On Adversarial Robustness: A Neural Architecture Search perspective

Chaitanya Devaguptapu[1] Devansh Agarwal[1*] Gaurav Mittal[2*] Pulkit Gopalani [3]
Vineeth N Balasubramanian[1]
[1]Indian Institute of Technology, Hyderabad, India
[2]Microsoft
[3]Indian Institute of Technology, Kanpur, India

Abstract

Adversarial robustness of deep learning models has gained much traction in the last few years. Various attacks and defenses are proposed to improve the adversarial robustness of modern-day deep learning architectures. While all these approaches help improve the robustness, one promising direction for improving adversarial robustness is unexplored, i.e., *the complex topology of the neural network architecture. In this work, we address the following question: "Can the complex topology of a neural network give adversarial robustness without any form of adversarial training?". We answer this empirically by experimenting with different hand-crafted and NAS-based architectures. Our findings show that, for small-scale attacks, NAS-based architectures are more robust for small-scale datasets and simple tasks than hand-crafted architectures. However, as the size of the dataset or the complexity of task increases, hand-crafted architectures are more robust than NAS-based architectures. Our work is the first large-scale study to understand adversarial robustness purely from an architectural perspective. Our study shows that random sampling in the search space of DARTS (a popular NAS method) with simple ensembling can improve the robustness to PGD attack by nearly 12%. We show that NAS, which is popular for achieving SoTA accuracy, can provide adversarial accuracy as a free add-on without any form of adversarial training. Our results show that leveraging the search space of NAS methods with methods like ensembles can be an excellent way to achieve adversarial robustness without any form of adversarial training. We also introduce a metric that can be used to calculate the trade-off between clean accuracy and adversarial robustness. Code and pre-trained models will be made available at https://github.com/tdchaitanya/nas-robustness*

*Equal contribution
Corresponding author: cs19mtech11025@iith.ac.in

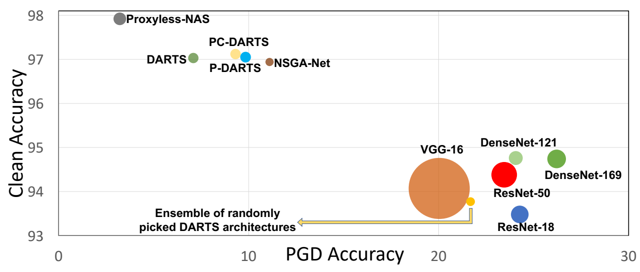

Figure 1. **NAS based architectures are *slightly* better than hand-crafted architectures in terms of test accuracy. But hand-crafted architectures are *significantly* more robust to PGD attacks than NAS architectures**. Qualitative comparison of test-set accuracy and PGD accuracy of NAS and hand-crafted architectures on CIFAR-10 dataset. Bubble size represents the number of parameters.

1. Introduction

The choice of neural network architecture and its complex topology play a crucial role in improving the performance of several deep learning based applications. However, in most cases, these architectures are typically designed by experts in an ad-hoc, trial-and-error fashion. Early efforts on Neural Architecture Search (NAS) [48] alleviate the pain of hand-designing these architectures by partially automating the process of finding the right topology that can result in best-performing architectures. Since the work by [48], there has been much interest in this space. Many researchers have come up with unique approaches [45, 4, 31] to improve the performance besides decreasing the computational cost. Current SoTA (state-of-the-art) on image classification and object detection [38, 39] are developed using NAS, which shows how important a role NAS plays in solving standard learning tasks, especially in computer vision.

Adversarial robustness is defined as the accuracy of a model when adversarial examples (images perturbed with some imperceptible noise) are provided as input. Adversarial examples have the potential to be dangerous. [29] discusses an example where attackers could target autonomous vehicles by using stickers or paint to create an adversarial stop sign that the vehicle could interpret as a yield or other sign. One commonly used technique to improve the adversarial robustness of neural networks is adversarial training, but in most of the cases adversarial training decreases the accuracy on clean (un-perturbed) samples [23, 47]. So, it is essential to develop architectures that are inherently robust without any form of adversarial training. This forms the primary motivation of our work, *Can the complex topology of a neural network architecture provide adversarial robustness without any form of adversarial training?*

In an attempt to understand adversarial robustness purely from an architectural perspective, we seek to answer the following questions,

- In the absence of adversarial training, how do NAS-based architectures compare with hand-crafted architectures (like ResNets [13], DenseNets [14], *etc.*) in terms of adversarial robustness?
- Does an increase in the number of parameters of the architecture help improve robustness?
- Where does the source of adversarial vulnerability lie for NAS? Is it in the search space or in the way the current methods are performing the search?

To the best of our knowledge, our work is the first attempt at understanding adversarial robustness purely from an architectural perspective. We show that the complex topology of neural network architectures can be leveraged to achieve robustness without adversarial training. Additionally, we introduce two simple metrics, *Harmonic Robustness Score (HRS)* and *Per-parameter HRS (PP-HRS)* that combine: (1) the total number of parameters in a model; and (2) accuracy on both clean and perturbed samples, to convey how robust and deployment-ready a given model is when no adversarial training is performed.

We examine the adversarial robustness of different hand-crafted and NAS-based architectures in a wide range of scenarios and find that for large-scale datasets, complex tasks and stronger attacks (like PGD [23]), traditional hand-crafted architectures like ResNets and DenseNets are more robust than NAS-based architectures (Figure 1). This suggests that the adversarial robustness of a model depends significantly on network topology. Results of our study can be used to design network architectures that can give adversarial robustness with no additional adversarial training along with SoTA performance on unperturbed samples.

2. Related Work

Adversarial Attacks and Robustness: Adversarial examples, in general, refer to samples that are imperceptible to the human eye but can fool a deep classifier to predict a non-true class with high confidence. Adversarial examples can result in degraded performance even in the presence of perturbations too subtle to be perceived by humans.

