theoretically-grounded metrics such as NN-Mass. Note that, both NN-Density and NN-Mass relate network width, depth, and number of long-range links. For a fixed number of cells, a CNN architecture can be completely specified by $\{$depth per cell, width per cell, maximum long-range link candidates per cell$\}$ $=$ $\{d_c, w_c, t_c\}$. Hence, to perform NASE, we vary $\{d_c, w_c, t_c\}$ to obtain random architectures with varying NN-Density and NN-Mass values. Appendix D illustrates NN-Mass calculation for a given architecture using a concrete example. We next show the theoretical link between NN-Mass and generalization.

3.3 Provable Relationship between NN-Mass and Generalization

In this section, we theoretically prove the relationship between generalization and our proposed NN-Mass metric, a property of CNN architectures. We further show that models with similar NN-Mass values achieve similar test accuracy. To this end, we start with the PAC-Bayes theory which is used to bound the generalization error of a given (not necessarily a neural network-based) classifier. We also integrate the network theory with PAC-Bayes theory to derive our results.

Theorem 1 (McAllester Bound for Generalization Error (McAllester (1999a;b); Laviolette))**.** *Given any data generating distribution $\mathcal{D}$, any hypothesis class of predictors $\mathcal{H}$, any prior distribution P over the predictors, and any $\delta \in (0, 1]$, with probability at least $1-\delta$ and for all distributions Q over $\mathcal{H}$ for a randomly drawn training set $\mathcal{S}$ of N examples, the generalization bound is given by:*

$$\mathbb{E}_{\mathcal{D}}(L(f_Q)) \leq \mathbb{E}_{\mathcal{S}}(\hat{L}(f_Q)) + \sqrt{\frac{KL(Q||P) + \log\frac{2\sqrt{N}}{\delta}}{2N}},$$

where, f_Q is a classifier drawn from Q, $\mathbb{E}_{\mathcal{D}}(L(f_Q))$ denotes the expected error, $\mathbb{E}_{\mathcal{S}}(\hat{L}(f_Q))$ is the empirical error over the training set $\mathcal{S}$, and $KL(Q||P)$ denotes the Kullback-Leibler (KL) divergence between the distributions P and Q.

The above PAC-Bayes theorem bounds the generalization error of any classifier. Next, we prove generalization bounds for CNN architectures with long-range links in terms of NN-Mass. Without loss of generality, we assume in this section that the CNN architecture consists of a single cell containing d_c convolutional layers, each containing w_c channels per layer (*i.e.*, the width is w_c).

Theorem 2 (Generalization Bound for CNN Architectures in terms of NN-Mass)**.** *Consider single-cell CNN models with d_c layers and w_c channels per layer. Also, let t_c be the number of channels contributing long-range links. If Q denotes a probability distribution over CNNs with long-range links and an architecture $f_Q \sim Q$ has a NN-Mass of m, then, with probability at least $1-\delta$ the generalization error for this architecture is bounded as follows:*

$$\mathbb{E}_{\mathcal{D}}(L(f_Q)) \leq \mathbb{E}_{\mathcal{S}}(\hat{L}(f_Q)) + \sqrt{\frac{\frac{1}{2\sigma}(\frac{1}{6m} - \frac{1}{2}\log(\pi e m)) + \log\frac{2\sqrt{N}}{\delta}}{2N}},$$

where, σ is the maximum number of connections a channel can have in a given CNN architecture.

To prove the above theorem, we integrate, for the very first time, the PAC-Bayes theory of generalization [McAllester (1999a;b)] with the small-world theory from network science [Newman & Watts (1999); Monasson (1999)]. Essentially, we express the KL-divergence term in Theorem 1 as a function of NN-Mass. For complete proof, please refer to Appendix E.

Remark 1. The KL-divergence term (equation 14 in Appendix E) is a decreasing function of m. Hence, for a family of models with depth d_c and width w_c, Theorem 2 guarantees that the test error should reduce as NN-Mass increases. We show extensive empirical results for this observation.

As a natural consequence of Theorem 2, we next show that for two CNNs with a given width but different depths, models with similar NN-Mass values are expected to achieve similar generalization performance, even if the two models have vastly different number of parameters and layers!

Corollary 1 (Models with Similar Mass Achieve Similar Generalization Performance)**.** *Given two models of same width w_c, let f_Q^L be a deeper model with NN-Mass m_L, and let f_Q^S be a shallower model with NN-Mass m_S such that $m_L \leq m_S$. Then, the difference in expected test error of the two models is bounded with probability at least $1-\delta$ as follows:*

$$\mathbb{E}_{\mathcal{D}}(L(f_Q^L)) - \mathbb{E}_{\mathcal{D}}(L(f_Q^S)) \leq \mathbb{E}_{\mathcal{S}}(\hat{L}(f_Q^L)) - \mathbb{E}_{\mathcal{S}}(\hat{L}(f_Q^S)) + \sqrt{\frac{\frac{1}{2\sigma}\left[\frac{m_S - m_L}{6 m_L m_S} - \frac{1}{2}\log\frac{m_L}{m_S}\right] + \log\frac{4N}{\delta^2}}{2N}}$$

In other words, irrespective of number of parameters and layers, as the NN-Mass for the two models becomes similar, their test error is also expected to become similar.

The corollary follows directly from the KL-divergence term in Theorem 2 (see equation 14 in Appendix). See Appendix F for the complete proof of the closed-form bound above.

Remark 2. For the same width, a shallower and a deeper model with same NN-Mass intuitively implies that the shallower model is more densely connected than the deeper model (see equation 3). Hence, this might suggest that NN-Density can also help us identify models that achieve similar test accuracy. However, as we shall show empirically, NN-Density alone *cannot* be used to identify such models since the CNNs with different depths achieve similar test accuracy for different NN-Density values. In contrast, NN-Mass can identify CNNs that yield similar generalization performance *a priori* (*i.e.*, without training) since it has a closed-form equation 3 in terms of $\{d_c, w_c, t_c\}$.

Intuition behind why NN-Mass indicates generalization performance of CNN architectures. While proving Theorem 2, we show that a CNN with long-range links can be seen as a superposition of a lattice network and a random network (see Fig. 9 in Appendix E). Also, for $d_c >> 2$ (true for deep CNNs), the average degree (*i.e.*, the average number of connections for nodes) of the random network $\bar{k}_{\mathcal{R}|\mathcal{G}} = m/2$ (see equation 9 in Appendix E), and that of the lattice network is just w_c. Hence, the average degree of the overall CNN is $w_c + m/2$, which is independent of the depth d_c. Since average degree indicates how well-connected the network is, 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). As a result, we hypothesize that these topological properties might constrain the amount of information being learned by CNNs.

Next, we present detailed experimental evidence to support our theoretical findings.

4 Experimental Setup and Results

4.1 Experimental Setup

Our objective is to perform NASE by varying $\{d_c, w_c, t_c\}$ that generates random architectures with different NN-Mass and NN-Density values. Similar to prior art like Huang et al. (2017), we keep the total number of cells $= 3$ for all experiments. Overall, we conduct the following types of experiments for NASE on CIFAR-10 and CIFAR-100 datasets: (*i*) We show the impact of varying NN-Density (Remark 2). (*ii*) By providing extensive empirical evidence towards Theorem 2 and Corollary 1, we next show that NN-Mass can identify models that achieve similar test accuracy. (*iii*) We further show that our findings hold across models with different widths. (*iv*) We then demonstrate that NN-Mass is better for predicting generalization performance than *parameter counting*, a baseline used to indicate model generalization. (*v*) We predict test accuracy of completely unknown architectures. (*vi*) We also show that our findings hold for CIFAR-100 which is significantly more complex than CIFAR-10. All experiments are repeated three times with different random seeds.

Finally, to demonstrate the practical implications of our work, we exploit NN-Mass to directly design efficient CNNs which achieve accuracy comparable to larger networks with similar NN-Mass. More details for the experimental setup (*e.g.*, architecture details, learning rates, data augmentation, *etc.*) can be found in Appendix G (see Table 2). Next, we describe the results for the above experiments.

