Neural Architecture Search for Wide Spectrum Adversarial Robustness

Zhi Cheng[1*], Yanxi Li[1], Minjing Dong[1], Xiu Su[1], Shan You[2], Chang Xu[1*]

[1]School of Computer Science, Faculty of Engineering, The University of Sydney
[2]SenseTime Research
{zche6824, yali0722, mdon0736, xisu5992}@uni.sydney.edu.au, youshan@sensetime.com, c.xu@sydney.edu.au

Abstract

One major limitation of CNNs is that they are vulnerable to adversarial attacks. Currently, adversarial robustness in neural networks is commonly optimized with respect to a small pre-selected adversarial noise strength, causing them to have potentially limited performance when under attack by larger adversarial noises in real-world scenarios. In this research, we aim to find Neural Architectures that have improved robustness on a wide range of adversarial noise strengths through Neural Architecture Search. In detail, we propose a lightweight Adversarial Noise Estimator to reduce the high cost of generating adversarial noise with respect to different strengths. Besides, we construct an Efficient Wide Spectrum Searcher to reduce the cost of adjusting network architecture with the large adversarial validation set during the search. With the two components proposed, the number of adversarial noise strengths searched can be increased significantly while having a limited increase in search time. Extensive experiments on benchmark datasets such as CIFAR and ImageNet demonstrate that with a significantly richer search signal in robustness, our method can find architectures with improved overall robustness while having a limited impact on natural accuracy and around 40% reduction in search time compared with the naive approach of searching. Codes available at: https://github.com/zhicheng2T0/Wsr-NAS.git

Introduction

With the tremendous success of Convolutional Neural Networks (CNNs) (He et al. 2015; Szegedy et al. 2016; Howard et al. 2017; Cai et al. 2017), more and more CNNs are applied in security-sensitive settings. However, CNNs have a major limitation where they are vulnerable under adversarial attack (Madry et al. 2019; Goodfellow, Shlens, and Szegedy 2014; Dong et al. 2018), a small and targeted noise can mislead the network. To deal with such a limitation, various robust training techniques have been proposed for CNNs (Madry et al. 2019; Goodfellow, Shlens, and Szegedy 2014; Zhang et al. 2019; Shafahi et al. 2019; Wong, Rice, and Kolter 2020; Zhang et al. 2020). Moreover, there also exists a class of techniques which train networks to gain performance guarantees when the adversarial noise strength is under a certain strength limit (Cohen, Rosenfeld, and Kolter 2019; Lütjens, Everett, and How 2019; Li et al. 2019; Lécuyer et al. 2019). Currently, a few pioneering works have also attempted to address the adversarial robustness limitation of CNNs in the perspective of Neural Architecture Search (Guo et al. 2020; Dong et al. 2020; Mok et al. 2021; Hosseini, Yang, and Xie 2021; Sun et al. 2023).

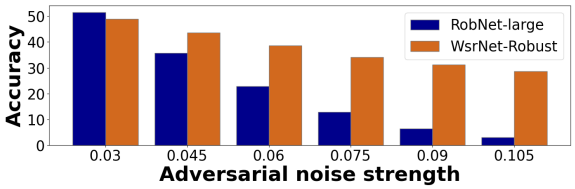

Figure 1: Robust accuracy comparison at different adversarial noise strengths between RobNet-large(Guo et al. 2020) (6.9M parameters, clean accuracy 78.6%) and the proposed WsrNet-Robust(4.5M parameters, clean accuracy 78.4%).

However, a common limitation of the existing research is that the robustness of the networks is commonly optimized on a pre-selected small adversarial noise strength, causing there to be a potentially limited performance when under stronger attacks at some larger unseen strengths (as in Fig. 1), leaving the networks vulnerable to adversarial attack under real-world scenarios.

Currently, there are limited works that deal with the issue of simultaneously improving network robustness under different adversarial noise strengths. From the perspective of training, the only existing method (Song et al. 2018) requires simultaneously training multiple networks that have good adversarial robustness at different ranges of adversarial noise strengths. With the different networks, a more robust combined decision can be made. We argue the design would not only bring a considerable increase in model size but also brings a considerable increase in the computational cost when training and testing the network.

In this research, we aim to address the limitations in the existing research by finding Neural Architectures with wide spectrum adversarial robustness (WsrNets). When trained with the commonly used adversarial training techniques (i.e. TRADES (Zhang et al. 2019) or PGD (Madry et al. 2019)), on a single model without having a significant increase in model parameters (opposing to the multiple models trained

*Corresponding authors

in parallel as in (Song et al. 2018)), the WsrNets found have improved average robust accuracy (average model accuracy on adversarial examples with different adversarial noise strengths) while maintaining high clean accuracy (accuracy on clean data without adversarial noises). To find such WsrNets, we propose a search algorithm named **Neural Architecture Search for Wide Spectrum Adversarial Robustness(Wsr-NAS)** leveraging the One-Shot-NAS framework (Liu, Simonyan, and Yang 2018; Xu et al. 2020).

To prevent a significant increase in computational costs when simultaneously generating adversarial noise at different strengths during the search, we propose a lightweight **Adversarial Noise Estimator (AN-Estimator)**. The AN-Estimator can be trained to generate adversarial noises for each input based on a few existing adversarial noises corresponding to the input, allowing adversarial noises at different strengths to be generated at a much lower cost.

On the other hand, in the One-Shot-NAS framework, adjusting network architecture during search requires calculating gradient w.r.t. to the architecture weights on the validation set. To reduce the cost of performing such an operation on multiple adversarial validation sets, we propose an **Efficient Wide Spectrum Searcher (EWSS)**. To adjust the network architecture during the search, only a backward pass over the lightweight EWSS is required, significantly reducing the computational cost.

One may find it counter-intuitive that architectural search can benefit adversarial robustness. An intuitive example is that in DSNet(Du et al. 2021), by having multiple branches with vastly different designs in its network blocks, the difficulty for the adversarial noise to simultaneously attack all branches increases, hence increasing the adversarial robustness of the network. In our research, by using NAS, we give the algorithm freedom in finding architectures with various properties beneficial for improving adversarial robustness.

With extensive experiments, we found that by having a much richer search signal in robustness, our proposed Wsr-NAS algorithm can find WsrNets that have improved overall robustness while having around 40% reduction in search time relative to the naive technique of searching. Moreover, we have also empirically validated that the WsrNets generalizes cross different training techniques (e.g. TRADES and PGD), different datasets (e.g. CIFAR-10(Krizhevsky and Hinton 2009) and ImageNet(Deng et al. 2009)) and under different attack algorithms (e.g. FGSM(Goodfellow, Shlens, and Szegedy 2014), PGD(Madry et al. 2019) and Black-Box attack(Papernot, McDaniel, and Goodfellow 2016a)).