Existing adversarial attacks can be broadly classified into white-box and black-box attacks. The difference between these lies in the knowledge of the adversaries. In white-box attacks, the adversaries have the full knowledge of the target model, including the model architecture and parameters. In a black-box setting, the adversaries knowledge is very limited and may not know details about the model.

In the frameworks of these threat models, several effective adversarial attacks have been proposed over the years such as L-BFGS [37], FGSM [10], BIM [19], C&W attacks [2] JSMA [30], Deep-Fool [25], R-FGSM [40], StepLL [18], PGD [23] and most recently SparseFool [24], F-FGSM [42] and AutoPGD [6]. For more information on adversarial attacks and defenses, please see [3, 33]. White-box is a stronger setting where attackers can access the model parameters and architecture. It is also closely related to the network topology aspect of our study. So we mainly focus on the white-box setting in our work and present some results on black-box attacks in the Appendix.

One popular way to improve the adversarial robustness of deep learning models is adversarial training (AT) [11]. The basic idea of AT is to create and incorporate adversarial samples during training. A critical downside of AT is that it is time-consuming[35]. In addition to the gradient computation needed to update the network parameters, each stochastic gradient descent (SGD) iteration requires multiple gradients computations to produce adversarial images.

Neural Architecture Search (NAS) automates the design of neural network architectures for a given task. Over the years, several approaches have emerged to search architectures using methods ranging from Reinforcement Learning (RL) [48], Neuro-evolutionary approaches [32], Sequential Decision Processes [20], One-shot methods [31] and fully differentiable Gradient-based methods [21]. While most of these algorithms attempt to search a cell architecture (micro search) due to the computational cost involved and repeat the cell a fixed number of times, few recent approaches have also demonstrated searching the full architecture (macro search).

Most of the early approaches are based on RL and neuro-evolutionary algorithms, making the search process computationally intensive. Recently these have been replaced by one-shot fully-differentiable gradient-based NAS methods, such as DARTS [21], which are orders of magnitude faster than non-differentiable techniques and have gained much traction recently. P-DARTS [5] bridges the gap between search and evaluation by progressively increasing search depth. Partially-Connected DARTS [44], a SoTA approach in NAS, significantly improves the efficiency of one-shot NAS by sampling parts of the super-network and adding edge normalization to reduce redundancy and uncertainty in search. DenseNAS [9], a more recent method, attempts to improve search space design by further searching block counts and block widths in a densely connected search space. Despite a plethora of these methods and their applications, there has been minimal effort to understand the adversarial robustness of final learned architectures.

Adversarial Robustness of Architectures: [23] is one of the early papers to talk about adversarial robustness of network architectures. It shows that when training with unperturbed samples, increasing the capacity of the network in terms of width, depth, and the number of parameters can alone help improve the robustness for datasets like MNIST and CIFAR-10. Recently, [43] echoes this observation by showing the depth of the network helps to improve the adversarial robustness during adversarial training. Both [23, 43] talk about robustness mainly in the context of adversarial training. However, our results show that when no adversarial training is performed, increasing parameters alone only helps to a certain point and beyond that, it reduces the adversarial robustness of the model.

Very recently, there have been limited efforts to improve adversarial robustness using architecture search [12, 41]. [12] proposes a robust architecture search framework by leveraging one-shot NAS. However, the proposed method adversarially trains the entire NAS search space before starting the search process, making it harder to assess the contribution of just the architecture to the adversarial robustness. [41] uses black-box attacks to generate a fixed set of adversarial examples on CIFAR-10 and uses these examples to search for a robust architecture using NAS. The experimental setting is constrained and does not reflect the true robustness of the model as the adversarial examples are fixed a priori. No study is done on white-box attacks. Both [12] and [41] do not make any comparisons with existing NAS methods (which, as per our study, are already robust to an extent).

In this work, we mainly focus on evaluating the robustness of SoTA NAS methods on white-box attacks across datasets of different sizes, including large-scale datasets such as ImageNet [7] and compare them with hand-crafted models like ResNets and DenseNets. As a part of our study, we introduce metrics that can be used to estimate the trade-off between clean accuracy and adversarial robustness when comparing architectures within and across different families.

3. Robustness of NAS models: A study

We carefully design our experimental setting to answer the questions stated in Section 1. We begin by describing the design of our experiments, providing details about datasets, models, attacks, and metrics.

Datasets: Since we want to compare the robustness of architectures across different dataset scales and complexities, we choose four different image classification datasets. In addition to the standard CIFAR-10 [16] dataset, which consists of 60K images of 32×32 resolution, we also choose CIFAR-100 [17] to test if the same robustness trends hold when the labels turn from coarse to more fine-grained and the number of classes increase by a factor of 10. To study the robustness trend for tougher tasks like fine-grained image classification where the classes are semantically and perceptually more similar, we choose Flowers-102 dataset [26], which consists of 8189 flowers images split across 102 categories with number of images in each category being between 40 and 258. Since most real-world applications deal with large-scale datasets, we also test robustness on ImageNet [7] dataset, consisting of $\sim$1.3M images from 1000 classes. This makes our study more complete when compared to earlier works.

Architectures: We select most commonly used NAS methods including DARTS [21], P-DARTS [5], Proxyless-NAS [1], NSGA-Net [22], along with recent methods like PC-DARTS [44] and DenseNAS [9]. We evaluate five well-known handcrafted architectures and at least four NAS architectures on each dataset mentioned above for a fair comparison. For all experiments, we either use pre-trained models made available by the respective authors or train the models from scratch until we obtain the performance reported in the respective papers. For the results on Flowers-102 dataset, we explicitly search for an architecture using the code provided by [46]. The results for NSGA-Net are only available for CIFAR-10/100 because its implementation does not support Imagenet. Similarly, the implementation of DenseNAS does not support CIFAR-10/100, so the results are shown only for ImageNet. ProxylessNAS provides pre-trained models for only CIFAR-10 and ImageNet, so we show results only for these two datasets.

