When NAS Meets Robustness:
In Search of Robust Architectures against Adversarial Attacks

Minghao Guo[1*]   Yuzhe Yang[2*]   Rui Xu[1]   Ziwei Liu[1]   Dahua Lin[1]

[1]The Chinese University of Hong Kong   [2]MIT CSAIL

Abstract

*Recent advances in adversarial attacks uncover the intrinsic vulnerability of modern deep neural networks. Since then, extensive efforts have been devoted to enhancing the robustness of deep networks via specialized learning algorithms and loss functions. In this work, we take an **architectural perspective** and investigate the patterns of network architectures that are resilient to adversarial attacks. To obtain the large number of networks needed for this study, we adopt one-shot neural architecture search, training a large network for once and then finetuning the sub-networks sampled therefrom. The sampled architectures together with the accuracies they achieve provide a rich basis for our study. Our "robust architecture Odyssey" reveals several valuable observations: 1) densely connected patterns result in improved robustness; 2) under computational budget, adding convolution operations to direct connection edge is effective; 3) flow of solution procedure (FSP) matrix is a good indicator of network robustness. Based on these observations, we discover a family of robust architectures (RobNets). On various datasets, including CIFAR, SVHN, Tiny-ImageNet, and ImageNet, RobNets exhibit superior robustness performance to other widely used architectures. Notably, RobNets substantially improve the robust accuracy (∼5% absolute gains) under both white-box and black-box attacks, even with fewer parameter numbers. Code is available at https://github.com/gmh14/RobNets.*

1. Introduction

Deep neural networks are shown to be vulnerable to adversarial attacks, where the natural data is perturbed with human-imperceptible, carefully crafted noises [9, 15, 32]. To mitigate this pitfall, extensive efforts have been devoted to adversarial defense mechanisms, where the main focus has been on specialized adversarial learning algorithms [9, 21], loss/regularization functions [13, 41], as well as image preprocessing [34, 28, 35, 38]. Yet, there is an orthogonal dimension that few studies have explored: **the intrinsic**

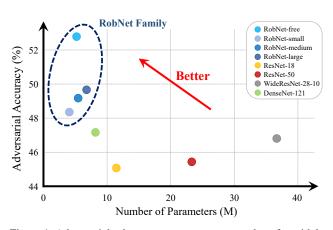*Equal contribution. Order determined by alphabetical order.

Figure 1. Adversarial robustness *vs.* parameter numbers for widely used architectures and the proposed RobNet family on CIFAR-10. All models are adversarially trained using PGD with 7 steps, and evaluated by PGD white-box attack with 100 steps. RobNets exhibit superior robustness performance to other architectures, even with fewer parameter numbers.

influence of network architecture on network resilience to adversarial perturbations. Although the importance of architectures in adversarial robustness has emerged in the experiments of several previous work [30, 36, 21], more comprehensive study on the role of network architectures in robustness remains needed.

In this work, we take the first step to systematically understand adversarial robustness of neural networks from an *architectural perspective*. Specifically, we aim to answer the following questions:

1. *What kind of network architecture patterns is crucial for adversarial robustness?*

2. *Given a budget of model capacity, how to allocate the parameters of the architecture to efficiently improve the network robustness?*

3. *What is the statistical indicator for robust network architectures?*

It is nontrivial to answer the above questions, since we need to train a massive number of networks with different

architectures and evaluate their robustness to gain insights, which, however, is exceedingly time-consuming, especially when adversarial training is used. Thanks to the method of one-shot neural architecture search (NAS), it becomes more accessible to evaluate robustness among a large number of architectures. Specifically, we first train a supernet for once, which subsumes a wide range of architectures as sub-networks, such as ResNet [11] and DenseNet [12]. Then we sample architectures from the supernet and finetune the candidate architectures for a few epochs to obtain their robust accuracy under adversarial attacks. We further conduct extensive analysis on the obtained architectures and have gained a number of insights to the above questions:

1) We present a statistical analysis on $1,000$ architectures from our cell-based search space, and discover a strong correlation between the density of the architecture and the adversarial accuracy. This indicates that densely connected pattern can significantly improve the network robustness.

2) We restrict the number of parameters under three different computational budgets, namely small, medium, and large. Our experimental results suggest that adding convolution operations to direct edges is more effective to improve model robustness, especially for small computational budgets.

3) We further release the cell-based constraint and produce studies on cell-free search space. For this setting, we find that the distance of flow of solution procedure matrix between clean data and adversarial data can be a good indicator of network robustness.

By adopting these observations, we search and design a family of robust architectures, called *RobNets*. Extensive experiments on popular benchmarks, including CIFAR [14], SVHN [23], Tiny-ImageNet [16], and ImageNet [7], indicate that RobNets achieve a remarkable performance over widely used architectures. Our studies advocate that future work on network robustness could concentrate more on the intrinsic effect of network architectures.