Related Works

Adversarial Attack. In terms of adversarial attack, the most commonly used attack techniques are FGSM (Goodfellow, Shlens, and Szegedy 2014) and PGD (Madry et al. 2019). Some other existing techniques includes (Dong et al. 2018; Szegedy et al. 2014; Goodfellow, Shlens, and Szegedy 2015; Papernot, McDaniel, and Goodfellow 2016b). The goal of the adversarial attacks are defined as finding a noise δ with strength smaller than ϵ on a dataset D, so that when the noise is added on input x (with labels y), the Neural Network under attack (represented by θ) would output a prediction desired by the adversary (through maximizing loss L set by the adversary). More formally, the attack model is defined as:

$$\delta = \mathbf{argmax}_{\delta, s.t. ||\delta|| < \epsilon} L(\theta, x + \delta, y). \quad (1)$$

In this research, we follow the same formulation.

Adversarial Training. To improve the adversarial robustness of deep learning models, currently there exist various forms of adversarial training techniques (Goodfellow, Shlens, and Szegedy 2014; Madry et al. 2019; Shafahi et al. 2019; Zhang et al. 2019, 2020; Wong, Rice, and Kolter 2020). Within adversarial training, to reduce the impact of adversarial noise on the networks, adversarial examples are generated to train the networks. However, adversarial training commonly comes at the cost of slightly reducing network accuracy on natural images(Zhang et al. 2019).

One-Shot Neural Architecture Search. The overall goal of Neural Architecture Search (NAS) is to automatically discover network architectures leveraging search. The NAS method used in this research is a class of differentiable One-Shot NAS algorithms (Liu, Simonyan, and Yang 2018; Xu et al. 2020; Li et al. 2020) that is highly efficient in time. In terms of performing robust-NAS, the differentiable One-Shot NAS framework is commonly leveraged (Dong et al. 2020; Mok et al. 2021; Zoph et al. 2018; Bai et al. 2021). Within this class of search, a super-network is maintained to allow different sub-networks to be sampled and evaluated according to architecture weights assigned to different operations in the network blocks (cells). By updating the architecture weights through back-propagation on a validation set following a search objective function, the cell architecture can be adjusted efficiently.

Neural Architecture Search for Wide Spectrum Robustness

To find Neural Architectures with wide spectrum adversarial robustness (WsrNets) that have improved robustness on a wide range of adversarial noise strengths, we propose an algorithm named Neural Architecture Search for Wide Spectrum Adversarial Robustness (Wsr-NAS). To reduce the cost of generating adversarial noises at different strengths, we proposed an Adversarial Noise Estimator. To reduce the cost of updating network architecture during the search, we leverage a variant of the One-Shot-NAS framework(Liu, Simonyan, and Yang 2018; Xu et al. 2020) named PVLL(Li et al. 2020), where it facilitates search by leveraging a validation loss estimator.

Preliminaries on Super-Network and Search Space

Within Wsr-NAS, we maintain a super-network with micro search space (Liu, Simonyan, and Yang 2018). The super-network has a hierarchical architecture and the entire network is formed by a stack of cells with identical configurations. Within the super-network, the cells can be represented by a Directed-Acyclic-Graph where each cell contains 7 nodes denoted as $nodes = \{\mathbf{I^{(i)}} | 0 \leq i \leq 6\}$ (each node represents a set of hidden feature maps). Within the 7 nodes, $\mathbf{I}^{(0)}$ and $\mathbf{I}^{(1)}$ are outputs from the two previous cells,

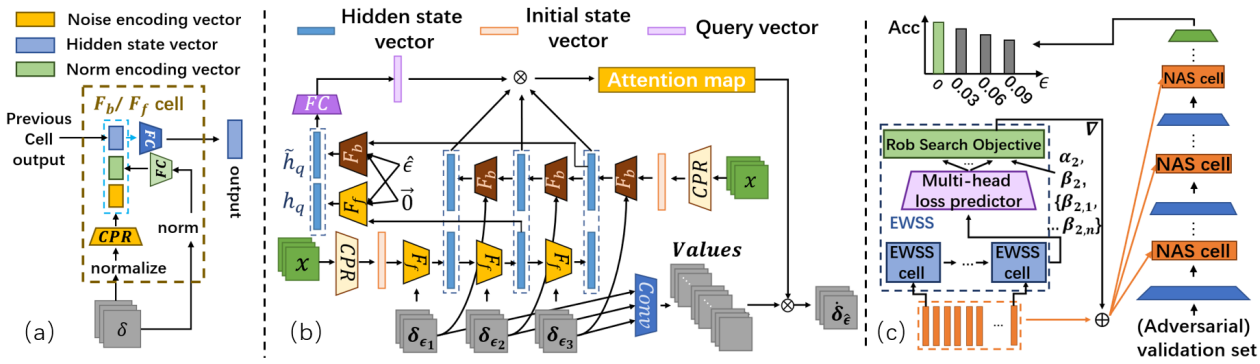

Figure 2: (a) Architecture of hidden cells (F_b or F_f) in the AN-Estimator. (b) Architecture of the AN-Estimator. (c) Illustration of Efficient Wide Spectrum Searcher. Given different α, β and β_i, architectures can be updated accordlingly.

node $\mathbf{I}^{(2)}$ to node $\mathbf{I}^{(5)}$ are calculated by:

$$\mathbf{I}^{(j)} = \sum_{i<j}^{} \sum_{k=1}^{K} h_{i,j}^{(k)} \cdot O^{(k)}(\mathbf{I}^{(i)}), \tag{2}$$

where $O^{(k)}(\cdot)$ represents the k^{th} candidate operation (such as convolution with different kernel sizes, pooling layer or residual connections) on the edge from $\mathbf{I}^{(\mathbf{i})}$ to $\mathbf{I}^{(j)}$; $h_{i,j}^{(k)} \in \mathbf{H}$ represents the weighting calculated using GumbelSoftmax function according to adjustable weight $a_{i,j}^{(k)} \in \mathbf{A}$ corresponding to each operations on each edges. The output node is formed by $\mathbf{I}^{(6)} = \cup_{i=2}^{5} \mathbf{I}^{(i)}$. By adjusting the weight $a_{i,j}^{(k)}$ and having it normalized by the GumbelSoftmax function, the architecture of the super-network can be adjusted. By having the weight $a_{i,j}^{(k)}$ adjusted according to a search objective and sampling a sub-cell according to the largest weights in $a_{i,j}^{(k)}$, a final cell that minimizes the loss of the search objective can be obtained to form the final network.