Ensemble of Architectures: When compared with single architecture, an ensemble of architectures are known to be adversarially more robust [27]. To understand the effectiveness of ensembling, in Section 4.4, we random sample cells from the DARTS search space using the code provided by [46]; and stack these cells to create small architectures, since this is randomly sampling, the search cost associated with building these architectures is zero. After sampling, we follow the standard DARTS training protocol to train these architectures. In general, for the CIFAR-10 dataset, DARTS architectures having 20 cells are trained for 600 epochs. Following this, the number of epochs for training each network in the ensemble is determined based on the number of cells in that network. Effectively the ensemble as a whole is trained for 600 epochs to ensure we make a fair comparison with existing approaches. After training each of these networks separately, we train a simple linear

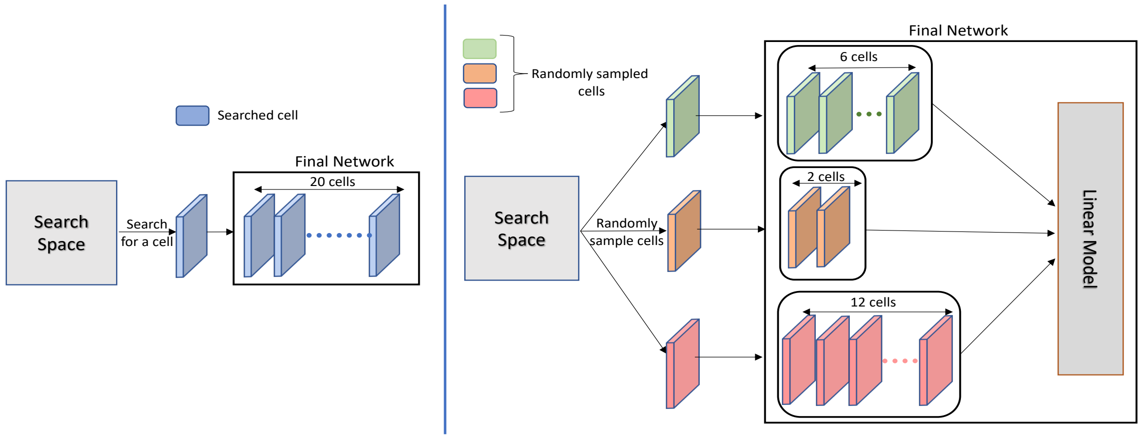

Figure 2. *Left:* Standard procedure for building architectures from DARTS search space; *Right:* Procedure for building ensembles using DARTS search space. 12, 6, 2 can be replaced with any values that sum to 20.

model to combine the individual model outputs. This linear model is trained only for two epochs. The difference between standard DARTS and ensembling by sampling from DARTS search space is visually shown in Figure 2. More details on the structure of the linear model are discussed in Section 4.4

Adversarial Attacks: For adversarial robustness, we test against the standard attacks like FGSM [10], PGD [23] and also report results on recently introduced F-FGSM [42] and AutoPGD [6]. For all these attacks, we use a perturbation value of $8/255(0.03)$ which denotes the maximum noise added to each pixel in the input image as perturbation. The step size is $2/255$ (0.007) with the attack iterations set as 10. Moreover, we run the AutoPGD attack with 10 random restarts. All these parameter choices are standard and widely used in the community [28, 8, 42]. Architectures are trained using standard training protocols, and no adversarial training is performed. We use the library provided by [15] for all the adversarial attacks in our experiments.

Metrics: We use Clean Accuracy and Adversarial Accuracy as our performance metrics. Clean accuracy refers to the accuracy on the unperturbed test set as provided in the dataset. For each attack, we measure Adversarial accuracy by perturbing the test set examples using various attacks in the methods listed in the above section (FGSM, F-FGSM, PGD and AutoPGD).

One of the main problems with adversarially trained models is that their clean accuracy is usually less than standard non-adversarially trained models. Adversarial vulnerability is a side-effect of overfitting to the training set [34]. While this overfitting gives good performance on the clean test set, it makes the model vulnerable to adversarial examples. If the accuracy of a model on clean samples is not good, it is not useful when deployed in situations where unperturbed samples are more frequent. On the other hand, if the model has SoTA performance on a clean test-set, it becomes vulnerable to adversarial examples. There is no well-defined metric to capture this trade-off between clean and adversarial accuracy.

To this end, we introduce a metric, called ***Harmonic Robustness Score (HRS)***, that is defined as the harmonic mean of the clean and adversarial accuracy of a given model. HRS captures the balance of a model's performance to unperturbed inputs and robustness to an adversarial attack . Consider a model with clean accuracy C and Adversarial accuracy A (both in percentage), HRS for that model is calculated as follows:

$$\mathrm{HRS} = 2 \cdot \frac{\mathrm{C} \cdot \mathrm{A}}{\mathrm{C} + \mathrm{A}} \tag{1}$$

The harmonic mean is better at reflecting extreme differences in input values, compared to Arithmetic mean. Therefore, if one of the clean or adversarial accuracy is *very* low, then the harmonic mean of C and A would be more reflective of the same.