2. Related Work

Adversarial Attack and Defence. Deep neural networks (NNs) can be easily fooled by adversarial examples [9, 32, 20], where effective attacks are proposed such as FGSM [9], BIM [15], C&W [5], DeepFool [22], MI-FGSM [8], and PGD [21]. Extensive efforts have been proposed to enhance the robustness, including preprocessing techniques [3, 28, 38], feature denoising [35], regularization [41, 13, 17], adding unlabeled data [6, 29], model ensemble [33, 24], where adversarial training [9, 21] turns out to be the most effective and standard method for improving robustness. A few empirical attempts on robustness of existing network architectures have been made [30, 36], but no convincing guidelines or conclusions have yet been achieved.

Neural Architecture Search. Neural architecture search (NAS) aims to automatically design network architectures to replace conventional handcrafted ones. Representative techniques include reinforcement learning [44, 45, 43, 10, 1], evolution [27, 31] and surrogate model [18], which have been widely adopted to the search process. However, these methods usually incur very high computational cost. Other efforts [19, 26, 2] utilize the weight sharing mechanism to reduce the costs of evaluating each searched candidate architecture. Towards a fast and scalable search algorithm, our investigation here is based on the one-shot NAS [2, 4].

3. Robust Neural Architecture Search

3.1. Preliminary

In this section, we briefly introduce the concept of one-shot NAS and adversarial training for the ease of better understanding of our further analysis.

One-Shot NAS. The primary goal of NAS [45, 18, 19, 2] is to search for computation cells and use them as the basic building unit to construct a whole network. The architecture of each cell is a combination of operations chosen from a pre-defined operation space. In one-shot NAS, we construct a *supernet* to contain every possible architecture in the search space. We only need to train the supernet for once and then at evaluation time, we can obtain various architectures by selectively zero out the operations in the supernet. The weights of these architectures are directly inherited from the supernet, suggesting that weights are *shared* across models. The evaluated performance of these one-shot trained networks can be used to rank architectures in the search space, since there is a near-monotonic correlation between one-shot trained and stand-alone trained network accuracies. We refer readers to [2] for more details of this order preservation property.

In one-shot NAS, the one-shot trained networks are typically only used to rank architectures and the best-performing architecture is *retrained* from scratch after the search. In our work, however, we do not aim to get one *single* architecture with the highest robust accuracy, but to study the effect of different architectures in network robustness. Thus, we do not involve retraining stage in our method but fully utilize the property of accuracy order preservation in one-shot NAS.

Robustness to Adversarial Examples. Network robustness refers to how network is resistant to adversarial inputs. The problem of defending against bounded adversarial perturbations can be formulated as follows:

$$\min_{\theta} \ \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{x'\in S} \ \mathcal{L}\left(y, M(x';\theta)\right) \right], \quad (1)$$

where $S = \{x' : ||x - x'||_p < \epsilon\}$ defines the set of allowed perturbed inputs within l_p distance, M denotes the

(a) Search space of RobNet
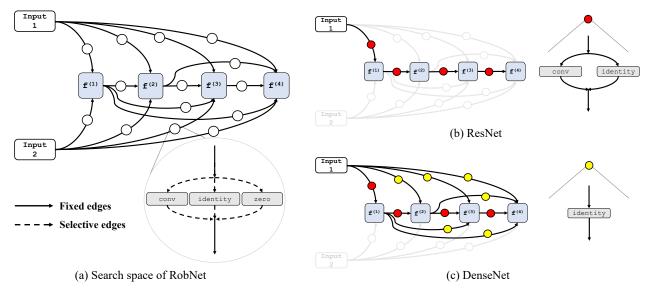
(b) ResNet

(c) DenseNet

Figure 2. Overview of the search space in robust architecture search: (a) The search space of RobNet. We only consider three candidate operations: 3×3 separable convolution, identity, and zero. We do not restrict the number of edges between two intermediate nodes to be one, which means that there could be multiple operations between two nodes. Such design benefits us to explore a larger space with more variants of network topology, including many typical human-designed architectures such as (b) ResNet and (c) DenseNet.

model and $\mathcal{D}$ denotes the data distribution. One promising way to improve network robustness is adversarial training. [21] proposed to use Projected Gradient Descent (PGD) to generate adversarial examples and augment data during training, which shows significant improvements of network robustness. In our study, we focus on adversarial attacks bounded by l_∞ norm and use PGD adversarial training to obtain robust networks of different architectures.

3.2. Robust Architecture Search Framework

We now describe the core components of our robust architecture search framework. Our work is based on conventional one-shot architecture search methods [2, 19], with certain modifications to facilitate adversarial training and further analysis. We introduce them in detail accordingly.