Adversarial Noise Estimator

Within Wsr-NAS, we propose an AN-Estimator to reduce the high cost of simultaneously generating adversarial noises at different strengths using the super-network during the search. When given an clean input $\mathbf{x} \in \mathbb{R}^{c \times w \times h}$ (c is the number of channels, w and h are the width and height of the image) where adversarial noises at different strengths need to be generated, the proposed AN-Estimator takes in $\mathbf{x}$, a set of N_1 adversarial noises $\{\delta_{\epsilon_1}, ..., \delta_{\epsilon_{\mathbf{N_1}}}\}$ corresponding to $\mathbf{x}$ that have different strengths and generate a new adversarial noise $\hat{\delta}_{\hat{\epsilon}}$ for $\mathbf{x}$ at a different strength. Hence, the AN-Estimator can be expressed as:

$$\hat{\delta}_{\hat{\epsilon}} = \phi(\mathbf{x}, \delta_{\epsilon_1}, ..., \delta_{\epsilon_{\mathbf{N_1}}}, \hat{\epsilon}). \tag{3}$$

To facilitate AN-Estimator training during the search, we include a memory

$$M_a = \{(\{x_t, \delta_{t,\epsilon_1}, ..., \delta_{t,\epsilon_{N_1}}, \hat{\epsilon}_t\}, \delta_{\hat{\epsilon}_t}), \forall 1 \leq t \leq T_a\}, \tag{4}$$

where the memory has size T_a, $\delta_{\hat{\epsilon}_t}$ is the label for the input $\{x_t, \delta_{t,\epsilon_1}, ..., \delta_{t,\epsilon_{N_1}}, \hat{\epsilon}_t\}$ at index t and the memory is a queue with a limited length. As the Wsr-NAS algorithm progresses, we generate a small set of new training data for the AN-Estimator once the super-network is updated and trained. in this way, the latest, most efficient adversarial noises for the super-network can be stored in the memory for training the AN-Estimator, while the older, less effective ones will be discarded. Such a design allows the AN-Estimator to be trained and updated as the super-network changes during the search.

To train the AN-Estimator, we propose to minimize the MSE loss defined as:

$$L_a(\phi) = \frac{1}{T_a} \sum_{t=1}^{T_a} \|\phi(x_t, \delta_{t,\epsilon_1}, ..., \delta_{t,\epsilon_{N_1}}, \hat{\epsilon}_t) - \delta_{\hat{\epsilon}_t}\|_2^2. \tag{5}$$

To reduce computational cost when searching for WsrNets, we impose a lightweight requirement on the AN-Estimator. Hence, to generate adversarial noise with good quality, the architecture of the AN-Estimator requires careful design. In this research, we propose three potentially effective architecture designs to be investigated empirically.

Weighted Average AN-Estimator. As a naive baseline, we propose the weight-less Weighted Average AN-Estimator (WA-ANE). This WA-ANE generates a new adversarial noise with strength $\hat{\epsilon}$ of an input x by calculating the weighted average of existing adversarial noises of input x that has neighboring strengths with $\hat{\epsilon}$. More formally:

$$\begin{aligned} &\phi_{WA}(\mathbf{x}, \delta_{\epsilon_1}, ..., \delta_{\epsilon_{N_1}}, \hat{\epsilon}) \\ &= \begin{cases} (\frac{|\hat{\epsilon}-\epsilon^-|}{|\hat{\epsilon}-\epsilon^-|+|\hat{\epsilon}-\epsilon^+|})\delta_{\epsilon^-} + (\frac{|\hat{\epsilon}-\epsilon^+|}{|\hat{\epsilon}-\epsilon^-|+|\hat{\epsilon}-\epsilon^+|})\delta_{\epsilon^+}, \\ if\ \epsilon_1 < \hat{\epsilon} < \epsilon_{N_1} \\ \frac{\hat{\epsilon}}{\epsilon_1}\delta_{\epsilon_1},\ if\ \hat{\epsilon} \leq \epsilon_1 \\ \frac{\hat{\epsilon}}{\epsilon_{N_1}}\delta_{\epsilon_{N_1}},\ if\ \hat{\epsilon} \geq \epsilon_{N_1}. \end{cases} \end{aligned} \tag{6}$$

Where ϵ^- indicate the largest ϵ_i in the existing adversarial noises that is smaller than $\hat{\epsilon}$, ϵ^+ indicate the smallest ϵ_i in the existing adversarial noises that is larger than $\hat{\epsilon}$. With such design, with the transferability property of adversarial noises (Papernot, McDaniel, and Goodfellow 2016a), we can potentially remove the cost of training an AN-Estimator during the search.

Attention AN-Estimator. The Attention AN-Estimator (A-ANE) generates a new adversarial noise for input $\mathbf{x}$ by first obtaining a result $\dot{\delta}$ leveraging the WA-ANE, then using a proposed encoder with trainable weights, the A-ANE generate an "adjustment signal" $\ddot{\delta}$ to be added onto the WA-ANE output. More formally, we have

$$\begin{aligned}\phi_E(x, \delta_{\epsilon_1}, ..., \delta_{\epsilon_{N_1}}, \hat{\epsilon}) &= \dot{\delta} + \ddot{\delta} \\ &= \phi_{WA}(\mathbf{x}, \delta_{\epsilon_1}, ..., \delta_{\epsilon_{N_1}}, \hat{\epsilon}) + \\ &\quad F_A(x, \delta_{\epsilon_1}, ..., \delta_{\epsilon_{N_1}}, \hat{\epsilon}, \theta_A), \quad (7)\end{aligned}$$

where $F_A(\cdot)$ is the encoder with trainable weights θ_A. With such a design, we provide more flexibility for the AN-Estimator when the naive result $\dot{\delta}$ is not sufficient for performing an effective attack.

To model the sequential relationship within the adversarial noise sequence $\{\delta_{\epsilon_1}, ..., \delta_{\epsilon_{N_1}}\}$, we take inspiration from Neural Machine Translation (Bahdanau, Cho, and Bengio 2014), where the architecture proposed for $F_A(\cdot)$ is demonstrated in Fig.2. Within $F_A(\cdot)$, we include a bi-directional RNN module to model the adversarial noises in the form of a sequence and an attention mechanism to aggregate information in the sequence to form the new adversarial noise.

The hidden cells $F_f(\cdot)$ and $F_b(\cdot)$ in the **bi-directional RNN** are formed by Fully-Connected layers (FC-layer) and Convolution-Pooling-Reshape (CPR) modules. The hidden cells take in both the output hidden state from the previous hidden cell and an adversarial noise in the form of feature maps as inputs. To reduce computational costs, the normalized adversarial noise would be encoded into a noise encoding vector using the CPR module, and the norm of the adversarial noise would be encoded into a norm encoding vector using an FC-layer. The previous hidden state vector and the two newly encoded vectors would be concatenated to form vector $\mathbf{h_c}$ representing the combined information of the adversarial noise sequence until the current time step. By using another FC-layer, the output hidden-state vector can be encoded according to $\mathbf{h_c}$. The initial state of the bi-directional RNN is formed by encoding the natural input $\mathbf{x}$ into the form of a hidden-state vector using a CPR module.