A weighted version of HRS, the HRS_β score, can also be used. This is a measure of the model's performance to clean as well as perturbed images, weighted according to what the end-user prefers – clean accuracy, or the adversarial accuracy. For a given use case, one might be preferred more to the other, and hence this metric can be used accordingly. HRS_β is given by,

$$\mathrm{HRS}_\beta = (\beta^2 + 1) \cdot \frac{\mathrm{C} \cdot \mathrm{A}}{\beta^2 \mathrm{C} + \mathrm{A}} \tag{2}$$

where β can be interpreted as the importance of adversarial accuracy over clean accuracy. Since we do not have any particular preference to adversarial accuracy over clean accuracy (or vice-versa), we use $\beta = 1$ for reporting HRS values. The adversarial accuracy for all models is measured on perturbed inputs obtained using the PGD [23] attack. (Choice of PGD is arbitrary, and can be replaced with any other attack).

When comparing performances of architectures belonging to the same family, number of parameters play an important role. A huge parameter difference can easily improve clean and adversarial accuracy, but this comes with huge training and inference time. So we further define per-parameter harmonic robustness score (PP-HRS) to measure the clear accuracy verses adversarial robustness trade-off within a family of architectures. PP-HRS compares the parameters of the model with the parameters in the baseline model of that family. In a family of architectures ($\mathcal{F}$), consider a baseline model m_b having p_b number of parameters, now for a model $m_i \in \mathcal{F}$ with p_i number of parameters, PP-HRS is calculated as follows,

$$\text{PP-HRS} = \text{HRS} * \frac{p_b}{f(p_i)} \tag{3}$$

where the function $f(p_i)$ can be defined as per requirement. We use $f(p_i) = p_i$ as our function of choice for PP-HRS. The main motivation behind this choice is to consider the improvement of accuracy with number of parameters. For example, in the EfficientNet [38] family, we can compute the average increase in accuracy per unit increase in the number of parameters, for the purpose of comparison.

4. Analysis and Results

In this section, we compare and contrast the robustness of different architectures in a wide-range of scenarios and answer questions listed in Section 1.

4.1. How do NAS-based models compare with hand-crafted models in terms of architectural robustness?

The HRS and robustness of different hand-crafted and NAS-based architectures on CIFAR-10, CIFAR-100, ImageNet and Flowers-102 datasets are shown in Tables 1, 2, 3, 4 respectively.

In the case of CIFAR-10 and CIFAR-100, NAS-based architectures outperform hand-crafted architectures in terms of architectural robustness for attacks like FGSM and F-FGSM by a significant margin. However, for stronger and most commonly used attacks like PGD and AutoPGD, NAS-based architectures fail significantly compared to hand-crafted models. In terms of HRS, for CIFAR-10 dataset, the difference in the best-performing NAS and hand-crafted models is 21%.

This trend seen in CIFAR-10/100 for attacks like FGSM and F-FGSM do not hold for large-scale datasets like ImageNet and relatively complex tasks like fine-grained classification. In the case of Imagenet, handcrafted models are more robust than NAS-based architectures for all the attacks. Similarly, for the task of fine-grained classification on Flowers-102 dataset, handcrafted models like DenseNet-169 and VGG-16 beat NAS based architectures by a significant margin. Even in terms of clean accuracy, for which the NAS-based models are generally known to be better than handcrafted models, NAS architectures fail by a margin of ~1.5% for the Flowers-102 dataset.

This trend of robustness for all four datasets is clearly shown in Figure 4. As the dataset size or the task complexity increases, hand-crafted models start to be better for all the three adversarial attacks. For stronger attacks like PGD, handcrafted models are more robust when compared to NAS-based architectures at any given dataset scale. While NAS-based architectures achieve SoTA clean accuracy in general, the robustness of these architectures is very erratic.

Model	Clean %	FGSM	F-FGSM	PGD	AutoPGD	HRS
ResNet-18	93.48	52.43	48.33	24.27	23.13	38.53
ResNet-50	94.38	50.05	45.78	23.45	22.35	37.57
DenseNet-121	94.76	50.94	47.14	24.06	22.66	38.38
DenseNet-169	94.74	53.53	49.47	**26.21**	**24.35**	**41.06**
VGG16 BN	94.07	52.42	46.16	20.03	18.63	33.03
DARTS [21]	97.03	58.53	45.03	7.09	6.10	13.21
PDARTS [5]	**97.12**	58.67	47.62	9.31	7.98	16.99
NSGA Net [22]	96.94	**66.08**	56.16	11.1	9.82	19.92
Proxyless-NAS [1]	97.92	51.73	**58.38**	3.22	4.24	6.23
PC-DARTS [44]	97.05	60.55	48.65	9.84	8.36	17.87

Table 1. **For simple attacks, NAS based architectures are robust, but for strong attacks hand-crafted architectures are better**. Quantitative comparison of clean accuracy and adversarial robustness on **CIFAR-10** dataset (Top-1 Accuracy)

Model	Clean %	FGSM	F-FGSM	PGD	AutoPGD	HRS
ResNet-18	63.87	17.08	17.12	6.05	5.39	11.05
ResNet-50	73.09	19	18.12	5.63	5.16	10.45
DenseNet-121	78.71	22.9	22.22	7.28	6.68	13.33
DenseNet-169	82.44	22.73	21.66	**7.37**	**6.90**	**13.53**
VGG16 BN	72.05	17.09	15.15	4.27	3.81	8.06
DARTS [21]	82.43	24.91	16.34	2.32	1.89	4.51
PDARTS [5]	83.07	27.69	20.23	3.09	2.66	5.96
NSGA Net [22]	**85.44**	**34.93**	**24.1**	2.26	1.94	4.40
PC-DARTS [44]	81.83	26.22	18.35	2.93	2.51	5.66

Table 2. **For simple attacks, NAS based architectures are robust, but for strong attacks hand-crafted architectures are better**. Quantitative comparison of clean accuracy and adversarial robustness on **CIFAR-100** dataset (Top-1 Accuracy)