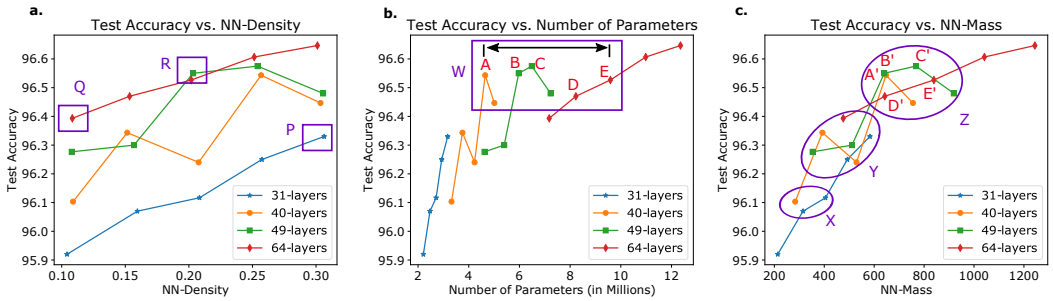

Figure 3: CIFAR-10 Width Multiplier $wm = 2$: (a) Shallower models with higher density can reach comparable accuracy to deeper models with lower density. This does not help since models with different depths achieve comparable accuracies at different densities. (b) Models with very different #parameters (box W) achieve similar test accuracies. (c) Models with similar accuracy often have similar NN-Mass: Models in W get clustered into Z. Results are reported as mean of three runs.

4.2 Results

Unless stated otherwise, the results are for CIFAR-10 (CIFAR-100 is presented towards the end).

4.2.1 NASE: NN-Mass as an Indicator of Generalization of CNN Architectures

Impact of Varying NN-Density. We train different deep networks with varying NN-Density (see Table 2 models in Appendix G). Fig. 3(a) shows that shallower models with higher density can reach accuracy comparable to deeper models with lower density (which is quite reasonable; see Remark 2). However, NN-Density alone does *not* help us identify a *family of models* that yields similar generalization despite having significantly different number of parameters/layers. Specifically, while we can say that for a given width, a shallower model might outperform a deeper model provided it is connected densely enough, NN-Density does not specify how dense the connections must be. Moreover, models with different depths achieve comparable test accuracies for 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. 3(a)). Therefore, NN-Density alone *cannot* be used to identify CNNs with similar generalization performance. Hence, we next focus on NN-Mass.

Impact of Varying NN-Mass on Generalization. Fig. 3(b) shows the test accuracy of the trained models *vs.* total parameters. As evident, some 40-layer models with 5M parameters (*e.g.*, model A in Fig. 3(b) within box W) perform comparably to several 64-layer models with more than 8M parameters (models D,E in Fig. 3(b)). Hence, models with highly different numbers of parameters and layers can achieve comparable accuracies (see all models within box W in Fig. 3(b)).

To explain this, we show test accuracy *vs.* NN-Mass in Fig. 3(c). Two observations are worth noting:

1. **The higher the NN-Mass, the higher the test accuracy.** This observation reinforces Remark 1 and Theorem 2 that higher NN-Mass should result in lower generalization error.
2. **Irrespective of number of parameters/layers, models with similar NN-Mass achieve similar accuracy (Corollary 1).** All models within box W (models A-E in Fig. 3(b)) cluster into bucket Z (models A'-E' in Fig. 3(c)). Same holds for models within X and Y.

These observations emphasize that the NN-Mass is an important indicator of generalization performance, and is able to identify a *family of models* that obtains similar test accuracy.

The above results are for width multiplier, $wm = 2$, and vary NN-Mass indirectly due to changing NN-Densities. Since it is clear that NN-Mass is highly correlated with generalization, we now directly vary NN-Mass for models with $wm \in \{1, 3\}$ to ensure that the above observations hold true for CNNs with different widths. More specifically, in Fig. 4(a), we observe that for $wm = 1$, models in boxes U and V have significantly different number of parameters and, yet, they achieve a similar test accuracy. Again, when plotted against NN-Mass (see Fig. 4(b)), models within the boxes U and V in Fig. 4(a) concentrate into buckets W and Z, respectively (see also other buckets).

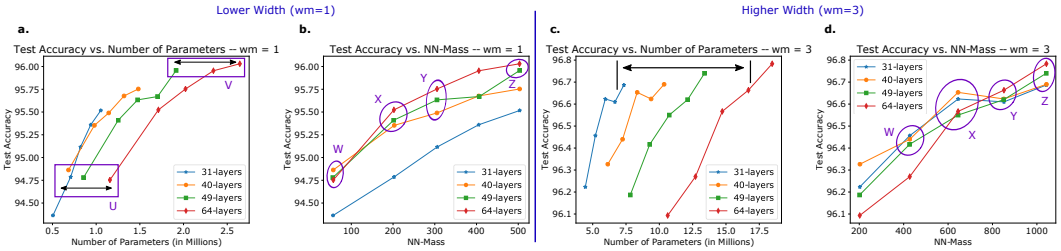

Figure 4: Similar observations hold for low- ($wm = 1$) and high-width ($wm = 3$) models: (a, b) Many models with very different #parameters (boxes U and V) get clustered into buckets W and Z (see also other buckets). (c, d) For high-width, we observe significantly tighter clustering compared to the low-width case. Results are reported as mean of three runs.

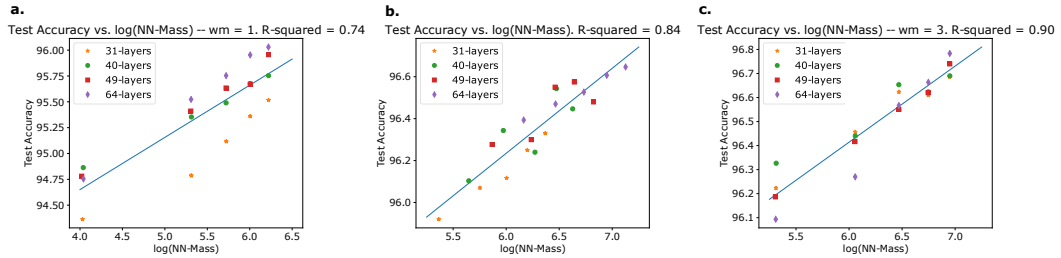

Figure 5: Impact of varying width: (a) Width multiplier, $wm = 1$, (b) $wm = 2$, and (c) $wm = 3$. As width increases, capacity of small (shallower) models increases and, therefore, the accuracy-gap between models of different depths reduces. Hence, the R^2 for linear fit increases as width increases.

Note that, for $wm = 1$, the 31-layer models do not fall within the buckets (see blue line in Fig. 4(b)). We hypothesize that this could be because of the following tradeoff. Specifically, since Corollary 1 states that the difference in test errors is bounded by the sum of (*i*) difference in training errors, and (*ii*) difference between NN-Mass values, the former term might dominate for low-capacity models (and, thus, the difference in test errors would increase). For instance, the training accuracy of 31-layer models is found to be much lower (*e.g.*, 0.66%-0.9%) than that of 64-layer models. In contrast, 40-layer models have only 0.27%-0.4% lower training error than the 64-layer CNNs. Hence, this suggests that a tradeoff between training accuracy difference and NN-Mass values should affect the difference in test accuracies of various architectures. We next show that as the width multiplier increases further, the shallower models perform much more similar to deeper models.

Fig. 4(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. 4(d) contains models ranging from $\{$31 layers, 6.7M parameters$\}$, all the way to $\{$64 layers, 16.7M parameters$\}$). In general, we observe that as the width increases, the capacity of the CNNs increases and, hence, the curves on Accuracy *vs.* NN-Mass plot come closer to each other. We next quantify the above observation by fitting a linear model to predict Accuracy using log(NN-Mass). As evident from Fig. 5, the goodness-of-fit (R^2) increases from 0.74 to 0.84 to 0.90, as the width increases. This demonstrates that as the width of the CNN increases, NN-Mass becomes a better indicator of generalization.

Comparison between NN-Mass and Parameter Counting. Direct parameter counting is often used as a baseline for comparison in many generalization studies. In Appendix H.1 (Fig 10), we show that NN-Mass significantly outperforms parameter counting as an indicator of generalization for CNNs. Specifically, for low-width models ($wm = 2$), a linear model between test accuracy and log(#parameters) yields an $R^2 = 0.76$ (compared to $R^2 = 0.84$ for NN-Mass, see Fig. 10(a, b)); this indicates that parameter counting is a decent predictor of generalization for low-width models. However, for high-width models ($wm = 3$), parameter counting cannot predict generalization performance at all. More precisely, the parameter count achieves an $R^2 = 0.14$ for $wm = 3$ (Fig. 10(c)). On the other hand, NN-Mass achieves a significantly higher $R^2 = 0.90$ (see Fig. 10(d)).