Within the **attention mechanism**, we encode **query vectors** by extracting hidden states δ_{ϵ^-} and δ_{ϵ^-} from the forward and backward RNN sequence. To form $\tilde{\mathbf{h}}_q$ and $\mathbf{h_q}$ as in Fig.2(b), within F_b and F_f, we use the hidden-state vectors extracted as previous cell hidden-state so that information in the adversarial noise sequence near target strength $\hat{\epsilon}$ can be included in the query; we encode the norm encoding vector according to $\hat{\epsilon}$ to indicate the target strength to be generated in the query and we use zero vector as the noise encoding vector. With $\mathbf{h_c}$ in F_f and F_b formed, the query can be encoded. To form the **key vectors**, we directly concatenate the hidden-state vector pairs from the bi-directional RNN, since the hidden-state vector sequence is a compressed representation of information in the adversarial noise sequence. To form the **value feature maps**, we process the adversarial noise sequence using a shared convolutional module. With the query, keys, and values, the new adversarial noise can be aggregated by leveraging the attention mechanism.

Algorithm 1: Wsr-NAS algorithm

1: Split training data D into D_t and D_v
2: Initialize a warm-up architecture population: $P = \{\mathbf{H}_i | i = 1, ..., N\}$
3: **for** each $\mathbf{H}_i \in P$ **do**
4: Warm-up architecture $\mathbf{H}_i$ for 1 epoch
5: **end for**
6: Initialize memory M_v and M_a
7: **for** each $\mathbf{H}_i \in P$ **do**
8: Adversarially train arch. $\mathbf{H}_i$ on D_t for 1 epoch
9: Update M_a using $\mathbf{H}_i$ and B random samples from D_t
10: Train AN-Estimator over M_a for K_a iterations
11: **for** each $\mathbf{x}_j$ in D_v **do**
12: Generate $\{\delta_{\epsilon_1}, \delta_{\epsilon_2}, ..., \delta_{\epsilon_{N_1}}\}$ using $\mathbf{H}_i$
13: Generate $\{\hat{\delta}_{\hat{\epsilon}_1}, \hat{\delta}_{\hat{\epsilon}_2}, ..., \hat{\delta}_{\hat{\epsilon}_{N_2}}\}$ using the AN-Estimator
14: Evaluate $\mathbf{H}_i$ over $\{\mathbf{x}, ..., \mathbf{x} + \delta_{\epsilon_{N_1}}, ..., \mathbf{x} + \hat{\delta}_{\epsilon_{N_2}}\}$
15: **end for**
16: Update M_v according to evaluation results over D_v
17: **end for**
18: Train VLE in EWSS over M_v for K_v times
19: **for** t = 1 $\rightarrow$ T **do**
20: Sample an architecture with $\mathbf{H}_i$
21: Adversarially train arch. $\mathbf{H}_i$ on D_t for 1 epoch
22: Update M_a, AN-Estimator and M_v as in line 9 - 16
23: Update VLE in EWSS over M_v
24: Update $\mathbf{A}$ using the EWSS
25: **end for**

With such architectural design, we gain three advantages. (1) Instead of processing each adversarial noise separately, the bi-directional RNN architecture allows the A-ANE to leverage the sequential information in the adversarial noise sequence. (2) The attention mechanism allows information across the entire adversarial noise sequence to be freely extracted according to the target adversarial noise strength $\hat{\epsilon}$. (3) The raw adversarial noise sequence can be processed and filtered before being aggregated as Values by the attention mechanism to form $\ddot{\delta}$.

Encoding AN-Estimator. Similar to the A-ANE, the Encoding AN-Estimator (E-ANE) has a WA-ANE branch and an encoding branch with trainable weights, where the encoding branch of the E-ANE is also formed by a bi-directional RNN and an attention mechanism. However, within the E-ANE, in the bi-directional RNN, instead of processing the hidden states in the form of vectors, the hidden states are in the form of feature maps. To allow this configuration, the hidden cells are formed by convolutional layers. The query vector is encoded from the hidden states with strength neighboring $\hat{\epsilon}$ in a pipeline similar to encoding the query vectors in the A-ANE; the key vectors and the value feature maps are encoded from the hidden-state feature map sequence.

The potential advantage of the E-ANE design is that we allow more processing before forming the Values, potentially allowing a better quality for the value feature maps.

Empirical Result. Across the three designs, we find empirically that the A-ANE design has the best performance, potentially due to good compatibility between the A-ANE architecture and the task of generating adversarial noise based on existing noises given limited estimator capacity.

NAS Type	Model Name	Train Type	Params (M)	Natural (%)	Avg. Rob. (%)	FGSM Acc. (%)
Standard	WRN-34-10 (Zagoruyko and Komodakis 2017)	Standard	46.2	95.01	-	0
Adv. Train	ResNet-50*(He et al. 2015)	Multi-Adv	23.5	80.2	14.8	-
	ResNet-18(He et al. 2015)	PGD-7	11.2	78.4	-	49.8
	ResNet-50(He et al. 2015)	PGD-7	23.5	79.2	-	53.6
	WRN-34-10* (Zagoruyko and Komodakis 2017)	TRADES-1	46.2	88.6	20.8	-
Standard NAS	AmoebaNet(Real et al. 2019)	PGD-7	3.2	83.4	-	56.4
	PVLL*(Li et al. 2020)	PGD-7	3.5	81.3	18.6	-
	NAS-Net*(Zoph et al. 2018)	TRADES-1	3.3	91.8	17.4	-
	DARTS* (Liu, Simonyan, and Yang 2018)	TRADES-1	3.2	91.5	14.4	-
Robust NAS	RobNet-S(Guo et al. 2020)	PGD-7	4.4	78.1	-	53.9
	RobNet-M(Guo et al. 2020)	PGD-7	5.7	78.3	-	54.6
	RobNet-L(Guo et al. 2020)	PGD-7	6.9	78.6	13.8	55.0
	RACL*(Dong et al. 2020)	PGD-7	3.6	80.7	23.4	57.7
	AdvRush*(Mok et al. 2021)	PGD-7	4.2	82.0	21.7	-
	RobNet-S(Guo et al. 2020)	TRADES-1	4.4	90.5	15.8	-
	RACL*(Dong et al. 2020)	TRADES-1	3.6	91.0	15.4	-
	AdvRush*(Mok et al. 2021)	TRADES-1	4.2	91.6	15.8	-
	E2R-NAS*(Yue et al. 2020)	TRADES-1	4.0	90.1	18.0	-
	NAS-OOD*(Bai et al. 2021)	TRADES-1	4.4	91.2	21.3	-
Wsr-NAS (Naive)	WsrNet-Naive-1	TRADES-1	4.4	89.6	21.7	-
	WsrNet-Naive-3	TRADES-1	4.5	89.6	22.1	-
	WsrNet-Naive-6	TRADES-1	4.4	91.2	22.9	-
Wsr-NAS (Full)	WsrNet-Basic	PGD-7	4.5	80.8	24.8	62.2
	WsrNet-Robust	PGD-7	4.5	78.3	27.9	67.2
	WsrNet-Plus	PGD-7	4.7	80.7	25.7	-
	WsrNet-Basic	TRADES-1	4.5	90.4	22.4	-
	WsrNet-Plus	TRADES-1	4.7	91.3	24.2	-

Table 1: Comparisons between WsrNets and existing baselines. * indicates recreated results.