Model	Clean %	FGSM	F-FGSM	PGD	AutoPGD	HRS
ResNet18	89.08	32.75	18.03	2.41	21.65	4.70
ResNet50	92.86	46.28	26.22	4.68	20.93	8.90
DenseNet121	91.97	56.20	38.11	6.932	24.20	12.89
DenseNet169	92.81	**61.89**	**44.22**	**10.46**	**27.15**	**18.80**
VGG16	91.52	33.34	13.54	1.55	19.57	3.05
DARTS	91.26	54.41	31.18	2.94	20.81	5.70
P-DARTS	92.61	55.53	33.87	4.11	20.67	7.86
PC-DARTS	92.49	58.90	37.86	4.75	21.35	9.04
Proxyless-NAS	92.54	59.56	39.69	6.48	22.28	12.11
DenseNAS-Large	92.80	47.91	27.25	2.97	19.62	5.76
DenseNAS-R3	**93.81**	54.99	32.11	4.32	19.94	8.25

Table 3. **As the scale of the problem increases, hand-crafted architectures are more robust than NAS based architectures**. Quantitative comparison of clean accuracy and adversarial robustness on **ImageNet** dataset (Top-5 Accuracy)

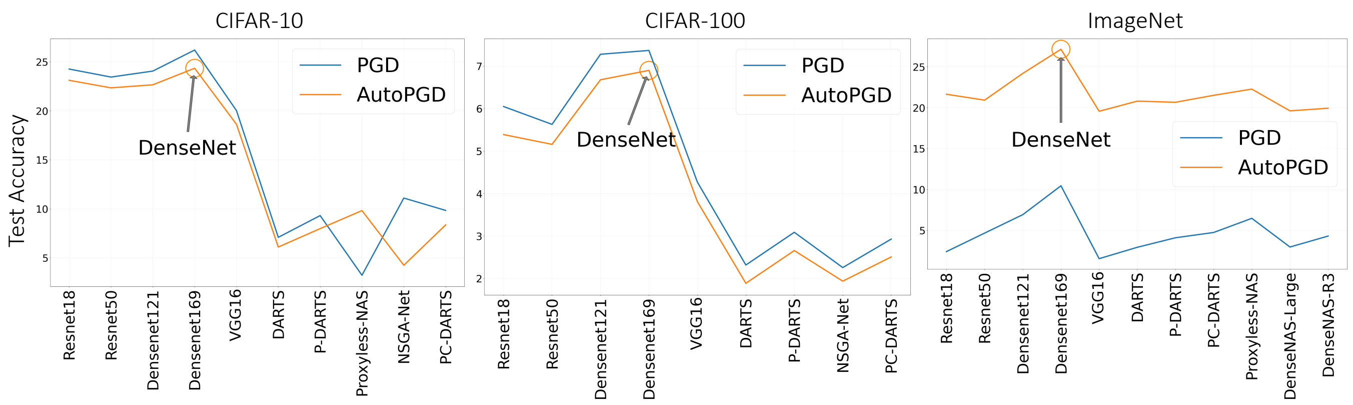

Figure 3. **DenseNets are always more robust**. Qualitative comparison of accuracy of different models for PGD and AutoPGD attacks on CIFAR-10, CIFAR-100 and ImageNet datasets

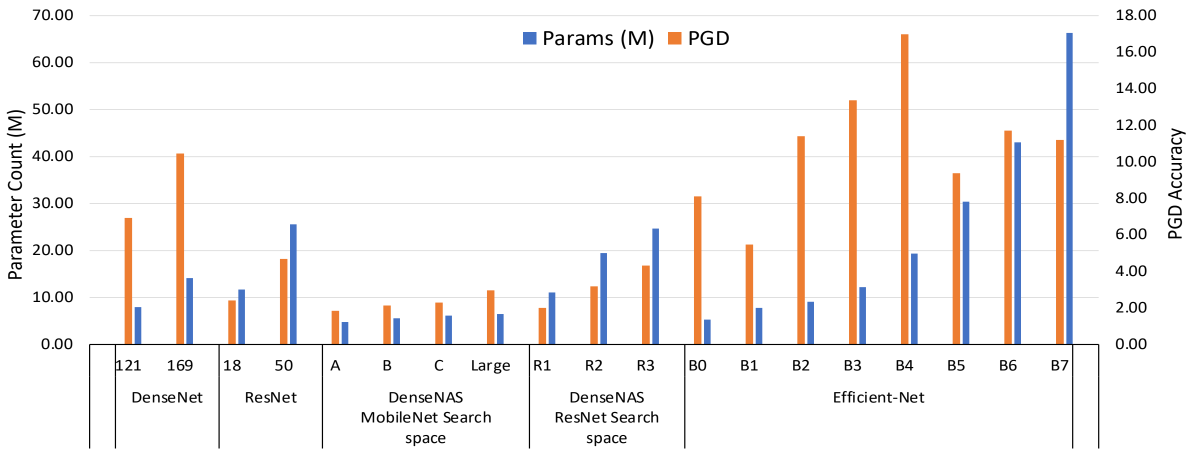

Figure 4. Comparison of PGD accuracy and Parameter count across different family of architectures

Model	Clean %	FGSM	F-FGSM	PGD	AutoPGD	HRS
ResNet-18	95.48	54.33	51.16	11.23	10.38	20.10
ResNet-50	97.31	53.97	52.38	11.36	10.01	20.34
DenseNet-121	97.19	67.4	58.61	16	13.80	27.48
DenseNet-169	**97.44**	69.11	62.76	18.56	16.48	31.18
VGG16 BN	95.24	**72.16**	**66.06**	**27.59**	**26.74**	**42.78**
DARTS [21]	95.97	64.47	59.95	19.29	18.19	32.12
PDARTS [5]	95.12	55.31	51.16	9.52	8.55	17.31
NSGA Net [22]	92.55	40.05	33.58	2.69	2.08	5.23
PC-DARTS [44]	94.02	54.7	45.3	6.84	6.23	12.75

Table 4. **For difficult tasks like fine-grained classification, hand-crafted models with more parameters are robust**. Quantitative comparison of clean accuracy and adversarial robustness on Flowers-102 dataset (Top-1 Accuracy)

In summary, as the dataset size (in terms of both samples and number of classes) or the complexity of the task increases, NAS-based architectures are more vulnerable to adversarial attacks than hand-crafted models when no explicit adversarial training is performed.