We note that our work cannot be compared against generalization indicators presented in Arora et al. (2018); Neyshabur et al. (2015; 2017b;a); Bartlett et al. (2017), because these works do *not* consider architectural aspects of the generalization problem and do *not* deal explicitly with CNN architectures with long-range links. The objective of this prior art is to understand how optimization properties

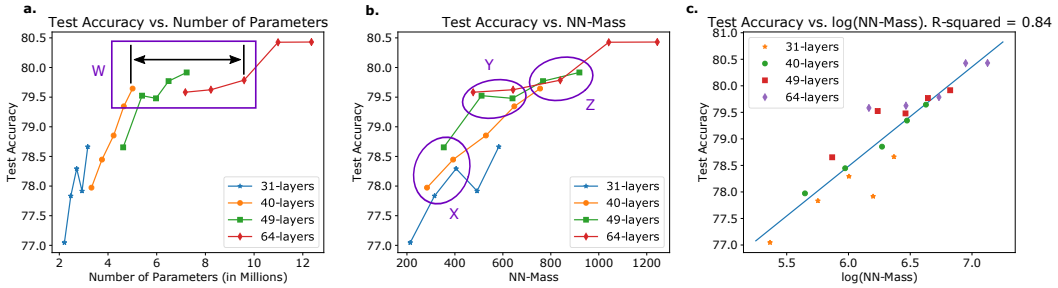

Figure 6: Similar results are obtained for the CIFAR-100 dataset ($wm = 2$). (a) Models in box W have significantly different #parameters but achieve similar accuracy. (b) These models get clustered into buckets Y and Z. (c) The R^2 value for fitting a linear regression model is 0.84 which shows that NN-Mass is a good predictor of test accuracy. Results are reported as mean of three runs.

(*e.g.*, sharpness of minima), noise-stability, weight-norms, *etc.*, affect generalization. Hence, the prior research does not explicitly provide any insights into the architecture itself. In contrast, our problem is to explicitly understand the impact of CNN *architectures* on generalization.

Exploiting NN-Mass to Predict Test Accuracy of Unknown Architectures. Since NN-Mass is a good indicator of generalization performance, we next use it to predict test accuracies of completely unknown architectures. Specifically, for $wm = 2$, we train the linear model shown in Fig. 5(b) on the initial set of $\{31, 40, 49, 64\}$-layer models, and use this linear model to predict the test accuracy of the unknown $\{28, 43, 52, 58\}$-layer CNNs with different NN-Densities. The complete details of this experiment and the results are presented in Appendix H.2. We show that a linear model trained on CNNs of depth $\{31, 40, 49, 64\}$ ($R^2 = 0.84$; see Fig. 5(b)) can successfully predict the test accuracy of unknown CNNs of depth $\{28, 43, 52, 58\}$ with a high $R^2 = 0.79$ (see Fig. 11 in Appendix H.2).

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 shown in Table 2 (Appendix G) from scratch on CIFAR-100. Fig. 6(a) shows the test accuracy of various models as a function of number of parameters. As evident, several models achieve similar accuracy despite having highly different number of parameters (*e.g.*, see models within box W in Fig. 6(a)). Again, these models get clustered together when plotted against NN-Mass. Specifically, models within box W in Fig. 6(a) fall into buckets Y and Z in Fig. 6(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. 6(c)). Therefore, our observations hold true across multiple image classification datasets.

To summarize, we show that (*i*) As NN-Mass increases, the test error of CNNs reduces, and (*ii*) NN-Mass can identify models that yield similar test accuracy, despite having very different #parameters/layers. We next use the latter observation to directly design efficient architectures.

4.2.2 Case Study: Directly Designing Compressed Models with NN-Mass

We now directly exploit the NN-Mass for model compression. Appendix H.3 explains how compressed models can be designed via NN-Mass. Following the setup in recent NAS works like DARTS [Liu et al. (2018)], we train our models for 600 epochs and report their test accuracy.

Table 1 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 compressed 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 compressed 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 compressed CNNs. Indeed, NAS involves an initial "search-phase" over a space of operations to find the architectures [Zoph et al. (2018)]. Similarly, model compression techniques like pruning [Li et al. (2016)] and quantization [Hubara et al. (2017)] also involve some kind of finetuning. In contrast, our models can be directly found using the closed

Table 1: 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 reported from Liu et al. (2018) which uses a similar setup for training the final model discovered after the search.

Model	Architecture design method	#Parameters/ #FLOPS	Number of layers/ cells/long-range links (t_c)	Specialized search space?	NN-Mass	Test Accuracy
DARTS (first order)	NAS [Liu et al. (2018)]	3.3M/–	–/20 cells/–	Yes	–	$97.00 \pm 0.14\%$
DARTS (second order)	NAS [Liu et al. (2018)]	3.3M/–	–/20 cells/–	Yes	–	$\mathbf{97.24 \pm 0.09}\%$
Train large models to be compressed	Manual	11.89M/3.63G	64/3 cells/ [90, 170, 300]	No	1126	$97.02 \pm 0.06\%$
	Manual	8.15M/2.54G	64/3 cells/ [50,100,150]	No	622	$96.99 \pm 0.07\%$
Proposed	Directly via NN-Mass	**5.02M/1.59G**	40/3 cells/ [60,130,170]	No	755	$\mathbf{97.00 \pm 0.06}\%$
Proposed	Directly via NN-Mass	**4.69M/1.51G**	37/3 cells/ [70,140,180]	No	813	$\mathbf{96.93 \pm 0.10}\%$
Proposed	Directly via NN-Mass	**3.82M/1.2G**	31/3 cells/ [70,140,200]	No	856	$\mathbf{96.82 \pm 0.05}\%$

form equation 3 of NN-Mass (see Appendix H.3), which does not involve any intermediate training/finetuning or even an initial search-phase like prior NAS methods. Therefore, NN-Mass can indicate the generalization properties of various architectures *a priori* (*i.e.*, without any training)!

As evident from Table 1, our 5M parameter model reaches a test accuracy of 97.00%, while the 4.6M (3.82M) parameter model[4] 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 total parameters by up to $3\times$ compared to the 11.89M parameter model. Moreover, the improvement in number of FLOPS is also up to $3\times$. Hence, NN-Mass results in a new model compression method which operates at architecture-level and does not rely on pruning/quantization.

Finally, Table 1 shows CIFAR-10 results for DARTS [Liu et al. (2018)], a competitive NAS baseline. As shown, with slightly lower 3.3M parameters, the first order DARTS achieves comparable accuracy to our proposed NN-Mass-based compressed models. Moreover, DARTS second order achieves a slightly higher accuracy ($\sim 0.2\%$ higher). However, it should be noted that 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)], dialated convolutions [Yu & Koltun (2015)], *etc*. In contrast, we use the regular convolutions with only concatenation-type long-range links in our work and present a theoretically-grounded approach. We show that it is *not* just the specialized convolutions that result in models that attain high accuracy with less parameters, but also *certain CNN architectures* are inherently more accurate. Therefore, NN-Mass can be used to design efficient architectures even when the search space is limited to regular convolutions.

5 Conclusion and Future Work

In this paper, we have proposed a new, theoretically-grounded, architecture-level metric called *NN-Mass* that can indicate the generalization properties of CNN architectures *a priori* (*i.e.*, without training). By integrating PAC-Bayes and small-world network science for the very first time, we have also theoretically proved two key properties of NN-Mass: (*i*) For a given depth and width, the higher the NN-Mass, the lower the generalization error, and (*ii*) Irrespective of total number of parameters, models with similar NN-Mass yield similar test accuracy. We have further presented extensive empirical evidence for the above theoretical findings by conducting experiments on real datasets such as CIFAR-10/100. Finally, we have used these new insights to directly design compressed models which reduce parameters/FLOPS by up to $3\times$, while losing minimal accuracy compared to the large CNN (*e.g.*, 96.82% test accuracy *vs.* $\sim 97\%$ for large CNN on CIFAR-10 dataset).

The present work opens several new directions in deep network generalization with implications for architecture search and model compression. As a future work, we plan to extend, both in theory and practice, NN-Mass for networks with branches and depth-wise separable convolutions. We further plan to bridge the theory between architectural aspects of generalization presented in this work *vs.* the generalization ideas discussed in prior art (*e.g.*, weight-norms, *etc.*).

[4]Unlike the 5M parameter model in Table 1 (which we partially explored by training it for 200 epochs in the last section during NASE), we never trained these 31- and 37-layer compressed models before. In fact, we never trained any other 37-layer models at all throughout this work. Hence, these models are completely new!

REFERENCES

Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. *arXiv preprint arXiv:1802.05296*, 2018.

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.

Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pp. 6240–6249, 2017.