Efficient Wide Spectrum Searcher

In the process of searching for WsrNets, in the case of naively applying the One-Shot-NAS framework, a major source of time consumption is the process of calculating gradient to update architecture weights $\mathbf{H}$ over the validation set formed by not only normal inputs but also N_1+N_2 times more adversarial validation data. To address such a limitation, we propose an **Efficient Wide Spectrum Searcher (EWSS)**. as in Fig.2(c), the EWSS is formed by two components, a Validation Loss Estimator (VLE) formed by an RNN architecture encoder combined with a multi-head output layer to estimate the adversarial and non-adversarial validation loss based on architecture weight $\mathbf{H}$ and a **Robust Search Objective** designed to allow proper balancing between clean accuracy and robustness at different adversarial noise strengths. With the EWSS, when given an architecture weight $\mathbf{H}$ and appropriate configurations in the Robust Search Objective, gradient direction to update the architecture weight $\mathbf{H}$ can be calculated, significantly reducing the cost of updating architecture weight during the search.

More formally, we describe the VLE in EWSS as:

$$\hat{\mathbf{L}} = \{\hat{L}_{natural}, \hat{L}_{\epsilon_1}, ..., \hat{L}_{\epsilon_{N_1+N_2}}\} = \psi(\mathbf{H}). \tag{8}$$

Within Eqn.8, $\hat{\mathbf{L}} \in \mathbb{R}^{N_1+N_2+1}$ is the set of estimates on the natural validation loss and adversarial validation loss on $N_1 + N_2$ different adversarial noise strengths for network obtained basing on architecture weight $\mathbf{H}$; the VLE $\psi(\cdot)$ is formed by an RNN as architecture encoder followed by a dense layer with $n + 1$ outputs. To store training data for the VLE, a memory $M_v = \{(\mathbf{H}_t, \mathbf{L}_t), \forall 1 \leq t \leq T\}$ with size T is included (for a reason similar to the memory in the AN-Estimator). To train the VLE, during the search, we minimize the mean square error (MSE) loss defined as

$$L_v(\psi) = \frac{1}{T}\sum_{t=1}^{T} \|\psi(\mathbf{H}_t) - \mathbf{L}_t\|_2^2 \tag{9}$$

over the memory M_v, where t represents the index in the memory, L_t is the validation loss values evaluated on $\mathbf{H}_t$.

Leveraging the VLE defined, we propose the Robust Search Objective in EWSS as:

$$\begin{aligned} &\min_A \alpha\hat{\mathbf{L}}_{natural} + \beta\sum_{i=1}^{n}\beta_i\hat{\mathbf{L}}_{\epsilon_i} \\ &s.t.\ \mathbf{H} = GumbelSoftmax(\mathbf{A}, \xi, \tau), \\ &\textstyle\sum_{i=1}^{n}\beta_i = 1, \alpha+\beta = 1, \alpha > 0, \beta > 0, \beta_i > 0\ \forall i \end{aligned} \tag{10}$$

where ξ is an i.i.d sample from Gumbel(0,1) and τ is temperature. With the value of α, β and different β_i selected, the

Model	Train Type	Para. (M)	Acc. (%) 0.03	Acc. (%) 0.1	Acc. (%) 0.17	Acc. (%) 0.25
WRN-34 -10	S	46.2	0	-	-	-
ResNet-50	MA	23.5	37.0	12.5	6.3	3.5
ResNet-18	P7	11.2	45.6	-	-	-
ResNet-50	P7	23.5	45.8	-	-	-
WRN-34 -10*	T1	46.2	46.1	18.1	11.2	7.9
Amoeba -Net	P7	3.2	39.5	-	-	-
PVLL*	P7	3.5	47.4	15.7	7.3	3.8
NAS-Net*	T1	3.3	49.1	15.5	4.4	0.5
DARTS*	T1	3.2	46.9	9.3	1.23	0.1
RobNet-S	P7	4.4	48.3	-	-	-
RobNet-M	P7	5.7	49.1	-	-	-
RobNet-L	P7	6.9	49.4	3.1	1.8	0.7
RACL*	P7	3.6	47.9	22.4	14.0	9.4
AdvRush*	P7	4.2	48.7	22.3	10.8	4.9
RobNet-S	T1	4.4	46.9	11.6	4.11	1.27
RACL*	T1	3.6	46.9	11.6	2.8	0.3
AdvRush*	T1	4.2	48.2	11.8	3.1	0.3
E2R -NAS*	T1	4.0	40.5	14.5	10.2	6.9
NAS -OOD*	T1	4.4	50.3	20.6	10.5	3.8
WsrNet-N1	T1	4.4	45.8	19.1	13.3	8.6
WsrNet-N3	T1	4.5	45.9	20.3	13.3	8.7
WsrNet-N6	T1	4.4	46.3	20.1	15.6	9.41
WsrNet-B	P7	4.5	48.9	22.3	15.7	12.4
WsrNet-R	P7	4.5	48.4	27.2	20.3	15.5
WsrNet-P	P7	4.7	48.9	23.1	17.0	13.7
WsrNet-B	T1	4.5	45.9	20.0	14.1	9.4
WsrNet-P	T1	4.7	47.5	22.1	16.0	11.3

Table 2: Model performances at different adversarial noise strengths. (S: Standard, MA: Multi-Adv, PX: PGD-X, TX: TRADES-X, "-B": -Basic, "-R": -Robust, "-P": -Plus), "NX": -Naive-X

architecture weight $\mathbf{A}$ in the super-network can be adjusted according to the gradient of Eqn. 10 by $\mathbf{A}^{'} \leftarrow \mathbf{A} + c\nabla_{\mathbf{A}}$.