Search Space. Following [45, 18, 19, 2], we search for computation cell as the basic building unit to construct the whole network architecture. Each cell is represented as a directed acyclic graph $G = (V, E)$ consisting of N nodes. Each node $V^{(i)}$ corresponds to a intermediate feature map $f^{(i)}$. Each edge $E^{(i,j)}$ represents a transformation $o^{(i,j)}(\cdot)$ chosen from a pre-defined operation pool $\mathcal{O} = \{o_k(\cdot), k = 1, \ldots, n\}$ containing n candidate operations (see Fig. 2(a)). The intermediate node is computed based on all of its predecessors: $f^{(j)} = \sum_{i<j} o^{(i,j)}(f^{(i)})$. The overall inputs of the cell are the outputs of previous two cells and the output of the cell is obtained by applying concatenation to all the intermediate nodes. For the ease of notation, we introduce architecture parameter $\alpha = \{\alpha_k^{(i,j)} | \alpha_k^{(i,j)} \in \{0,1\}, i, j = 1, \ldots, N, k = 1, \ldots, n\}$ to represent can-

didate architectures in the search space. Each architecture corresponds to a specific architecture parameter α. For edge $E^{(i,j)}$ of an architecture, the transformation can then be represented as $o^{(i,j)}(\cdot) = \sum_{k=1}^{n} \alpha_k^{(i,j)} o_k(\cdot)$. We refer *direct edges* to those $E^{(i,j)} \in \{E^{(i,j)} | j = i + 1\}$ and refer *skip edges* to those $E^{(i,j)} \in \{E^{(i,j)} | j > i + 1\}$.

The main differences in search space between our work and conventional NAS lie in two aspects: 1) We shrink the total number of candidate operations in $\mathcal{O}$, remaining only: 3×3 separable convolution, identity, and zero. This helps to lift the burden of adversarial training, while remaining powerful candidate architectures in the search space [37]. 2) We do not restrict the maximal number of operations between two intermediate nodes to be one (i.e., $o^{(i,j)}(\cdot)$ could contain at most n operations). As shown in Fig. 2, such design encourages us to explore a larger space with more variants of network architectures, where some classical human-designed architectures can emerge such as ResNet and DenseNet.

Robust Search Algorithm. We develop our robust search algorithm based on the one-shot NAS approach [2]. Specifically, we set all the elements in architecture parameter α as 1 to obtain a supernet containing all possible architectures. During the training phase of the supernet, for each batch of training data, we randomly sample a candidate architecture from the supernet (by arbitrary setting some of the elements in α to 0). This path dropout technique is incorporated to decouple the co-adaptation of candidate architectures [2]. We then employ the min-max formulation in Eq. (1) to generate adversarial examples with respect to the sampled
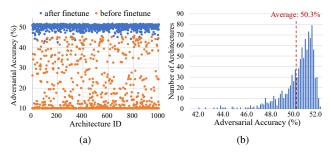
(a) (b)

Figure 3. Statistical results over 1,000 sampled architectures: (a) The robust accuracy improvement before and after finetuning. (b) Histogram of adversarial accuracy for all sampled architectures.
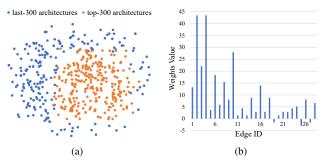

(a) (b)

Figure 4. Analysis of the selected top 300 (robust) architectures and last 300 (non-robust) architectures: (a) Visualization of t-SNE on α for all 600 architectures. The embedding of α is separable between robust and non-robust networks, which demonstrates the architecture has an influence on network robustness. (b) Values of weights of the trained linear classifier. We observe that almost all of the weight values are positive, indicating that there is a strong correlation between architecture density and adversarial accuracy.

sub-network, and perform adversarial training to minimize the adversarial loss. Such mechanism ensures adversarial examples generated during training are not specific to one architecture. We also provide the pseudo code of our robust search algorithm in the Appendix.

Robustness Evaluation. Once obtaining the supernet after robust training, we can collect candidate architectures by random sampling from the supernet and inheriting weights. Rather than direct evaluating the network on validation dataset as vanilla NAS methods, we find that finetuning the sampled network with adversarial training for only a few epochs can significantly improve the performance, which is also observed in [42]. The intuition behind finetuning is that while the training scheduler tries to inflate the robustness of each architecture, it yet needs to maintain the overall performance of *all* candidate architectures. The adversarial accuracy before and after finetuning for 1,000 randomly sampled candidate architectures is illustrated in Fig. 3(a). It can be clearly seen that the robustness performance has been largely improved.

After finetuning, we evaluate each candidate architecture on validation samples that are adversarially perturbed by the white-box PGD adversary. We regard the adversarial accuracy as the indicator of the network robustness.