Alon Brutzkus, Amir Globerson, Eran Malach, and Shai Shalev-Shwartz. Sgd learns over-parameterized networks that provably generalize on linearly separable data. *arXiv preprint arXiv:1710.10174*, 2017.

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

Ronald J Evans and J Boersma. The entropy of a Poisson distribution (C. Robert Appledorn). *SIAM Review*, 30(2):314–317, 1988.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

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.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

V. E. Hoggatt, Chih yi Wang, R. T. Hood, J. L. Brown, and C. H. Cunkle. E1366: Two related triangles. *The American Mathematical Monthly*, 67(1):82–84, 1960. ISSN 00029890, 19300972. URL http://www.jstor.org/stable/2308942.

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.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *JMLR*, 18 (1):6869–6898, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Liangzhen Lai, Naveen Suda, and Vikas Chandra. Deep convolutional neural network inference with floating-point weights and fixed-point activations. *arXiv preprint arXiv:1703.03073*, 2017.

Francois Laviolette. A tutorial on pac-bayesian theory. *NeurIPS 2017 Tutorial (NeurIPS 2017 Workshop on (Almost) 50 shades of Bayesian Learning: PAC-Bayesian trends and insights)*. URL https://bit.ly/2ySQPsU.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv:1608.08710*, 2016.

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

Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*, pp. 8157–8166, 2018.

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

David A McAllester. Some PAC-Bayesian Theorems. *Machine Learning*, 37(3):355–363, 1999a.

David A McAllester. PAC-Bayesian model averaging. In *COLT*, volume 99, pp. 164–170. Citeseer, 1999b.

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.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pp. 1376–1401, 2015.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pp. 5947–5956, 2017a.

Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1707.09564*, 2017b.

Maxwell Nye and Andrew Saxe. Are efficient deep representations learnable? *arXiv preprint arXiv:1807.06399*, 2018.

Umit Y Ogras and Radu Marculescu. " it's a small world after all": Noc performance optimization via long-range link insertion. *IEEE Transactions on very large scale integration (VLSI) systems*, 14(7):693–706, 2006.

Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2902–2911. JMLR. org, 2017.

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.

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

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

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

Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10734–10742, 2019.

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

Tien-Ju Yang, Yu-Hsin Chen, and Vivienne Sze. Designing energy-efficient convolutional neural networks using energy-aware pruning. *arXiv:1611.05128*, 2016.

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.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 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 Long-range links in CNNs and Network Science

Many recent innovations in deep learning architecture design have resulted in state-of-the-art deep networks that achieve excellent classification accuracy on complex vision and natural language applications. One of the most important innovations is the concept of *shortcut connections* in deep networks which enable complex architectures and have pushed the accuracy far beyond the traditional CNNs. For instance, Resnets [He et al. (2016)] introduced residual blocks which add feature maps at alternate convolution layers. Similarly Densenets [Huang et al. (2017)] have dense blocks that contain all-to-all connections. In contrast to Resnets, Densenets do not add feature maps but instead concatenate them together. The core idea in both of these models is to improve the information flow through the deep networks with the help of such shortcut connections between various layers.

Indeed, shortcut connections have long been a subject of study in the field of network science, *e.g.*, small world networks [Watts & Strogatz (1998b)] that specifically deal with networks with long-range and short-range links (*e.g.*, in social, biological, transportation networks [Newman et al. (2011)], and even multicore networks [Ogras & Marculescu (2006)]). In this paper, we view the shortcut connections in deep networks as being analogous to the long-range links in generic networks such as social and transportation networks. Hence, network science can be a good choice for studying the dynamics of long-range links in deep networks. Since Densenets [Huang et al. (2017)] have been shown to achieve higher accuracy than Resnets, we focus on concatenation-type long-range links.

B Computing network weights for CNN channel connections

Note that, in a standard CNN, a convolutional layer with n input channels and m output channel consists of m filters, each with $[k \times k \times n]$ dimensions. That is, as shown in Fig. 2(b), red kernel in the filter convolves with red input channel, green kernel convolves with green input channel, and so on. The output of all such *channel-wise convolutions* are added together to obtain a *single* output channel (*e.g.*, violet output channel in Fig. 2(b)). However, this implicitly assumes that all input

channels contribute *equally* to all output channels. Clearly, this assumption is counter-intuitive since our overall objective in an image classification problem is to separate out the identifying features of various classes at the final convolution layer (*i.e.*, output channels at the final layer must activate for *different* features). Therefore, adding the outputs of channel-wise convolutions at each intermediate layer can make the training process of CNN inherently hard.

To alleviate the above problem, we explicitly assign *different* contributions from each input channel i to each output channel j as probabilities α_{ij}. Specifically, for each output channel, we create a vector $\boldsymbol{q_j} = \{q_{1j}, q_{2j}, \ldots, q_{nj}\}$ with Kaiming-normal initialization [He et al. (2015)], where each element q_{ij} of this vector denotes an unnormalized contribution from an input channel i to the given output channel j. Then, to generate the probabilities α_{ij}, we simply compute the softmax of $\boldsymbol{q_j}$. The probabilities thus obtained are used as contributions from input channels to this output channel (*e.g.*, $\{\alpha_R, \alpha_G, \alpha_B\}$ in Fig. 2(b)). We call the unnormalized weight vector $\boldsymbol{q_j}$ as *contribution weights*, and the probabilities α_{ij}'s as *contribution probabilities* throughout this paper. Hence, in contrast to a standard convolution where individual channel-wise convolutions are directly added to obtain one output channel (*i.e.*, in a traditional CNN, $\alpha_{ij} = 1$ for all connections), the final convolution in our proposed model is computed as a weighted sum of channel-wise convolutions (where, weights are given by α_{ij}). Of note, throughout the training as well as inference, we keep these probabilities fixed.

Note that, *all* channel contributions (both long-range and short-range) are quantified by probabilities α_{ij}'s. Specifically, instead of defining the contribution weight vectors ($\boldsymbol{q_i}$) for each output channel, we can directly initialize a contribution weight matrix $Q = [\boldsymbol{q_1}^T, \boldsymbol{q_2}^T, \ldots, \boldsymbol{q_n}^T]$ for all channels in a cell. Then, the contribution probabilities α_{ij}'s are obtained by taking column-wise softmax of Q. Next, we show that fixing random probabilities, in fact, leads to better test accuracy than fixing constant contributions from input to output channels.

B.1 Impact of Input-to-Output Contributions

To evaluate the impact of various α_{ij}'s on CNN generalization, we train three separate models: (a) For the first model, we initialize the contribution weights for each layer to zeros and then take the softmax. Therefore, in this case, all contribution probabilities (α_{ij}'s) are constant (1/N, N being the number of input channels per layer). (b) In the next model, we directly initialize α_{ij}'s to all ones, which is the traditional CNN case where all channel-wise convolutions are directly added to obtain each output channel. The above two cases are *equal-contribution cases*. (c) Finally, to account for the proposed *unequal-contribution case*, we follow the process described in Appendix B above by fixing the contribution weights to Kaiming initialization [He et al. (2015)] and then take the softmax. Table 2 shows the architecture of CNNs used in this section: all models have 46-layers, $t_c = 200$, and width-multiplier of 2, which amounts to about 8M parameters. The models are trained for 350 epochs using stochastic gradient descent algorithm.

Fig. 7 demonstrates the training and test accuracy for the three models. As shown, the model with proposed unequal contributions (via random probabilities) achieves about 96.6% accuracy, which is about 1% higher than the corresponding equal-contributions cases. This is a significant result because achieving accuracy beyond 96% is very hard for CIFAR-10 dataset [Zoph & Le (2016); Huang et al. (2017); Zagoruyko & Komodakis (2016); Liu et al. (2018)]. Hence, this clearly demonstrates that not all input channels contribute equally to all output channels, and that even fixing the contributions to random probabilities significantly improves the generalization and convergence of the model. Note that, the α_{ij} contributions do not come at any additional cost in terms of number of parameters since these values are fixed and can be directly incorporated into convolution weights by element-wise multiplication. Therefore, this simple observation can be used to improve the accuracy of CNNs without any overhead in terms of number of parameters.