Within Eqn. 10, by having the α and β terms, we allow customized priority over accuracy and overall robustness. With different β_i being summed up to 1, by adjusting certain β_i to be larger relative to others, the clean accuracy of the WsrNet found would not be impacted while increasing robustness at the corresponding adversarial noise strength.

Search Procedure

The algorithm for Wsr-NAS is demonstrated in Algorithm. 1. The search is split into two phases, the warm-up phase and the search phase. In the warm-up phase, the super-network is first warmed-up (lines 3 - 5). Then, in lines 7 - 17, we simultaneously warm up the AN-Estimator and the VLE in EWSS. With the warm-up phase, we allow the EWSS to be equipped with prior knowledge on the validation loss landscape of natural images, normal adversarial examples and adversarial examples generated by the AN-Estimator before the search phase. In the search phase, at each step, we first update the network architecture weight using the EWSS, followed by training for the VLE and AN-Estimator. Such configuration allows the EWSS and the AN-Estimator to be updated as the super-network changes during the search.

Within the search, by applying the AN-Estimator and the EWSS, by reducing the cost of generating adversarial noises and avoiding the cost of naively performing the search on the adversarial validation set, we perform Wsr-NAS efficiently.

Search Config	WsrNet -Basic	WsrNet -Plus	WsrNet -Naive-1
N_1, N_2	3,3	3,8	1,0
Total Time	3.6 days	4.0 days	2.6 days
Search Config	WsrNet -Naive-3	WsrNet -Naive-6	WsrNet -Naive-11
N_1, N_2	3,0	6,0	11,0
Total Time	3.4 days	5.2 days	7.4 days*
Search Config	RobNet	MetaQNN	AmobaNet
N_1, N_2	1,-	-,-	-,-
Total Time	22.6 days*	450 × 7 days	10 × 8 days*

Table 3: total GPU time for different search algorithms or search configurations. $*$ indicates estimated time.

Experiments

Within this research, we perform Wsr-NAS on CIFAR-10 only, where the discovered architectures are tested on CIFAR-10 and ImageNet. We let dataset D_t and D_v to be different halves of the CIFAR-10 training set, let M_a be 600, let K_a be 1 and let B in the Wsr-NAS algorithm to be 10. To generate $N_1 + N_2$ adversarial noises at different strengths, we first divide the strengths into N_1 intervals, each interval contains $(N_1 + N_2)/N_1$ adversarial noise strengths. Within each interval, the adversarial noise corresponding to the first strength is generated normally, while the remaining noises are generated with the AN-Estimator. To find WsrNet using a moderate number or a large number of adversarial noise strengths, we search for WsrNet-Basic and WsrNet-Plus by setting (N_1, N_2) to be (3,3) and (3,8) respectively. α and β are set as 0.8 and 0.2 by default. A-ANE has been used for all search experiments except when searching for WsrNet-Plus, where a modified AN-Estimator has been used to generate adversarial noises at 8 strengths. Within all experiments, the step size of the PGD attack would be set to $\epsilon_i \times 2.5/20$ at adversarial noise strength ϵ_i as in (Madry et al. 2019). Also, we used L_{inf} norm bound for limiting adversarial noise strengths within the main paper.

Main Results

To illustrate the performance of WsrNets found by Wsr-NAS, we search for WsrNets under different configurations and compare their retraining results with the retrain-

Model Name	Params	Nat. Acc. (%)	Rob. Acc. (%)	Acc. ϵ_1 (%)	Acc. ϵ_3 (%)	Acc. ϵ_5 (%)
ResNet50	23.5M	55.5	-	30.3	-	-
RACL	12.0M	56.1	8.5	30.3	4.9	1.1
AdvRush	12.2M	31.5	4.1	15.4	2.1	0.2
WsrNet -Basic	13.1M	**57.2**	**9.0**	**31.0**	**5.3**	**1.3**

Table 4: Generalizing different Robust-NAS architectures to ImageNet. Tained under Fast Adversarial Training(Wong, Rice, and Kolter 2020), $\epsilon_1 = 0.015, \epsilon_3 = 0.045, \epsilon_5 = 0.075$.

Avg. Nat. Acc.	Avg. Rob. Acc.	A -ANE	WA -ANE	E -ANE$^-$	E -ANE
73.0%	38.7%	43.6%	67.4%	71.1%	48.9%

Table 5: Attack performance of different AN-Estimators.

Case	Adv. Train (s)	A.E. Gen. (s)	Search (s)	EWSS (s)	AN-E Train Test(s)
Basic	610.6	1151.5	191.3	-	-
Wsr -NAS	610.6	576.3	-	37.1	27.6

Table 6: Average GPU time required for different parts in a single search loop iteration of Wsr-NAS (on CIFAR-10).

ing results of various existing baseline models in Tab.1 and Tab.2. To perform comparisons in average adversarial robustness, we retrained the models leveraging either PGD-7 or TRADES. Within Table 1 and 2, to evaluate the overall robustness of a model, we calculate the average accuracy of the model across four different adversarial noise strengths across a wide range ($\epsilon = 0.03, 0.10, 0.17, 0.25$).

To compare the performance of WsrNet-Basic and WsrNet-Plus with models found using existing robust-NAS, we train WsrNet-Basic, WsrNet-Plus, RobNet-small, RACL, AdvRush E2R-NAS and NAS-OOD under TRADES-1 (Zhang et al. 2019). As in Tab.1, the WsrNet-Basic and WsrNet-Plus model has a clean accuracy of 90.4% and 91.3% respective, which is comparable to the five existing models. At the same time, WsrNet-Basic and WsrNet-Plus have a large improvement on overall adversarial robustness (22.4% and 24.2%) relative to the five existing models (15.8%, 15.4%, 15.8% 18.0% and 21.3%). When the WsrNet-Basic, WsrNet-Plus and the different existing models are trained on PGD-7, we can also find a similar result. Note that on PGD-7, the retraining hyperparameters of RACL, AdvRush, WsrNet-Basic, WsrNet-Robust and WsrNet-Plus have been tuned to trade-off clean accuracy and average robust accuracy for easier comparisons. The WsrNet-Robust is WsrNet-Basic with hyper-parameter tuned so that it gains a lower 78.3% clean accuracy but better average robust accuracy of 27.9% for comparison with RobNet-L at the same clean accuracy.

To compare the performance of WsrNets with robust models gained solely relying on single strength adversarial training, we also compare WsrNet-Basic with WRN-34-10, where both models are trained under TRADES-1. Although having a parameter count more than 10 times smaller than WRN-34-10, the WsrNet-Basic gained a 1.8% higher clean accuracy and 1.6% higher overall robustness.