4.2. Does an increase in the number of parameters of architecture help improve robustness?

[36] and [23] observed that within the same family of architectures, increasing the number of network parameters helps improve robustness. We therefore hypothesize that increasing model capacity benefits network robustness. To study this claim, we compare the robustness of five families of architectures on the ImageNet dataset with respect to the parameter count. For comparing the trends, we use PGD accuracy along with the Per-parameter Harmonic Robustness Score (PP-HRS). The five different families of architectures we considered for this study are mentioned below.

First, we choose all the eight different variants of the EfficientNet family [38]. EfficientNet is a family of models that are developed by taking a NAS-based base model and scaling its width, depth and input image resolution proportionately using a set of compound-scaling coefficients which are searched via extensive grid search. We also study

Family	Variant	Params (M)	Clean %	PGD	AutoPGD	PP-HRS
Efficient-Net	B0	5.29	91.36	8.11	25.61	**14.90**
	B1	7.79	88.89	5.47	24.90	7.00
	B2	9.11	92.77	11.40	25.77	11.79
	B3	12.23	**93.04**	13.37	27.15	10.11
	B4	19.34	92.73	**16.99**	**29.64**	7.86
	B5	30.39	90.95	9.37	25.28	2.96
	B6	43.04	91.86	11.71	26.17	2.55
	B7	**66.35**	91.57	11.20	27.82	1.59
DenseNAS	A	4.77	90.94	1.84	**19.67**	3.61
	B	5.58	91.89	2.13	19.37	3.56
	C	6.13	92.31	2.29	19.22	3.48
	Large	**6.48**	**92.80**	**2.97**	19.62	**4.24**
	R1	11.09	91.33	2.01	19.77	**3.93**
	R2	19.47	92.47	3.19	19.60	3.51
	R3	**24.66**	**93.81**	**4.32**	**19.94**	3.71
ResNet	18	11.69	89.08	2.41	**21.65**	**4.69**
	50	**25.56**	**92.86**	**4.68**	20.93	4.08
DenseNet	121	7.98	91.97	6.93	24.20	**12.89**
	169	**14.15**	**92.81**	**10.46**	**27.15**	10.60

Table 5. Comparison of parameter count vs Adversarial accuracy for five different family of architectures on ImageNet dataset

a recent family of SoTA NAS-based models called DenseNAS [9]. DenseNAS architectures are developed using two different search spaces. DenseNAS-A/B/C and Large are developed using a MobileNetV2-based search space, and DenseNAS-R1, R2, R3 are developed using a ResNet-based search space. These networks are listed in the increasing order of their parameters. Lastly, to also understand the trend in hand-crafted models, we study the robustness of standard DenseNet and ResNet models.

All the results of this comparison are shown in Table 5 and Figure 4. In 4 out of 5 families considered for this study, an increase in parameters increases both clean and adversarial accuracy. The maximum value of the parameter count in these four families in nearly 26 million. This trend of increase in robustness with parameter count is also seen in the fifth family (EfficientNet) but only up to a parameter count of 20 million. Increasing the parameters *alone* beyond 20 million results in a decrease of both clean and adversarial accuracy. This is probably why EfficientNet considers different image sizes for each of the eight networks. After a certain point, increasing the parameters alone will not help improve robustness, and EfficientNet, which has the best adversarial accuracy in the case of ImageNet dataset, conveys this.

"In what family of architectures, is the increase in parameter count helping the performance?" To better understand this, we report PP-HRS in Table 5. In the case of DenseNAS models developed using MobileNet-V2 search space, an increase in parameters from DenseNAS-A to DenseNAS-Large is improving both clean accuracy and adversarial robustness, which as a result lead to improved PP-HRS. For all the other families, the increased parameter count does not give a significant and sufficient improvement in the PP-HRS and adversarial robustness.

In summary, adversarial robustness can be improved by increasing the number of parameters, but this holds only to an extent. Beyond a certain point (approximately 20-25 million as per our analysis), increasing parameters alone cannot improve adversarial robustness.

4.3. What makes EfficientNets more robust than other architectures?

In comparison to the best performing NAS and hand-crafted architectures in Table 5 (and as discussed in Section 4.1), the family of EfficientNet models are significantly better in terms of robustness to adversarial attacks. Among all the architectures compared, EfficientNet-B0 has the highest PP-HRS of 14.90. In case of PGD, the best performing EfficientNet model, EfficientNet-B4, outperforms all hand-crafted architectures by atleast 6%. This is a significant improvement particularly at the scale of ImageNet dataset. Further, for AutoPGD, all EfficientNet models perform better than all hand-crafted architectures (except for DenseNet-169 which is still worse than EfficientNet-B4 and B7).

One significant difference between EffcientNet and existing NAS and hand-crafted models is the scaling factor. Most of the hand-crafted and NAS-based architectures are developed in a micro-style, *i.e.*, a small cell (like the ResNet block or DARTS cell) is developed/searched, and it is stacked to build the full architectures of varying depths and parameter sizes. In the case of EfficientNet, this scaling is done systematically using a compound scaling method using a coefficient (ϕ) to scale width, depth, and resolution in a principled way[38]. This ϕ is specified by the user to control the resources available for scaling the model proportionately in terms of width, depth and resolution. In our analysis, for consistency, we keep the image resolution fixed at 224×224.