C Derivation of Density of a Cell

Note that, the maximum number of channels 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 channels up to layer $(i-2) = w_c(i-1)$ (see Fig. 2(c)). Indeed, if t_c is sufficiently large, initial few layers may not have t_c channels that can supply long-range links. For these layers, we use all available channels for

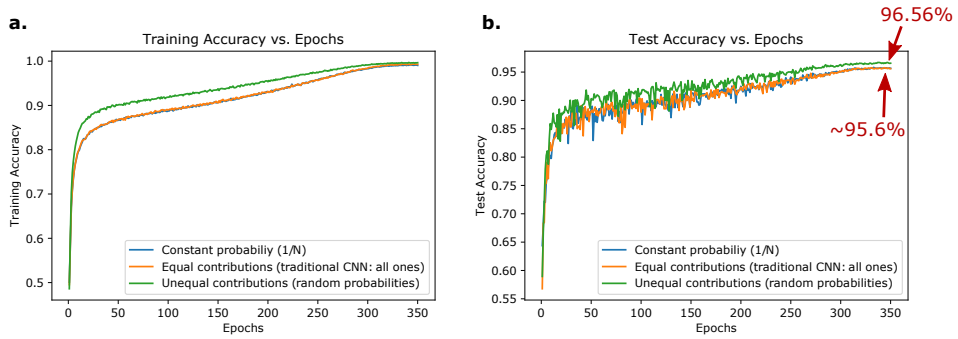

Figure 7: Not all input channels contribute equally to all output channels. (a) Training accuracy for three cases: (*i*) Set α_{ij}'s to constant probabilities by initializing the input-to-output contribution weights to zero at every layer and then taking the softmax. This results in constant probabilities as contributions from input to output channel at each layer (N is the number of input channels). (*ii*) A traditional CNN case where all channel-wise convolutions are directly added (*i.e.*, contributions from all inputs to all outputs (α_{ij}'s) are all ones). (*iii*) Channel-wise contributions are fixed to random probabilities, *i.e.*, α_{ij}'s are random. (b) Test accuracy for the above three cases demonstrates that the unequal contributions from input-to-output channels achieves significantly higher accuracy.

long-range links. Therefore, for a given layer i, number of long-range links (l_i) is given by:

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

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

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

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

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

Using equation 5 and equation 6, we can rewrite equation 1 as:

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

□

D EXAMPLE: COMPUTING NN-MASS AND NN-DENSITY FOR A CNN

Given a CNN architecture shown in Fig. 8, we now calculate its NN-Density and 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, first cell can have a maximum of three long-range link *candidates* per layer (*i.e.*, previous channels that can supply long-range links), second cell can have a maximum of four long-range link candidates per layer, and so on. Note that, the contribution probabilities (α_{ij}'s) are computed using the procedure in Appendix B (*i.e.*, we first generate contribution weights via Kaiming initialization, and then take the softmax to get probabilities). Moreover, as mentioned before, we randomly choose $\min\{w_c(i-1), t_c\}$ channels for long-range links at each layer. The inset

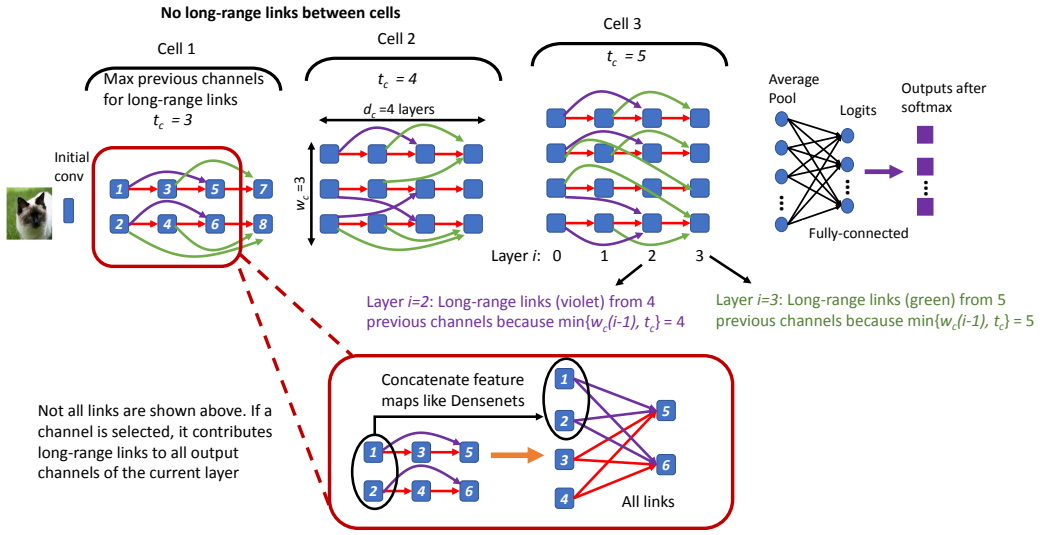

Figure 8: An example CNN to calculate NN-Density and NN-Mass. Not all links are shown in the main figure for simplicity. The inset shows the contribution from all long-range and short-range links: The feature maps for randomly selected channels are concatenated at the current layer (similar to Densenets [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.

of Fig. 8 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 equation 2 and equation 3 to compute the NN-Density and NN-Mass values. Putting the values in the equations, we obtain $\rho_{avg} = 0.78$ and $m = 28$. Consequently, the set $\{d_c, w_c, t_c\}$ can be used to specify the architecture of any CNN with concatenation-type long-range links. Therefore, to perform NASE, we vary $\{d_c, w_c, t_c\}$ to obtain architectures with different NN-Mass and NN-Density values.

E Proof of Theorem 2

Theorem 2 (Generalization Bound for CNN Architectures in terms of NN-Mass)**.** *Consider single-cell CNN models with d_c layers and w_c channels per layer. Also, let t_c be the number of channels contributing long-range links. If Q denotes a probability distribution over CNNs with long-range links and an architecture $f_Q \sim Q$ has a NN-Mass of m, then, with probability at least $1 - \delta$ the generalization error for this architecture is bounded as follows:*

$$\mathbb{E}_{\mathcal{D}}(L(f_Q)) \leq \mathbb{E}_{\mathcal{S}}(\hat{L}(f_Q)) + \sqrt{\frac{\frac{1}{2\sigma}(\frac{1}{6m} - \frac{1}{2}\log(\pi e m)) + \log\frac{2\sqrt{N}}{\delta}}{2N}},$$

where, σ is the maximum number of connections a channel can have in a given CNN architecture.

Proof. To prove Theorem 2, it only suffices to express the KL-divergence term in Theorem 1 as a function of m. As a result, we must model the distributions Q and P for various CNNs. Since we are dealing with the hypothesis class of CNN *architectures*, network science can be used to explicitly model the *distributions of the CNN topology*. Hence, in order to study the distributions over CNN architectural topology (*i.e.*, how various channels are connected together), network science concepts such as random networks, small-world networks, *etc.*, can provide valuable insights. Towards this, *the problem of computing distributions over CNN architectures can be equivalently seen as the problem of modeling connectivity in various network topologies*. Now, it has been established in network science literature [Newman et al. (2011)] that the connectivity of a network can be quantified using its degree distribution (recall that degree of a node in a network is given by total number of links connected to it). Hence, degree distribution can be used for modeling the distributions over various topologies of CNN architectures.

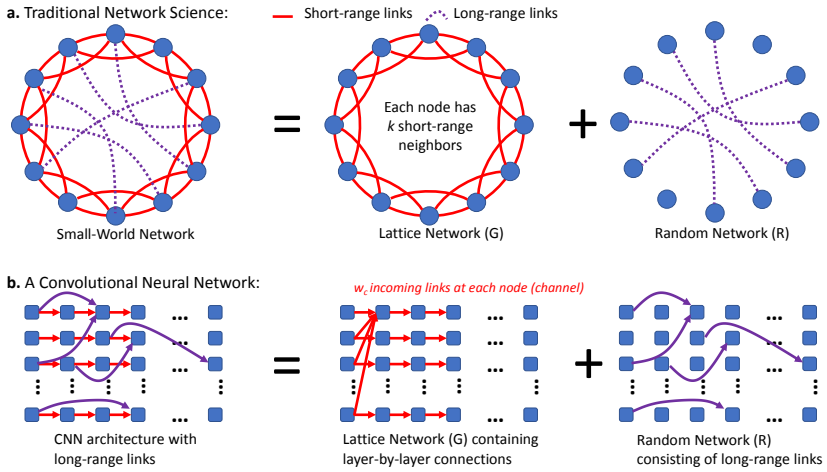

Figure 9: (a) Small-World Networks in traditional network science are modeled as a superposition of a lattice network ($\mathcal{G}$) and a random network $\mathcal{R}$ [Watts & Strogatz (1998a); Newman & Watts (1999); Monasson (1999)]. (b) A CNN with both short-range and long-range links can be similarly modeled as a random network superimposed on a lattice network. Not all links are shown for simplicity.