3.3. Analysis of Cell-Based Architectures

Having set up the robust search framework, we would like to seek for answers for the first question raised in Sec. 3.1, that what kind of architecture patterns is crucial for adversarial robustness. We first conduct analysis of model robustness for cell-based architectures by following a typical setting in NAS methods [45, 19], where the architectures between different cells are shared.

Statistical Results. In cell-based setting, we adopt robust architecture search on CIFAR-10. We set the number of intermediate nodes for each cell as $N = 4$. Recall that we have 2 non-zero operations and 2 input nodes, so the total number of edges in the search space is 14. This results in a search space with the total complexity $(2^2)^{14} - 1 \approx$

10^8, where each architecture parameter is $\alpha \in \{0,1\}^{2 \times 14}$. For the training of the supernet, we choose 7-step PGD adversarial training with 0.01 step size. After training the supernet, we randomly sample 1,000 architectures from the supernet and finetune each of them for 3 epochs.

We plot the histogram of adversarial accuracy of these 1,000 architectures in Fig. 3(b). As the figure shows, although most of the architectures achieve relatively high robustness (with $\sim 50\%$ robust accuracy), there also exist a large number of architectures suffering from poor robustness (far lower from the average 50.3%). This motivates us to consider whether there exist some shared features among the robust networks.

To better visualize how distinguishable the architectures are, we first sort the 1,000 architectures with respect to the robust accuracy. Next, top 300 architectures are selected with a label of 1 and last 300 architectures with label of -1. Finally, t-SNE helps us to depict the α corresponding to each architecture. We visualize the low-dimensional embedding of 600 α in Fig. 4(a). As shown, the embedding of architecture parameter α is separable between robust and non-robust networks, which clearly demonstrates that architecture has an influence on network robustness.

This finding naturally raises a question: *Which paths are crucial to network robustness in architectures?* A straightforward idea is that we train a classifier which takes the architecture parameter as input and predicts whether the architecture is robust to adversarial attacks. In this case, the weights that correspond to crucial paths are expected to have larger values. We use the 600 architectures introduced above and their corresponding labels to train a classifier. Surprisingly, we find out that even a *linear* classifier[1] fits the data well (the training accuracy of these 600 data

---

[1] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
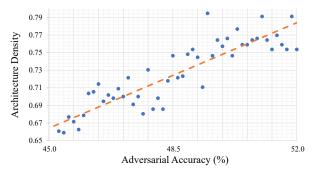
Figure 5. Correlation between architecture density and adversarial accuracy. We show a strong correlation between them, indicating that densely connected pattern can benefit the network robustness.
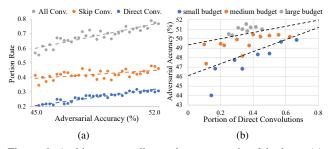

Figure 6. Architecture studies under computational budget: (a) Correlation between the number of different operations and the network robustness. When increasing the number of convolution operations, the adversarial accuracy increases steadily. Moreover, convolutions on direct edges contribute more on robust accuracy than those on skip edges. (b) Performance under different computational budgets. Under *small* and *medium* budget, the proportion of direct convolutions shows a positive correlation to adversarial accuracy, indicating that adding convolution operations to direct edges is more effective to improve model robustness under small computational budget.

points is 98.7%). The results are illustrated in Fig. 4(b). The figure reveals that almost all of the weight values are positive, which indicates a strong relationship between how denser one architecture is wired and how robust it is under adversarial attacks. To further explore the relationship, we perform an analysis of the correlation between adversarial accuracy and the density of the architecture. We define the *architecture density D* as the number of connected edges over the total number of all possible edges in the architecture, which can be expressed as:

$$D = \frac{|E_{connected}|}{|E|} = \frac{\sum_{i,j,k} \alpha_k^{(i,j)}}{|E|}. \qquad (2)$$

We illustrate the result in Fig. 5, which shows that there is a strong correlation between architecture density and adversarial accuracy. We posit that through adversarial training, densely connected patterns in the network are more beneficial against adversarial features and learn to be resistant to them. This gives us the answer to the first question in Sec. 1: *Densely connected pattern can benefit the network robustness*.

3.4. Architecture Strategy under Budget

It has been observed in many previous studies [30, 21] that, within the same family of architectures, increasing the number of parameters of the network would bring improvement of robustness. This is because such procedure will promote the model capacity, and thus can benefit the network robustness. However, if we are given a fixed total number of parameters (or we refer to as a computational budget), how to obtain architectures that are more robust under the limited constraint? In this section, we concentrate on how the pattern of an architecture influences robustness when given different fixed computational budgets. One advantage of our robust architecture search space for this study is that, the number of parameters of a network is positively correlated to the number of convolution operations in the architecture.