Letting NAS figure out the optimal way to scale a neural network would alleviate the compute required for grid-search (for hyperparameters α, β, γ in EfficientNet) and makes the complete process of finding an adversarially robust architecture end-to-end. But since Section 4.2 shows that NAS-based architectures are more vulnerable than hand-crafted ones for larger and complex datasets, it is important to better understand the source of this vulnerability to find more effective ways to scale neural networks that are also adverasarially robust. We address this in the next section.

4.4. Where does the source of adversarial vulnerability lie for NAS? Is it in the search space or in the way the current methods are performing the search?

In Section 4.2, we see that NAS-based architectures are more robust than hand-crafted architectures for small-scale datasets and simpler attacks. However, for stronger attacks like PGD, NAS-based architectures are not robust even at the scale of CIFAR-10. Most of the existing NAS methods perform the search on CIFAR-10 or a subset of ImageNet, and the discovered cell is stacked and trained for other datasets. To understand whether the problem lies in the search space or in the way search is being performed by the existing methods, we performed two simple experiments.

Our first experiment is motivated by [46]. [46] shows that a randomly sampled cell in the DARTS search space gives as good a clean accuracy as a searched cell. To test if this fact also holds for the case of adversarial robustness, we sampled random cells from the DARTS search space, stacked and trained them using the standard procedure, and tested their robustness on the CIFAR-10 dataset. Due to the randomness involved, we report the value over four different runs. Results of this experiment are shown in Table 6. We can observed that randomly sampled cells have a better PGD accuracy than the searched architecture. But the variance is very high which shows that relying on randomly sampled architectures for adversarial robustness is not a good idea. This leads us to our second experiment.

For the second experiment, we randomly sample cells from the DARTS search space to build small models (please refer to Figure 2 that illustrates this procedure). After training these models independently, we ensemble the outputs of all these models using a simple linear network. This linear model consists of 2 linear layers with batch normalization and one fully connected classifier layer towards the end that outputs logits based on the number of classes in the dataset. This linear model is just fine-tuned for two epochs. Entire ensemble is treated as one single-network when generating the adversarial examples. To make a fair comparison, we ensure that the ensemble as a whole has the same number of cells as the standard DARTS networks. Since the procedure uses randomly sampled architectures, we run the entire sample-train-ensemble procedure four times and report the mean value in Table 6. Due to the randomness involved, this is a computationally expensive procedure. Therefore, we restrict our experiments to CIFAR-10 dataset and DARTS search space.

Model	# cells	Params (M)	Clean %	PGD	AutoPGD
DARTS [21]	20	3.35	97.03	7.09	6.10
P-DARTS [5]	20	3.43	**97.12**	9.31	7.98
PC-DARTS [44]	20	3.63	97.05	9.84	8.36
RANDOM*	20	2.73 ± 0.49	95.57 ± 0.40	14.47 ± 4.70	12.56 ± 4.16
ENSEMBLE†	20	2.74 ± 0.41	93.77 ± 0.39	$\mathbf{21.68 \pm 0.35}$	20.78 ± 0.39

Value reported over four runs. $\star$ Randomly picked architectures from DARTS search-space. Value reported over four runs. † Ensemble of small, randomly picked architectures from DARTS search space.

Table 6. **Ensemble of randomly sampled DARTS cells is significantly more robust than a searched architecture**. Adversarial accuracy comparison of DARTS-based architectures on CIFAR-10.

Surprisingly this simple ensemble of randomly sampled architectures can improve the PGD accuracy of DARTS based models by nearly 12% and can decrease the variance by $\sim$10%. Now, this leads to the following interesting conclusions: (1) Learning to build a simple network to combine the outputs of randomly sampled architectures can give clean accuracy with adversarial robustness as an add-on. In this case, we used a simple linear model; replacing this with a searched NAS based architecture can improve the results further. (2) Using NAS to search for an ensemble of architectures can be a potential way to achieve adversarial robustness as an add-on to SoTA clean accuracy. In this case, the NAS objective should be modified to find small models that can complement each other. We plan to explore this in our future work. (3) Both random and ensemble-based topologies are able to provide significantly better adversarial robustness than existing NAS algorithms. This suggests that the search space itself is not the source of vulnerability. Rather, we need better search algorithms, potentially ensemble-based, that can leverage the same search space to build architectures that are inherently more robust even without any explicit adversarial training.

5. Conclusion

We present a detailed analysis of the adversarial robustness of NAS and hand-crafted models and show how the complex topology of neural networks can be leveraged to achieve adversarial robustness without any form of adversarial training. We also introduce a metric that can be used to calculate the trade-off between clean and adversarial accuracy within and across different families of architectures. Finally, we show that using NAS to find an ensemble of architectures can be one potential way to build robust and reliable models without any form of adversarial training.

Acknowledgement

This work has been partly supported by the funding received from DST, Govt of India, through the Data Science cluster of the ICPS program (DST/ICPS/CLUSTER/Data Science/2018/General).