Starting from the above ideas, we assume *a priori* that our CNN architecture does *not* have any long-range links. In other words, our prior distribution P for CNN architectures consists only of models with short-range links and no shortcut connections[5]. Therefore, any architecture drawn from prior distribution P will look like a lattice network $\mathcal{G}$ with $w_c \times d_c$ total channels, and each channel at layer i is connected to w_c channels from the previous layer. Let this prior distribution P be given by a distribution $P(\mathcal{G})$ over lattice networks $\mathcal{G}$.

Next, we model the distribution Q for the proposed CNN architectures with long-range links. Note that, the CNNs considered in our work have both short-range and long-range links (see Fig. 2(c) and Fig. 8(inset)). 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. 9. Hence, the distribution over connectivities c in the small-world network can be written as:

$$Q \sim P(\mathcal{G}, \mathcal{R}) = P(\mathcal{G}) \cdot P(\mathcal{R}|\mathcal{G}) \tag{8}$$

Since $P(\mathcal{R}|\mathcal{G})$ represents the random long-range links created on top of the lattice network $\mathcal{G}$, the connectivity of long-range links due to $\mathcal{R}|\mathcal{G}$ follows a Poisson Distribution. This is because the degree distribution of random networks has been shown to be Poisson Distribution [Barabasi (2016)]. The λ parameter (*i.e.*, the mean) of this Poisson Distribution is given by the average degree of the random network. Therefore, the average degree for $\mathcal{R}$ superimposed on $\mathcal{G}$ is given by:

$$\begin{aligned}
\lambda = \bar{k}_{\mathcal{R}|\mathcal{G}} &= \frac{\textit{Number of long-range links added by } \mathcal{R}}{\textit{Number of nodes}} \\
&= \frac{w_c \sum_{i=2}^{d_c-1} \min\{w_c(i-1), t_c\}}{w_c d_c} \\
&= \frac{m(d_c-1)(d_c-2)}{2d_c^2} && \text{(using equation 3 for one cell)} \\
&\approx \frac{m}{2} && \text{(when } d_c >> 2\text{, } \textit{e.g.}\text{, for deep CNNs)}
\end{aligned} \tag{9}$$

[5]Note that, choosing a prior without any long-range links makes sense even from an "evolution of CNN architectures" perspective since, initially, CNN architectures such as AlexNet [Krizhevsky et al. (2012)] and VGG-16 [Simonyan & Zisserman (2014)] were developed (which do not have any shortcut connections), and complex architectures with long-range connections such as NASNET [Zoph et al. (2018)], DenseNets [Huang et al. (2017)], *etc.* were proposed only later.

Then, the probability $Q(c) = P(\mathcal{G}) \cdot P(\mathcal{R}|\mathcal{G}))$ of observing connectivity c in the small-world network can be written as follows:

$$\begin{aligned} Q(c) &= P(\mathcal{G}) \cdot \frac{\lambda^c e^{-\lambda}}{c!} \\ &= P(\mathcal{G}) \cdot \frac{(\bar{k}_{\mathcal{R}|\mathcal{G}})^c e^{-\bar{k}_{\mathcal{R}|\mathcal{G}}}}{c!}, \end{aligned} \tag{10}$$

where, $c \geq \bar{k}_{\mathcal{G}}$, the average degree of the lattice network $\mathcal{G}$. Since average degree is given by total number of links divided by number of nodes, $\bar{k}_{\mathcal{G}} = (w_c^2(d_c - 1))/(w_c d_c) = (w_c(d_c - 1))/d_c \approx w_c$ when $d_c >> 2$ (and implicitly $d_c >> 1$, which is true for deep CNNs). Now, let σ be the upper bound on the connectivity c. Also, we assume that the prior distribution P of drawing a lattice network is a uniform distribution for support $c \in \{\bar{k}_{\mathcal{G}} - \sigma + 1, \bar{k}_{\mathcal{G}} - \sigma + 2, \ldots, \bar{k}_{\mathcal{G}} + \sigma\}$. Then, the probability of observing a lattice network $\mathcal{G}$ with connectivity c is given by:

$$P(\mathcal{G}) = \begin{cases} \frac{1}{2\sigma} & \text{if } c \in \{\bar{k}_{\mathcal{G}} - \sigma + 1, \bar{k}_{\mathcal{G}} - \sigma + 2, \ldots, \bar{k}_{\mathcal{G}} + \sigma\} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Note that, since our proposed CNN architecture (drawn from distribution Q) is also based on a lattice network, the probability of drawing a lattice $P(\mathcal{G})$ is same between P (*i.e.*, the prior) and Q (*i.e.*, our hypothesis).

We next use the above equations to simplify the KL-divergence term in Theorem 1. For deep CNNs with $d_c >> 2$, we have

$$\begin{aligned} KL(Q||P) &= -\sum_{c=\bar{k}_{\mathcal{G}}}^{\sigma} Q(c) \log\left(\frac{P(c)}{Q(c)}\right) \\ &= -\sum_{c=\bar{k}_{\mathcal{G}}}^{\sigma} P(\mathcal{G}) \cdot P(\mathcal{R}|\mathcal{G}) \log\left(\frac{P(\mathcal{G})}{P(\mathcal{G}) \cdot P(\mathcal{R}|\mathcal{G})}\right) \\ &= \sum_{c=\bar{k}_{\mathcal{G}}}^{\sigma} P(\mathcal{G}) \cdot P(\mathcal{R}|\mathcal{G}) \log(P(\mathcal{R}|\mathcal{G})) \\ &= \sum_{c=\bar{k}_{\mathcal{G}}}^{\sigma} \frac{1}{2\sigma} \cdot P(\mathcal{R}|\mathcal{G}) \log(P(\mathcal{R}|\mathcal{G})) \\ &= \frac{1}{2\sigma} \sum_{c=\bar{k}_{\mathcal{G}}}^{\sigma} P(\mathcal{R}|\mathcal{G}) \log(P(\mathcal{R}|\mathcal{G})) \\ &= \frac{1}{2\sigma}(-\mathbb{H}_{\text{Poisson}(\bar{k}_{\mathcal{R}|\mathcal{G}})}), \end{aligned} \tag{12}$$

where, $\mathbb{H}_{\text{Poisson}(\bar{k}_{\mathcal{R}|\mathcal{G}})}$ is the Entropy of $\mathcal{R}|\mathcal{G}$ which follows a Poisson Distribution (equation 10). For a large λ, the Entropy of a Poisson Distribution can be approximated as [Evans & Boersma (1988)]:

$$\mathbb{H}_{\text{Poisson}(\lambda)} = \frac{1}{2}\log(2\pi e\lambda) - \frac{1}{12\lambda} + O(\frac{1}{\lambda^2}) \tag{13}$$

From equation 9, equation 13, and equation 12 and by ignoring the higher order terms of λ (since it is large), we get the following expression:

$$KL(Q||P) = \frac{1}{2\sigma}\left(\frac{1}{6m} - \frac{1}{2}\log(\pi e m)\right) \tag{14}$$

Putting the above expression into the bound in Theorem 1, we immediately get the generalization bound in terms of NN-Mass as shown in Theorem 2. □

F PROOF OF COROLLARY 1

Corollary 1 (Models with Similar Mass Achieve Similar Generalization Performance)**.** *Given two models of same width w_c, let f_Q^L be a deeper model with NN-Mass m_L, and let f_Q^S be a shallower*

model with NN-Mass m_S such that $m_L \leq m_S$. Then, the difference in expected test error of the two models is bounded with probability at least $1-\delta$ as follows:

$$\mathbb{E}_{\mathcal{D}}(L(f_Q^L)) - \mathbb{E}_{\mathcal{D}}(L(f_Q^S)) \leq \mathbb{E}_{\mathcal{S}}(\hat{L}(f_Q^L)) - \mathbb{E}_{\mathcal{S}}(\hat{L}(f_Q^S)) + \sqrt{\frac{\frac{1}{2\sigma}\left[\frac{m_S - m_L}{6m_L m_S} - \frac{1}{2}\log\frac{m_L}{m_S}\right] + \log\frac{4N}{\delta^2}}{2N}}$$

In other words, irrespective of number of parameters and layers, as the NN-Mass for the two models becomes similar, their test error is also expected to become similar.