We first analyze the number of convolution operations with respect to network robustness, using the 1,000 architectures obtained in Sec. 3.3. The results are illustrated in Fig. 6(a). With the number of convolution operations increases, the adversarial accuracy improves steadily. We also plot the statistics for the number of convolutions on skip edges and direct edges, respectively. The results declare that convolutions on direct edges contribute more on adversarial accuracy than those on skip edges. This inspires us to dig deeper on the effect of the convolutions on direct edges for different computational budgets.

We consider three different computational budgets: *small*, *medium* and *large*. Since the maximum number of convolution operations in the cell-based setting is 14, we set the total number of convolutions smaller than 7 as *small* budget, between 8 and 10 as *medium* and larger than 11 as *large*. For each of the budget, we randomly sample 100 architectures, evaluate their adversarial accuracy following Sec. 3.2 and calculate the proportion of convolutions on direct edges among all convolutions. As illustrated in Fig. 6(b), the adversarial accuracy has clear boundaries between different budgets. Furthermore, for *small* and *medium* budget, the proportion of direct convolutions has a positive correlation to adversarial accuracy. This indicates that for smaller computational budget, adding convolutions to direct edges can efficiently improve the network robustness. We also note that this phenomenon is not obvious for the *large* setting. We speculate that for architectures within the *large* budget, densely connected patterns dominate the contributions of network robustness. With the above results, we conclude: *Under small computational budget, adding convolution operations to direct edges is more effective to improve model robustness*.

3.5. Towards a Larger Search Space

Relax the Cell-Based Constraint. In previous sections, we obtain several valuable observations for the cell-based setting. One natural question to ask is: What if we relax the constraint and permit all the cells in the network to have different architectures? Moreover, what can be the indicator for the network robustness in this cell-free setting? In this section, we relax the cell-based constraint and conduct studies on a larger architecture search space. The relaxation of the constraint raises an explosion of the complexity of the search space: for a network consisting of L cells, the total complexity increases to $(10^8)^L$. The exponential complexity makes the architecture search much more difficult to proceed.

Feature Flow Guided Search. To address the above challenges, here we propose a feature flow guided search scheme. Our approach is inspired by TRADES [41], which involves a loss function minimizing the KL divergence of the logit distribution between an adversarial example and its corresponding clean data. The value of this loss function can be utilized as a measurement of the gap between network robustness and its clean accuracy. Instead of focusing on the final output of a network, we consider the feature flow between the intermediate cells of a network. Specifically, we calculate the Gramian Matrix across each cell, denoted as flow of solution procedure (FSP) matrix [39]. The FSP matrix for the lth cell is calculated as:

$$G_l(x;\theta) = \sum_{s=1}^{h}\sum_{t=1}^{w} \frac{F_{l,s,t}^{in}(x;\theta) \times F_{l,s,t}^{out}(x;\theta)}{h \times w}, \quad (3)$$

where $F_l^{in}(x;\theta)$ denotes the input feature map of the cell and $F_l^{out}(x;\theta)$ denotes the output feature map. For a given network, we can calculate the distance of FSP matrix between adversarial example and clean data for each cell of the network:

$$L_l^{FSP} = \frac{1}{N}\sum_{x} \|(G_l(x;\theta) - G_l(x';\theta)\|_2^2. \quad (4)$$

We sample 50 architectures *without* finetuning for the cell-free search space and evaluate the gap of clean accuracy and adversarial accuracy for each architecture. We also calculate the FSP matrix distance for each cell of the network and show representative results in Fig. 7 (complete results are provided in Appendix). We can observe that for the cells in a deeper position of the network, the FSP distance has a positive correlation with the gap between network robustness and clean accuracy. This gives us the answer to the third question in Sec. 1: *A robust network has a lower FSP matrix loss in the deeper cells of network.*

By observing this phenomenon, we can easily adopt FSP matrix loss to filter out the non-robust architectures with
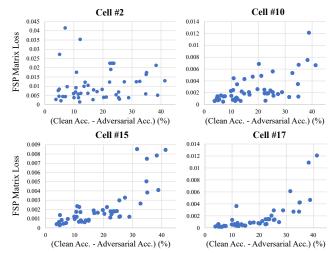

Figure 7. Analysis of FSP matrix distance as robustness indicator. We compute the FSP matrix distance for each cell, along with the performance gap between clean accuracy and adversarial accuracy (complete results in Appendix). For cells in deeper positions of the network, the FSP distance has a positive correlation with the gap between network robustness and its clean accuracy, which indicates that a robust network has a lower FSP matrix loss in the deeper cells of the network.

high loss values, which efficiently reduces the complexity of the search space. Thus, after the sampling process from supernet in cell-free setting, we first calculate FSP matrix loss for each architecture and reject those with high loss values. We then perform finetuning to get final robustness.