To compare the performance of WsrNet-Basic with multi-strength adversarial training, we train ResNet-50 using PGD-7 (Madry et al. 2019). In detail, for each batch of natural images obtained during training (with batch size 64), we generate adversarial examples for the batch at $\epsilon = \{0.03, 0.045, 0.06, 0.075, 0.09, 0.105\}$. By combining the 6 batches of adversarial examples generated as a new batch and training the ResNet-50 using the generated batch, we achieve multi-strength adversarial training. As in Tab.1, we found the resulting ResNet-50 model has a clean accuracy of 80.2% and an average robust accuracy of 14.8%, being 0.5% lower and 10.0% lower than WsrNet-Basic respectively.

AN-Estimator Ablation Study

Evaluation of Wsr-NAS Algorithm

Searching Under More Adversarial Noise Strengths. As in Tab.1, in WsrNet-1, 3, 6, by increasing the number of adversarial noise strengths from 1 to 6, we were able to gain an average robustness increase from 21.7% to 22.9%. Moreover, on TRADES-1 retraining, by comparing the WsrNet-Basic searched with 6 adversarial noise strengths with WsrNet-Plus searched with 11 adversarial noise strengths, we find that WsrNet-Plus was able to gain a 24.2% average robust accuracy, being 1.8% higher than WsrNet-Basic. With the results, we confirm that a richer search signal allows WsrNet with better overall robustness to be searched.

As in Tab.3, for Msr-NAS applied with the AN-Estimator (MsrNet-Basic, Plus), by increasing the number of adversarial noise strengths from 6 to 11, the search time increased by 0.4 days, which indicates that the time increase required per adversarial noise strength increase is 0.08 days. For Msr-NAS without applying the AN-Estimator (MsrNet-1,3,6,11), by increasing adversarial noise strength from 1 to 11, the search time increased by 4.8 days. Hence, the time increase required per adversarial noise strength increase is 0.48 days. Such a result confirms that the AN-Estimator indeed allows the Msr-Net to scale more efficiently. With such a capability, relative to WsrNet-Naive-6, WsrNet-Plus can achieve a better performance of 91.3% clean accuracy and 24.2% average robust accuracy (0.1% and 1.3% higher), while having a search time reduction of 1.2 days.

Speed Analysis. To perform search time comparisons with existing NAS techniques, we have included the search time required for Msr-NAS and multiple existing NAS techniques in Tab.3. For RobNet(Guo et al. 2020), the algorithm relies on random sampling for discovering new architectures, where the search time required is estimated to be 22.6 days. For MetaQNN(Baker et al. 2017), Reinforcement Learning

has been leveraged to perform the search, where searching for clean accuracy alone requires 7 days on 450 GPUs. Search time would further increase if adversarial examples need to be generated. Similarly, for AmobaNet(Real et al. 2019), to search for clean accuracy leveraging Evolutionary Algorithms, 8 days would be required on 10 GPUs. Compared with the Msr-NAS, which requires 3.6 to 4.0 GPU days to perform NAS under multiple adversarial noise strengths, the Msr-NAS is more efficient in time.

To evaluate the contribution of different design components in reducing search time, we include Tab.6. Within Alg.1, for the setting of searching under 6 adversarial noise strengths, within the second loop (line 7 - 18) and the third loop (line 19 and 24), for a single iteration of search, without the AN-Estimator and the EWSS, the average total time required are 1953.4s (610.6s + 1511.5s + 191.3s). However, by leveraging the AN-Estimator and the EWSS, the average time required for a single iteration in the loops would be reduced to 1251s (610.6s + 576.3s + 37.1s + 27.6s). In detail, the time of performing adversarial training under PGD-7 over D_t for one epoch remains unchanged (610.3s), but the average time required to generate adversarial example on the sampled networks has reduced to 575s, since adversarial noises are required to be generated using the network H only on three adversarial noise strengths. For the AN-Estimator, generating the adversarial noises for training over the memory M_a and performing inference over the validation set D_v requires only 27.6s. The total time required by the EWSS is 37.1s, where 27.2s is consumed for generating training data for the EWSS and 9.9s is consumed (when between lines 20 - 25 of Alg.1) for training the EWSS and using it to update the architecture weights.

Generalization. To demonstrate the robustness obtained through Wsr-NAS is not restricted to being effective only for PGD attacks, we trained WsrNet-Basic using PGD-7 and evaluated its performance under FGSM. As in Tab. 1, by being attacked by the FGSM, the WsrNet-Basic gained an accuracy of 62.2%, being 4.5% higher than the second best model RACL, indicating that the model architecture found generalizes across attacks. More attack generalization results (e.g. Black-Box attack (Papernot, McDaniel, and Goodfellow 2016a), AutoAttack (Croce and Hein 2020)) have been included in the Appendix.

To demonstrate the WsrNets searched on CIFAR-10 generalizes across datasets, we retrain WsrNet-Basic on ImageNet to compare with various baseline models. As in Tab. 4, on the ImageNet dataset, the WsrNet-Basic model has gained a clean accuracy of 57.2% and average robust accuracy of 9.0% (average robust accuracy tested on adversarial noise strengths $\{0.015,0.03,0.045,0.06,0.075,0.09\}$) which is consistently higher than RACL, AdvRush and ResNet-50.

In this section, the performance of AN-Estimators is compared under a lightweight constraint, so that designs more suitable for the needs of Wsr-NAS can be discovered.

To perform the comparison, on CIFAR-10, we pre-trained 7 different networks with different architectures and a different number of epochs to simulate the different networks obtained from the super-network in Wsr-NAS. On the training set of CIFAR-10, we generate training data for the AN-Estimator, where the network used to generate the adversarial noises is changed every 10 batches. Each version of the AN-Estimator is trained on the training data for 1 epoch. To evaluate the performance of different AN-Estimators, on the CIFAR-10 test set, using the AN-Estimators trained, we generate adversarial noises on strengths $\epsilon = 0.03$, $\epsilon = 0.06$ and $\epsilon = 0.09$ to attack the 7 pre-trained models.

As in Tab.5, the average test accuracy of the pre-trained networks is 73.0%, whereas the average robust accuracy of the pre-trained networks is 38.7%. With naive averaging, the adversarial noise generated by the WA-ANE can only reduce the performance of the pre-trained models by 5.6%, indicating the design to be ineffective. The E-ANE$^-$ model included is E-ANE without the attention stage, where the Query (feature maps) obtained from encoding the bi-directional RNN hidden states are used as the output of the AN-Estimator. By comparing the performance of 71.1% for E-ANE$^-$ with the performance of 48.9% for E-ANE, we can conclude that the attention mechanism included in E-ANE is essential for the AN-Estimator to gain good performance. With direct access to the entire hidden state sequence, better encoding can be obtained. The performance of the A-ANE is 43.6%, outperforming the E-ANE by 5.3%, indicating that in E-ANE, by having limited capacity, the bi-directional RNN has potentially over-compressed the adversarial noise sequence, causing the Value feature maps encoded from the RNN hidden-states to contain fewer useful information.