Proof. We first compute the difference between the expected test errors for the deeper model f_Q^L and the shallower model f_Q^S. Then, according to Theorem 1, we have:

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}(L(f_Q^L)) - \mathbb{E}_{\mathcal{D}}(L(f_Q^S)) \leq \mathbb{E}_{\mathcal{S}}(\hat{L}(f_Q^L)) - \mathbb{E}_{\mathcal{S}}(\hat{L}(f_Q^S)) \\ + \sqrt{\frac{KL(Q_L||P) + \log\frac{2\sqrt{N}}{\delta}}{2N}} - \sqrt{\frac{KL(Q_S||P) + \log\frac{2\sqrt{N}}{\delta}}{2N}}\end{aligned} \tag{15}$$

To simplify the difference of square roots, we use the following lemma.

Lemma 1 (Two Related Triangles (Hoggatt et al. (1960))). *If three non-negative numbers a, b, and c satisfy the triangle inequality (i.e., $a+b \geq c$, $b+c \geq a$, and $a+c \geq b$), then the following also holds:*

$$|\sqrt{b} - \sqrt{c}| \leq \sqrt{a} \leq \sqrt{b} + \sqrt{c}$$

Proof for this lemma is given in Hoggatt et al. (1960).

Let $b = \frac{KL(Q_L||P)+\log\frac{2\sqrt{N}}{\delta}}{2N}$, and let $c = \frac{KL(Q_S||P)+\log\frac{2\sqrt{N}}{\delta}}{2N}$. Then, all we need to do is to find a such that the triangle inequality between a, b, and c is satisfied. Once we find such an a, the difference of square roots in equation 15 can be bounded by $\sqrt{a}$. Towards this, we begin by finding a lower bound for $b+c$:

$$\begin{aligned} b + c &= \frac{1}{2N}(KL(Q_L||P) + KL(Q_S||P)) + \frac{1}{2N}\log\frac{4N}{\delta^2} \\ &\geq \frac{1}{2N}|KL(Q_L||P) - KL(Q_S||P)| + \frac{1}{2N}\log\frac{4N}{\delta^2} \\ &= \frac{1}{2N}(KL(Q_L||P) - KL(Q_S||P)) + \frac{1}{2N}\log\frac{4N}{\delta^2} = a, \end{aligned} \tag{16}$$

where, the first inequality follows from the fact that the sum of two non-negative real numbers will always be greater than or equal to the absolute value of their difference. The second equality follows from equation 14. Specifically, since KL-divergence is a decreasing function of NN-Mass, and since $m_L \leq m_S$, it follows that $KL(Q_L||P) \geq KL(Q_S||P)$.

Now that we have a lower bound on $b+c$, we use this value as a. In order to use Lemma 1, we only need to make sure that the chosen values for a, b, and c satisfy the triangle inequality. Note that, by construction, $b+c \geq a$. So, we only need to prove that $a+c \geq b$ and $a+b \geq c$. For the former, it is easy to see that $a+c = b + \frac{1}{2N}\log\frac{4N}{\delta^2} \implies a+c \geq b$ (since $\frac{1}{2N}\log\frac{4N}{\delta^2}$ is non-negative as the size of training set N is a large number and $\delta \in (0,1]$). For the latter, we have:

$$\begin{aligned} a + b &= \frac{1}{2N}KL(Q_L||P) - \frac{1}{2N}KL(Q_S||P) + \frac{1}{2N}\log\frac{4N}{\delta^2} + \frac{1}{2N}KL(Q_L||P) + \frac{1}{2N}\log\frac{2\sqrt{N}}{\delta} \\ &= \frac{2}{2N}KL(Q_L||P) - \frac{1}{2N}KL(Q_S||P) \qquad \text{(add and subtract } KL(Q_S||P)/2N\text{)} \\ &\quad - \frac{1}{2N}KL(Q_S||P) + \underbrace{\frac{1}{2N}KL(Q_S||P) + \frac{1}{2N}\log\frac{2\sqrt{N}}{\delta}}_{c} + \frac{1}{2N}\log\frac{4N}{\delta^2} \\ &= c + \underbrace{\frac{2}{2N}(KL(Q_L||P) - KL(Q_S||P))}_{\geq 0 \text{ since } m_L \leq m_S \implies KL(Q_L||P) \geq KL(Q_S||P)} + \underbrace{\frac{1}{2N}\log\frac{4N}{\delta^2}}_{\geq 0} \geq c \end{aligned} \tag{17}$$

Hence, a, b, and c satisfy the triangle inequality. Therefore, we can use Lemma 1 to bound the difference of square roots in equation 15. Putting KL-divergence from equation 14 into a in equation 16, we get:

$$a = \frac{\frac{1}{2\sigma}\left[\frac{m_S - m_L}{6m_L m_S} - \frac{1}{2}\log\frac{m_L}{m_S}\right] + \log\frac{4N}{\delta^2}}{2N} \tag{18}$$

Finally, using Lemma 1 on equation 15 with the above a, we get the corollary statement. Clearly, the difference in expected error for the two models reduces as their NN-Mass values become similar. Therefore, irrespective of the number of parameters, if two models have similar NN-Mass, they are expected to yield similar test accuracies. We will present extensive empirical evidence to verify this corollary. □

G Complete Details of Experimental Setup

We first analyze the impact of non-uniform contributions from input channels to output channels at all layers (Appendix B). Then, we perform the Neural Architecture Space Exploration with NN-Mass and NN-Density. Specifically, we run many experiments with CIFAR-10 and CIFAR-100 datasets for CNNs containing different number of layers, parameters, NN-Mass and NN-Density:

1. **Impact of Input-to-Output Contributions:** We train three separate models to evaluate the impact of various α_{ij}'s on CNN generalization: Two models for constant contribution from all inputs to all outputs at each layer, and one model for the proposed unequal contributions via random probabilities. As shown in Table 2, for these experiments, we create a 46-layer model with width-multiplier[6] of 2, and a single cell (*i.e.*, any channel from any layer can contribute to any other channel[7]). Also, $t_c = 200$ long-range link candidates are selected randomly at each layer. We have already explained the results of this experiment in Appendix B.
2. **Neural Architecture Space Exploration and Correlation of NN-Mass with Generalization:** Next, we explore the architecture design space using our proposed NN-Mass and NN-Density metrics. We further demonstrate that NN-Mass is indeed correlated with generalization (Theorem 2) and can be used to identify models with similar accuracy despite having different number of parameters (Corollary 1). All models in this category are trained for 200 epochs. We conduct the following classes of experiments:
   - **Impact of Varying NN-Density.** We first systematically analyze how the test accuracy changes when average density (as defined in equation 2) is varied for CNNs of different depths. Specifically, we fix the width of all models and vary the depth (total depth of the CNN with three cells = $3d_c + 4$) and the maximum number of long-range link candidates (t_c). We choose t_c for different cells such that the density for all networks varies as: $\{0.1, 0.15, 0.2, 0.25, 0.3\}$. That is, different t_c values result in architectures with different NN-Mass and number of parameters and, hence, allows us to explore the architecture design space (see Example 1 for an illustration on how the set $\{d_c, w_c, t_c\}$ is used to define an architecture). Complete details of various t_c values for different networks is given in Table. 2.
   - **Impact of Varying NN-Mass and Width.** The above experiment indirectly changes NN-Mass (due to varying NN-Density). We next explicitly vary the NN-Mass across models of different widths to analyze how the relationship between test accuracy and NN-Mass changes across a large spectrum of architectures. For all experiments, we explicitly quantify the relationship between accuracy of CNNs *vs.* NN-Mass by fitting a linear regression model. We report the goodness-of-fit (or coefficient of determination, R^2) of the linear model.

[6]Base number of channels in each group is [16,32,64]. Hence, a width-multiplier of 2 implies that each group will have [32,64,128] channels per layer.

[7]Note that for evaluating the impact of α_{ij}'s, we have fixed the number of cells to one; as explained in the footnote above, this cell contains three separate groups of [32,64,128] channels per layer. We found that creating more cells improves the accuracy and, therefore, in the rest of the paper, we will use three cells for all models.