4. Experiments

In this section, we empirically evaluate the adversarial robustness of the proposed RobNet family. Following the guidance of our three findings in Sec. 3, we train and select a set of representative RobNet models for evaluation. We focus on l_∞-bounded attacks and compare the RobNet family with state-of-the-art human-designed models.

4.1. Experimental Setup

Implementation Details. As described in Sec. 3, we use both cell-based and cell-free searching algorithm to select out a set of RobNet architectures, respectively. The robust search is performed only on CIFAR-10, where we use PGD adversarial training with 7 attack iterations and a step size of $2/255$ (0.01). For evaluation on other datasets, we directly transfer the RobNet architectures searched on CIFAR-10.

Specifically, we first follow the cell-based robust search framework to obtain architectures that exhibit *densely connected patterns*. Considering the strategy under budget, we further generate three cell-based architectures that all follow *more convolution operations on direct edges*, but with different computational budgets. We refer to the three selected architectures as **RobNet-small**, **RobNet-medium**,

Table 1. White-box attack results on CIFAR-10. We compare representative RobNet models with state-of-the-art architectures. All models are adversarially trained using PGD with 7 steps. All attacks are l_∞-bounded with a total perturbation scale of $8/255$ (0.031).

Models	Model Size	Natural Acc.	FGSM	PGD^{20}	PGD^{100}	DeepFool	MI-FGSM
ResNet-18	11.17M	78.38%	49.81%	45.60%	45.10%	47.64%	45.23%
ResNet-50	23.52M	79.15%	51.46%	45.84%	45.35%	49.18%	45.53%
WideResNet-28-10	36.48M	**86.43%**	53.57%	47.10%	46.90%	51.23%	47.04%
DenseNet-121	6.95M	82.72%	54.14%	47.93%	47.46%	51.70%	48.19%
RobNet-small	4.41M	78.05%	53.93%	48.32%	48.07%	52.96%	48.98%
RobNet-medium	5.66M	78.33%	54.55%	49.13%	48.96%	53.32%	49.34%
RobNet-large	6.89M	78.57%	54.98%	49.44%	49.24%	53.85%	49.92%
RobNet-large-v2	33.42M	85.69%	<u>57.18%</u>	<u>50.53%</u>	<u>50.26%</u>	<u>55.45%</u>	<u>50.87%</u>
RobNet-free	5.49M	82.79%	**58.38%**	**52.74%**	**52.57%**	**57.24%**	**52.95%**

and **RobNet-large**. Furthermore, we leverage *FSP guided search* described in Sec. 3.5 to efficiently generate cell-free robust architectures and select one representative model for evaluation, which is referred to as **RobNet-free**. Note that we are not selecting the *best* architecture, as the searching space is too large to allow us to do so. Instead, we follow the proposed algorithm to select *representative* architectures and study their robustness under adversarial attacks. More details of the selecting process and visualizations of the representative RobNet architectures can be found in Appendix.

We compare RobNet with widely used human-designed architectures, including ResNet [11], Wide-ResNet [40], and DenseNet [12]. All models are adversarially trained using PGD with 7 attack steps and a step size of $2/255$ (0.01). We follow the training procedure as in [21] and keep other hyper-parameters the same for all models.

Datasets & Evaluation. We first perform an extensive study on CIFAR-10 to validate the effectiveness of RobNet against black-box and white-box attacks. We then extend the results to other datasets such as SVHN, CIFAR-100, and Tiny-ImageNet. Finally, we show the benefits from RobNet are orthogonal to existing techniques. We provide additional results on ImageNet, as well as detailed training procedure and hyper-parameters in Appendix.

4.2. White-box Attacks

Main Results. We show the results against various white-box attacks in Table 1. We choose state-of-the-art network architectures that are widely used in adversarial literature [21, 35, 41] for comparison. As illustrated in the table, all the selected models from RobNet family can consistently achieve higher robust accuracy under different white-box attacks, compared to other models.

The strongest adversary in our white-box setting is the PGD attacker with 100 attack iterations (i.e., PGD^{100}). When zoom in to the results, we can observe that by only changing architecture, RobNet can improve the previous

arts under white-box attacks by **5.1%** from 47.5% to 52.6%.

The Effect of Dense Connections. Table 1 also reveals interesting yet important findings on dense connections. ResNet and its wide version (WideResNet) are most frequently used architectures in adversarial training [21, 35, 41]. Interestingly however, it turns out that the rarely used DenseNet model is more robust than WideResNet, even with much fewer parameters. Such observation are well-aligned with our previous study: *densely connected pattern largely benefits the model robustness*. Since RobNet family explicitly reveals such patterns during robust architecture search, they turn out to be consistently robust.