Conclusions

In our research, we proposed the Wsr-NAS algorithm that finds WsrNets with improved robustness over a wide range of adversarial noise strengths while maintaining sufficiently high clean accuracy. With extensive experiments, we find the WsrNets found to generalize across different datasets, attack types and training techniques. By proposing a lightweight AN-Estimator that is trained to efficiently generate adversarial noises during the search and an Efficient Wide Spectrum Searcher, considerable time reduction has been obtained.

Limitations A limitation of our research is that, although our method can find architectures with improved overall robustness, but at the small adversarial noise strength of 0.03, the robust accuracy of WsrNet can be 1% or 2% lower than existing SOTA models. We find overcoming this limitation to be a potential future work of the research.

Acknowledgements

This work was supported in part by the Australian Research Council under Project DP210101859 and the University of Sydney Research Accelerator (SOAR) Prize. The authors acknowledge the use of the National Computational Infrastructure (NCI) which is supported by the Australian Government, and accessed through the NCI Adapter Scheme and Sydney Informatics Hub HPC Allocation Scheme.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bai, H.; Zhou, F.; Hong, L.; Ye, N.; Chan, S. H. G.; and Li, Z. 2021. NAS-OoD: Neural Architecture Search for Out-of-Distribution Generalization. arXiv:2109.02038.

Baker, B.; Gupta, O.; Naik, N.; and Raskar, R. 2017. Designing Neural Network Architectures using Reinforcement Learning. In *International Conference on Learning Representations*.

Cai, H.; Chen, T.; Zhang, W.; Yu, Y.; and Wang, J. 2017. Reinforcement Learning for Architecture Search by Network Transformation. *CoRR*, abs/1707.04873.

Cohen, J.; Rosenfeld, E.; and Kolter, Z. 2019. Certified Adversarial Robustness via Randomized Smoothing. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 1310–1320. PMLR.

Croce, F.; and Hein, M. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *CoRR*, abs/2003.01690.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255.

Dong, M.; Li, Y.; Wang, Y.; and Xu, C. 2020. Adversarially Robust Neural Architectures. arXiv:2009.00902.

Dong, Y.; Liao, F.; Pang, T.; Su, H.; Zhu, J.; Hu, X.; and Li, J. 2018. Boosting Adversarial Attacks with Momentum. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9185–9193.

Du, X.; Zhang, J.; Han, B.; Liu, T.; Rong, Y.; Niu, G.; Huang, J.; and Sugiyama, M. 2021. Learning Diverse-Structured Networks for Adversarial Robustness. arXiv:2102.01886.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and Harnessing Adversarial Examples. arXiv:1412.6572.

Guo, M.; Yang, Y.; Xu, R.; Liu, Z.; and Lin, D. 2020. When NAS Meets Robustness: In Search of Robust Architectures Against Adversarial Attacks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385.

Hosseini, R.; Yang, X.; and Xie, P. 2021. DSRNA: Differentiable Search of Robust Neural Architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 6196–6205.

Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. Cite arxiv:1704.04861.

Krizhevsky, A.; and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario.

Lécuyer, M.; Atlidakis, V.; Geambasu, R.; Hsu, D.; and Jana, S. 2019. Certified Robustness to Adversarial Examples with Differential Privacy. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, 656–672. IEEE.

Li, B.; Chen, C.; Wang, W.; and Carin, L. 2019. Certified Adversarial Robustness with Additive Noise. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Li, Y.; Dong, M.; Wang, Y.; and Xu, C. 2020. Neural Architecture Search in A Proxy Validation Loss Landscape. In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 5853–5862. PMLR.

Liu, H.; Simonyan, K.; and Yang, Y. 2018. DARTS: Differentiable Architecture Search. Cite arxiv:1806.09055.

Lütjens, B.; Everett, M.; and How, J. P. 2019. Certified Adversarial Robustness for Deep Reinforcement Learning. *CoRR*, abs/1910.12908.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2019. Towards Deep Learning Models Resistant to Adversarial Attacks. arXiv:1706.06083.

Mok, J.; Na, B.; Choe, H.; and Yoon, S. 2021. AdvRush: Searching for Adversarially Robust Neural Architectures. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 12322–12332.

Papernot, N.; McDaniel, P.; and Goodfellow, I. 2016a. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*.

Papernot, N.; McDaniel, P.; and Goodfellow, I. 2016b. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples. arXiv:1605.07277.

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2019. Regularized Evolution for Image Classifier Architecture Search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01): 4780–4789.

Shafahi, A.; Najibi, M.; Ghiasi, M. A.; Xu, Z.; Dickerson, J.; Studer, C.; Davis, L. S.; Taylor, G.; and Goldstein, T. 2019. Adversarial training for free! In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Song, C.; Cheng, H.-P.; Yang, H.; Li, S.; Wu, C.; Wu, Q.; Li, H.; and Chen, Y. 2018. MAT: A Multi-strength Adversarial Training Method to Mitigate Adversarial Attacks. arXiv:1705.09764.

Sun, J.; Yao, W.; Jiang, T.; Li, C.; and Chen, X. 2023. A^3D: A Platform of Searching for Robust Neural Architectures and Efficient Adversarial Attacks. arXiv:2203.03128.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the Inception Architecture for Computer Vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2818–2826.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2014. Intriguing properties of neural networks. arXiv:1312.6199.

Wong, E.; Rice, L.; and Kolter, J. Z. 2020. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*.

Xu, Y.; Xie, L.; Zhang, X.; Chen, X.; Qi, G.-J.; Tian, Q.; and Xiong, H. 2020. PC-DARTS: Partial Channel Connections for Memory-Efficient Architecture Search. arXiv:1907.05737.

Yue, Z.; Lin, B.; Huang, X.; and Zhang, Y. 2020. Effective, Efficient and Robust Neural Architecture Search. arXiv:2011.09820.

Zagoruyko, S.; and Komodakis, N. 2017. Wide Residual Networks. arXiv:1605.07146.

Zhang, H.; Yu, Y.; Jiao, J.; Xing, E.; Ghaoui, L. E.; and Jordan, M. 2019. Theoretically Principled Trade-off between Robustness and Accuracy. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 7472–7482. PMLR.

Zhang, J.; Xu, X.; Han, B.; Niu, G.; Cui, L.; Sugiyama, M.; and Kankanhalli, M. S. 2020. Attacks Which Do Not Kill Training Make Adversarial Learning Stronger. *CoRR*, abs/2002.11242.

Zoph, B.; Vasudevan, V.; Shlens, J.; and Le, Q. V. 2018. Learning Transferable Architectures for Scalable Image Recognition. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8697–8710.