References

[1] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2019. 3, 5

[2] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. *arXiv e-prints*, page arXiv:1608.04644, Aug. 2016. 2

[3] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial Attacks and Defences: A Survey. *arXiv e-prints*, page arXiv:1810.00069, Sept. 2018. 2

[4] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1294–1303, 2019. 1

[5] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive Differentiable Architecture Search: Bridging the Depth Gap between Search and Evaluation. *arXiv e-prints*, page arXiv:1904.12760, Apr. 2019. 2, 3, 5, 6, 8, 11

[6] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *arXiv e-prints*, page arXiv:2003.01690, Mar. 2020. 2, 4

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 3

[8] Jiameng Fan and Wenchao Li. Adversarial Training and Provable Robustness: A Tale of Two Objectives. *arXiv e-prints*, page arXiv:2008.06081, Aug. 2020. 4

[9] Jiemin Fang, Yuzhu Sun, Qian Zhang, Yuan Li, Wenyu Liu, and Xinggang Wang. Densely Connected Search Space for More Flexible Neural Architecture Search. *arXiv e-prints*, page arXiv:1906.09607, June 2019. 3, 7

[10] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv e-prints*, page arXiv:1412.6572, Dec. 2014. 2, 4

[11] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv e-prints*, page arXiv:1412.6572, Dec. 2014. 2

[12] Minghao Guo, Yuzhe Yang, Rui Xu, and Ziwei Liu. When nas meets robustness: In search of robust architectures against adversarial attacks. *CVPR*, 2020. 3

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv e-prints*, page arXiv:1512.03385, Dec. 2015. 2

[14] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely Connected Convolutional Networks. *arXiv e-prints*, page arXiv:1608.06993, Aug. 2016. 2

[15] Hoki Kim. Torchattacks: A pytorch repository for adversarial attacks. *arXiv preprint arXiv:2010.01950*, 2020. 4

[16] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). 3

[17] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research). 3

[18] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv e-prints*, page arXiv:1607.02533, July 2016. 2

[19] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial Machine Learning at Scale. *arXiv e-prints*, page arXiv:1611.01236, Nov. 2016. 2

[20] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018. 2

[21] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable Architecture Search. *arXiv e-prints*, page arXiv:1806.09055, June 2018. 2, 3, 5, 6, 8, 11

[22] Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. NSGA-Net: Neural Architecture Search using Multi-Objective Genetic Algorithm. *arXiv e-prints*, page arXiv:1810.03522, Oct. 2018. 3, 5, 6

[23] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv e-prints*, page arXiv:1706.06083, June 2017. 2, 3, 4, 6

[24] Apostolos Modas, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Sparsefool: A few pixels make a big difference. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[25] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: a simple and accurate method to fool deep neural networks. *arXiv e-prints*, page arXiv:1511.04599, Nov. 2015. 2

[26] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008. 3

[27] Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. Improving adversarial robustness via promoting ensemble diversity. volume 97 of *Proceedings of Machine Learning Research*, pages 4970–4979, Long Beach, California, USA, 09–15 Jun 2019. PMLR. 3

[28] Tianyu Pang, Xiao Yang, Yinpeng Dong, Hang Su, and Jun Zhu. Bag of Tricks for Adversarial Training. *arXiv e-prints*, page arXiv:2010.00467, Oct. 2020. 4

[29] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical Black-Box Attacks against Machine Learning. *arXiv e-prints*, page arXiv:1602.02697, Feb. 2016. 1

[30] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The Limitations of Deep Learning in Adversarial Settings. *arXiv e-prints*, page arXiv:1511.07528, Nov. 2015. 2

[31] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient Neural Architecture Search via Parameter Sharing. *arXiv e-prints*, page arXiv:1802.03268, Feb. 2018. 1, 2

[32] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019. 2

[33] Kui Ren, Tianhang Zheng, Zhan Qin, and Xue Liu. Adversarial attacks and defenses in deep learning. *Engineering*, 6(3):346 – 360, 2020. 2

[34] Amartya Sanyal, Puneet K Dokania, Varun Kanade, and Philip H. S. Torr. How benign is benign overfitting? *arXiv e-prints*, page arXiv:2007.04028, July 2020. 4

[35] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for

free! In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 3358–3369. Curran Associates, Inc., 2019. 2

[36] 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. *arXiv e-prints*, page arXiv:1808.01688, Aug. 2018. 6

[37] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv e-prints*, page arXiv:1312.6199, Dec. 2013. 2

[38] Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv e-prints*, page arXiv:1905.11946, May 2019. 1, 5, 6, 7

[39] Mingxing Tan, Ruoming Pang, and Quoc V. Le. EfficientDet: Scalable and Efficient Object Detection. *arXiv e-prints*, page arXiv:1911.09070, Nov. 2019. 1

[40] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble Adversarial Training: Attacks and Defenses. *arXiv e-prints*, page arXiv:1705.07204, May 2017. 2

[41] Danilo Vasconcellos Vargas, Shashank Kotyan, and SPM IIIT-NR. Evolving robust neural architectures to defend from adversarial attacks. *arXiv preprint arXiv:1906.11667*, 2019. 3

[42] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv e-prints*, page arXiv:2001.03994, Jan. 2020. 2, 4

[43] Cihang Xie and Alan Yuille. Intriguing properties of adversarial training at scale. *arXiv e-prints*, page arXiv:1906.03787, June 2019. 3

[44] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. In *International Conference on Learning Representations*, 2020. 2, 3, 5, 6, 8, 11

[45] Shen Yan, Biyi Fang, Faen Zhang, Yu Zheng, Xiao Zeng, Hui Xu, and Mi Zhang. HM-NAS: Efficient Neural Architecture Search via Hierarchical Masking. *arXiv e-prints*, page arXiv:1909.00122, Aug. 2019. 1

[46] Antoine Yang, Pedro M. Esperança, and Fabio M. Carlucci. Nas evaluation is frustratingly hard. In *International Conference on Learning Representations*, 2020. 3, 8

[47] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically Principled Trade-off between Robustness and Accuracy. *arXiv e-prints*, page arXiv:1901.08573, Jan. 2019. 2

[48] Barret Zoph and Quoc V. Le. Neural Architecture Search with Reinforcement Learning. *arXiv e-prints*, page arXiv:1611.01578, Nov. 2016. 1, 2