The Effect of Parameter Numbers. Inspired by the finding of computational budget, we seek to quantify the robustness of RobNets with different parameter numbers. We compare three models with different sizes obtained by cell-based search (i.e., RobNet-small, RobNet-medium, and RobNet-large). As Table 1 reports, with larger computational budgets, network robustness is consistently higher, which is well aligned with our arguments.

We note that the model sizes of RobNets are consistently smaller than other widely adopted network architectures. Yet, the natural accuracy of RobNet model is unsatisfying when compared to WideResNet. To further study the influence of network parameters, we extend the RobNet-large model to have similar size as WideResNet by increasing the number of channels and stacked cells, while maintaining the same architecture within each cell. We refer to this new model as **RobNet-large-v2**. It turns out that by increasing the model size, not only can the robustness be strengthened, the natural accuracy can also be significantly improved.

The Effect of Feature Flow Guided Search. When releasing the cell-based constraints during robust searching, RobNet models can be even more robust. We confirm it by comparing the results of RobNet-free, which is obtained using FSP Guided Search as mentioned in Sec. 3.5, with

Table 2. Black-box attack results on CIFAR-10. We compare two representative RobNet architectures with state-of-the-art models. Adversarial examples are generated using transfer-based attack on the same copy of the victim network.

Models	FGSM	PGD^{100}
ResNet-18	56.29%	54.28%
ResNet-50	58.12%	55.89%
WideResNet-28-10	58.11%	55.68%
DenseNet-121	61.87%	59.34%
RobNet-large	**61.92%**	**59.58%**
RobNet-free	**65.06%**	**63.17%**

Table 3. White-box attack results across different datasets. We use two RobNet models searched on CIFAR-10 to directly perform adversarial training on new datasets. We apply PGD^{100} white-box attack on all models to evaluate adversarial robustness.

Models	SVHN	CIFAR-100	Tiny-ImageNet
ResNet-18	46.08%	22.01%	16.96%
ResNet-50	47.23%	22.38%	19.12%
RobNet-large	**51.26%**	**23.19%**	**19.90%**
RobNet-free	**55.59%**	**23.87%**	**20.87%**

other cell-based RobNet models. Remarkably, RobNet-free achieves higher robust accuracy with 6× fewer parameter numbers when compared to RobNet-large-v2 model.

4.3. Black-box Attacks

We further verify the robustness of RobNet family under black-box attacks. We follow common settings in literature [25, 21, 38] and apply transfer-based black-box attacks. We train a copy of the victim network using the same training settings, and apply FGSM and PGD^{100} attacks on the copy network to generate adversarial examples. Note that we only consider the *strongest* transfer-based attacks, i.e., we use *white-box attacks* on the independently trained copy to generate black-box examples.

The results are shown in Table 2. They reveal that both cell-based and cell-free RobNet models are more robust under transfer-based attacks. Note that here the source model has the same architecture as the target model, which makes the black-box adversary stronger [21]. We also study the transfer between different architectures, and provide corresponding results in the Appendix.

4.4. Transferability to More Datasets

So far, our robust searching has only been performed and evaluated on CIFAR-10. However, the idea of robust neural architecture search is much more powerful: we directly use the RobNet family searched on CIFAR-10 to apply on other datasets, and demonstrate their effectiveness. Such benefits

Table 4. Robustness comparison of different architectures with and without feature denoising [35]. We show the benefits from RobNet are orthogonal to existing techniques: RobNet can further boost robustness performance when combined with feature denoising.

Models	Natural Acc.	PGD^{100}
ResNet-18	78.38%	45.10%
ResNet-18 + Denoise	78.75%	45.82%
RobNet-large	78.57%	49.24%
RobNet-large + Denoise	**84.03%**	**49.97%**

come from the natural advantage of NAS that the searched architectures can generalize to other datasets [45, 43].

We evaluate RobNet on SVHN, CIFAR-100, and Tiny-ImageNet under white-box attacks, and set attack parameters as follows: total perturbation of $8/255$ (0.031), step size of $2/255$ (0.01), and with 100 total attack iterations. The training procedure is similar to that on CIFAR-10, where we use 7 steps PGD for adversarial training. We keep all the training hyperparameters the same for all models.

Table 3 shows the performance of RobNet on the three datasets and compares them with commonly used architectures. The table reveals the following results. First, it verifies the effectiveness of RobNet family: they consistently outperform other baselines under strong white-box attacks. Furthermore, the gains across different datasets are different. RobNets provide about 2% gain on CIFAR-100 and Tiny-ImageNet, and yield ∼10% gain on SVHN.