Table 2: Details of Experiments for varying α_{ij}'s and Average Densities

Experiment Type	Number of Cells	Max. Long-Range Link Candidates (t_c)	α_{ij}'s	Depth	Width Multiplier
Impact of α_{ij}'s	1	200	Constant(1/N)	46	2
	1	200	Ones (Traditional CNN)	46	2
	1	200	Random Probabilities	46	2
Impact of Average Density	3	[10,35,50] [20,45,75] [30,50,100] [40,60,120] [50,70,145]	Random Probabilities	31	2
Impact of Average Density	3	[20,40,70] [30,50,100] [40,80,125] [50,105,150] [60,130,170]	Random Probabilities	40	2
Impact of Average Density	3	[25,50,90] [35,80,125] [50,105,150] [70,130,170] [90,150,210]	Random Probabilities	49	2
Impact of Average Density	3	[30,80,117] [50,110,150] [70,140,200] [90,175,250] [110,215,300]	Random Probabilities	64	2

- **Comparison to Parameter Counting.** Parameter counting (*i.e.*, total number of trainable parameters in a model) has been a standard method to determine whether or not a given model will achieve high accuracy. Towards this, we demonstrate that NN-Mass is a significantly better metric than parameter counting for understanding generalization performance of various CNNs.
- **Predicting Test Accuracy of Unknown Architectures.** Finally, we train unknown models (*e.g.*, models with different depths, density, *etc.*) that were not trained as part of the initial Neural Architecture Space Exploration. The objective of this experiment is to determine if NN-Mass can be used to predict the test accuracy of models that were never trained before.

3. **Exploiting NN-Mass for Model Compression:** Finally, we *design* new models with NN-Mass comparable to (or slightly higher than) large state-of-the-art networks but significantly fewer layers and parameters. We train the large model and these newly designed, small models for 600 epochs and compare their test accuracies. This experiment is used to evaluate if NN-Mass can be used directly as measure for model compression.

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. Similar to setup in NAS prior works, 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.

H Additional Results

H.1 Comparison between NN-Mass and Parameter Counting

Direct parameter counting is often used as a baseline for comparison in the generalization literature [Arora et al. (2018)]. Here, we quantitatively demonstrate that while parameter counting can be a useful indicator of generalization for models with low width (but still not as good as NN-Mass), as the width increases, parameter counting cannot predict generalization. In contrast, we show that NN-Mass consistently predicts generalization performance with high accuracy. Specifically, in

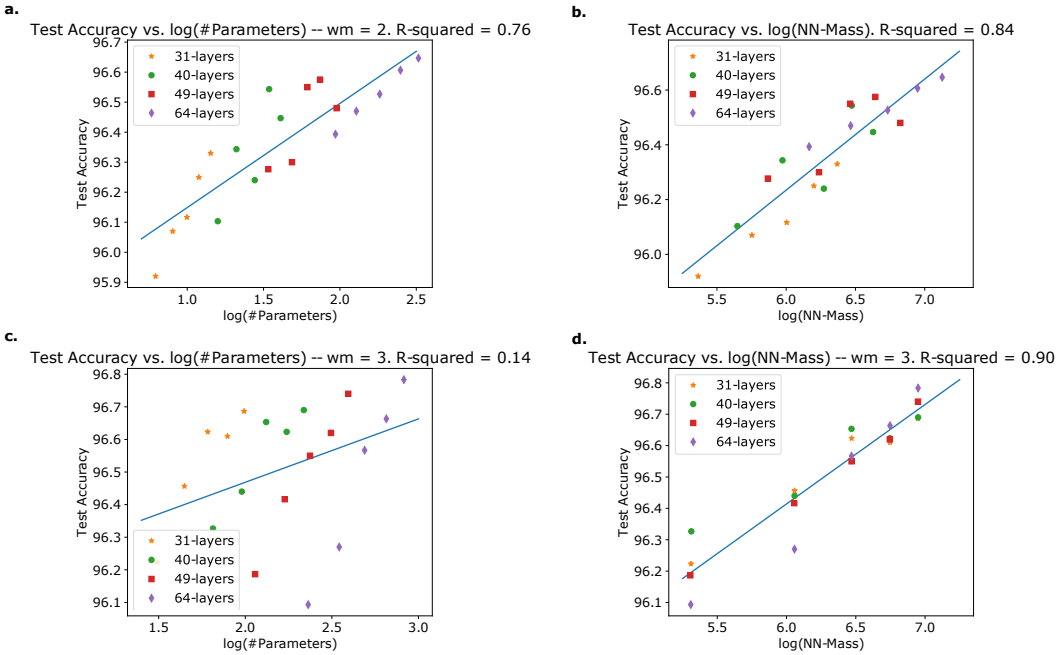

Figure 10: NN-Mass as an indicator of generalization performance compared to parameter counting. (a) For $wm = 2$, log(#parameters) fits the test accuracy with an $R^2 = 0.76$. (b) For the same $wm = 2$ case, log(NN-Mass) fits the test accuracy with a higher $R^2 = 0.84$. For lower width, parameter counting is a decent indicator of generalization performance. (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 $R^2 = 0.9$.

Fig. 10(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. 10(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 generalization than parameter counting for models with long-range links.

H.2 NN-Mass to Predict Test Accuracy of Unknown Architectures

We now demonstrate that NN-Mass can be used to predict the test accuracy of unknown architectures that have not been trained before. Towards this, we create a *testing set of new architectures* by training 20 previously unknown architectures with $wm = 2$, and $\{28, 43, 52, 58\}$ layers. For these models, we vary the NN-Density between $\{0.125, 0.175, 0.225, 0.275, 0.325\}$ which is different from the initial architecture space exploration setting in Fig 5(b) or Table 2 (in the initial setting, $\{31, 40, 49, 64\}$-layer models were trained for NN-Densities: $\{0.10, 0.15, 0.20, 0.25, 0.30\}$). We next use the linear model trained on the $\{31, 40, 49, 64\}$-layer models (see Fig. 5(b)) to predict the test accuracy of the unknown $\{28, 43, 52, 58\}$-layer CNNs. Note that, our testing set consists of models with both different number of layers and different NN-Densities (and, implicitly, different NN-Mass values) compared to the training set.

Fig. 11 shows that the testing $R^2 = 0.79$ (*i.e.*, the R^2 obtained by predicting the accuracy of models in the testing set) which is close to the training $R^2 = 0.84$ (see Fig. 5(b)). Hence, NN-Mass can be used to predict test accuracy of models which were never trained before.

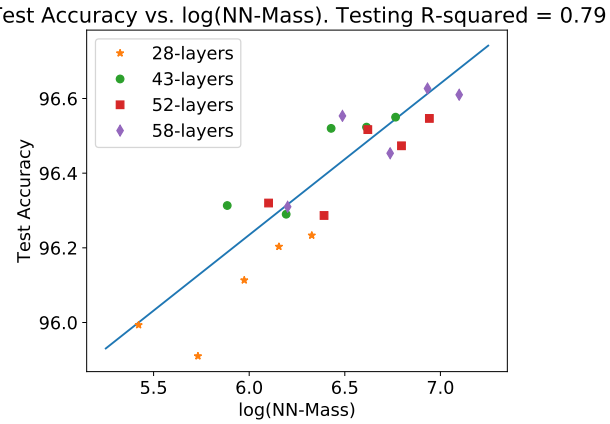

Figure 11: Linear modeled trained in Fig. 5(b) is used to predict the test accuracy of completely new architectures. The resulting $R^2 = 0.79$ is still high and is comparable to the training $R^2 = 0.84$. The linear model was trained on the test accuracies and NN-Mass of models with $\{31, 40, 49, 64\}$ layers, and densities varying as $\{0.10, 0.15, 0.20, 0.25, 0.30\}$. To create the testing set, we trained completely new models with $\{28, 43, 52, 58\}$ layers, and densities varying as $\{0.125, 0.175, 0.225, 0.275, 0.325\}$.

H.3 NN-MASS FOR DIRECTLY DESIGNING COMPRESSED ARCHITECTURES

Our theoretical and empirical evidence shows that NN-Mass is a reliable indicator for models which achieve similar accuracy despite having different number of layers and parameters. Therefore, this observation can be used for model compression 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 compressed model* using far fewer parameters and layers, but with a NN-Mass 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 a compressed 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 long-range links to add in each cell. Since, NN-Mass has a closed form equation (*i.e.*, equation 3), 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 compressed model is similar to that of the reference CNN, Corollary 1 suggests 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 newly compressed model and compare its test accuracy against that of the original large CNN.

Note that, our work is agnostic to what dataset is used since we rely solely on the properties of CNN architectures. That is, if we train the large CNN on a different dataset, our compressed model should still give accuracy close to that of the large CNN on the new dataset. Of course, the range of accuracy of different models will vary when the dataset is changed, but different architectures with similar NN-Mass should still yield a similar test accuracy. This observation is explicitly shown for CIFAR-10 and CIFAR-100 datasets in experiments (The "Results for CIFAR-100 Dataset" part in Section 4.2.1 shows that the same set of models that got clustered together for CIFAR-10 dataset, still form clusters for CIFAR-100 on the test accuracy *vs.* NN-Mass plot).