4.5. Boosting Existing Techniques

As RobNet improves model robustness from the aspect of network architecture, it can be seamlessly incorporated with existing techniques to further boost adversarial robustness. To verify this advantage, we select feature denoising technique [35] which operates by adding several denoising blocks in the network. We report the results in Table 4. As shown, the denoising module improves both clean and robust accuracy of RobNet, showing their complementariness. Moreover, when compared to ResNet-18, RobNet can better harness the power of feature denoising, gaining a larger improvement gap, especially on clean accuracy.

5. Conclusion

We proposed a robust architecture search framework, which leverages one-shot NAS to understand the influence of network architectures against adversarial attacks. Our study revealed several valuable observations on designing robust network architectures. Based on the observations, we discovered RobNet, a family of robust architectures that are resistant to attacks. Extensive experiments validated the significance of RobNet, yielding the intrinsic effect of architectures on network resilience to adversarial attacks.

References

[1] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *ICLR*, 2017. 2

[2] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *ICML*, 2018. 2, 3

[3] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *ICLR*, 2018. 2

[4] Han Cai, Chuang Gan, and Song Han. Once for all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019. 2

[5] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, 2017. 2

[6] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, Percy Liang, and John C Duchi. Unlabeled data improves adversarial robustness. *arXiv preprint arXiv:1905.13736*, 2019. 2

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 2

[8] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Xiaolin Hu, Jianguo Li, and Jun Zhu. Boosting adversarial attacks with momentum. In *CVPR*, 2018. 2

[9] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015. 1, 2

[10] Minghao Guo, Zhao Zhong, Wei Wu, Dahua Lin, and Junjie Yan. Irlas: Inverse reinforcement learning for architecture search. In *CVPR*, 2019. 2

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 7

[12] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 2, 7

[13] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*, 2018. 1, 2

[14] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 2

[15] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *ICLR Workshop*, 2017. 1, 2

[16] Fei-Fei Li, Andrej Karpathy, and Justin Johnson. Tiny imagenet visual recognition challenge. 2

[17] Ji Lin, Chuang Gan, and Song Han. Defensive quantization: When efficiency meets robustness. *arXiv preprint arXiv:1904.08444*, 2019. 2

[18] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *ECCV*, 2018. 2, 3

[19] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *ICLR*, 2019. 2, 3, 4

[20] Xin Liu, Huanrui Yang, Ziwei Liu, Linghao Song, Hai Li, and Yiran Chen. Dpatch: An adversarial patch attack on object detectors. *arXiv preprint arXiv:1806.02299*, 2018. 2

[21] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018. 1, 2, 3, 5, 7, 8

[22] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *CVPR*, 2016. 2

[23] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 2

[24] Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. Improving adversarial robustness via promoting ensemble diversity. In *ICML*, 2019. 2

[25] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016. 8

[26] Juan-Manuel Perez-Rua, Moez Baccouche, and Stephane Pateux. Efficient progressive neural architecture search. In *BMVC*, 2018. 2

[27] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI*, 2019. 2

[28] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In *ICLR*, 2018. 1, 2

[29] Robert Stanforth, Alhussein Fawzi, Pushmeet Kohli, et al. Are labels required for improving adversarial robustness? *arXiv preprint arXiv:1905.13725*, 2019. 2

[30] Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. Is robustness the cost of accuracy?– a comprehensive study on the robustness of 18 deep image classification models. In *ECCV*, 2018. 1, 2, 5

[31] Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. A genetic programming approach to designing convolutional neural network architectures. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 497–504. ACM, 2017. 2

[32] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 1, 2

[33] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *ICLR*, 2018. 2

[34] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. In *ICLR*, 2018. 1

[35] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *CVPR*, 2019. 1, 2, 7, 8

[36] Cihang Xie and Alan Yuille. Intriguing properties of adversarial training. *arXiv preprint arXiv:1906.03787*, 2019. 1, 2

[37] Saining Xie, Alexander Kirillov, Ross Girshick, and Kaiming He. Exploring randomly wired neural networks for image recognition. In *ICCV*, 2019. 3

[38] Yuzhe Yang, Guo Zhang, Dina Katabi, and Zhi Xu. ME-Net: Towards effective adversarial robustness with matrix estimation. In *ICML*, 2019. 1, 2, 8

[39] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *CVPR*, 2017. 6

[40] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 7

[41] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically principled trade-off between robustness and accuracy. In *ICML*, 2019. 1, 2, 6, 7

[42] Yuge Zhang, Zejun Lin, Junyang Jiang, Quanlu Zhang, Yujing Wang, Hui Xue, Chen Zhang, and Yaming Yang. Deeper insights into weight sharing in neural architecture search. *arXiv preprint arXiv:2001.01431*, 2020. 4

[43] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical block-wise neural network architecture generation. In *CVPR*, 2018. 2, 8

[44] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017. 2

[45] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2017. 2, 3, 4, 8