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; 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; Dong et al. 2018; 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).

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

Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.
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 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 across 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 $\delta$ with strength smaller than $\epsilon$ on a dataset $D$, so that when the noise is added on input $x$ (with labels $y$), the Neural Network under attack (represented by $\theta$) would output a prediction desired by the adversary (through maximizing loss $L$ set by the adversary). More formally, the attack model is defined as:

$$\delta = \underset{\delta, x, t}{\text{argmax}} \|\delta\| < \epsilon, L(\theta, x + \delta, y).$$

(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; Song et al. 2018; Madry et al. 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 propose 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 $\{I^{(i)}|0 \leq i \leq 6\}$ (each node represents a set of hidden feature maps). Within the 7 nodes, $I^{(0)}$ and $I^{(1)}$ are outputs from the two previous cells,
node $I^{(2)}$ to node $I^{(5)}$ are calculated by:

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

(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 $I^{(i)}$ to $I^{(j)}$. $h_{i,j}^{(k)} \in \mathbf{H}$ 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 $I^{(6)} = \bigcup_{i=2}^{N} I^{(i)}$. 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 input $x$ we aim to find $\hat{x}$, adversarial noise that is smaller than $\epsilon$ and 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:

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

(6)

Where $\epsilon^-$ indicates the largest $\epsilon_i$ in the existing adversarial noises that is smaller than $\hat{\epsilon}$, $\epsilon^+$ indicates the smallest $\epsilon_i$ 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.

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

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

(5)

To reduce computational cost when searching for Wsr-Nets, 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.

**Figure 2:** (a) Architecture of hidden cells ($F_h$ or $F_r$) in the AN-Estimator. (b) Architecture of the AN-Estimator. (c) Illustration of Efficient Wide Spectrum Searcher. Given different $\alpha, \beta$ and $\beta_r$, architectures can be updated accordingly.
Attention AN-Estimator. The Attention AN-Estimator (A-ANE) generates a new adversarial noise for input \( x \) by first obtaining a result \( \delta \) leveraging the WA-ANE, then using a proposed encoder with trainable weights, the A-ANE generates an “adjustment signal” \( \hat{\delta} \) to be added onto the WA-ANE output. More formally, we have

\[
\phi_E(x, \delta_{\epsilon_1}, ..., \delta_{\epsilon_{N_1}}, \hat{\delta}) = \delta + \hat{\delta} = \phi_W A(x, \delta_{\epsilon_1}, ..., \delta_{\epsilon_{N_1}}, \hat{\delta}) + F_A(x, \delta_{\epsilon_1}, ..., \delta_{\epsilon_{N_1}}, \hat{\delta}, \theta_A),
\]

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

To model the sequential relationship within the adversarial noise sequence \( \{\delta_{i_1}, ..., \delta_{i_{N_1}}\} \), 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 \( h_i \) 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 \( h_i \). The initial state of the bi-directional RNN is formed by encoding the natural input \( x \) into the form of a hidden-state vector using a CPR module.

Within the attention mechanism, we encode query vectors by extracting hidden states \( \delta_{\epsilon} \) and \( \delta_{\epsilon} \) from the forward and backward RNN sequence. To form \( h_i \) and \( h_{\epsilon} \), we use the hidden-state vectors extracted as previous cell hidden-state so that information in the adversarial noise sequence near target strength \( \epsilon \) can be included in the query; we encode the norm encoding vector according to \( \epsilon \) to indicate the target strength to be generated in the query and we use zero vector as the noise encoding vector. With \( h_{\epsilon} \) 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.

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 \( \epsilon \). (3) The raw adversarial noise sequence can be processed and filtered before being aggregated as Values by the attention mechanism to form \( \hat{\delta} \).

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 \( \epsilon \) 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.

Algorithm 1: Wsr-NAS algorithm

1: Split training data \( D \) into \( D_t \) and \( D_v \)
2: Initialize a warm-up architecture population:
\[ P = \{ H_{i} | i = 1, ..., N \} \]
3: for each \( H_{i} \) \( \in P \) do
4: Warm-up architecture \( H_{i} \) for 1 epoch
end for
5: Initialize memory \( M_{v} \) and \( M_{a} \)
6: for each \( H_{i} \) \( \in P \) do
7: Adversarially train arch. \( H_{i} \) on \( D_t \) for 1 epoch
8: Update \( M_{a} \) using \( H_{i} \) and \( B \) random samples from \( D_t \)
9: Train AN-Estimator over \( M_{a} \) for \( K_{a} \) iterations
end for
10: for each \( x_{i} \) in \( D_v \) do
11: Generate \( \{ \delta_{i_1}, \delta_{i_2}, ..., \delta_{i_{N_1}} \} \) using \( H_{i} \)
12: Generate \( \{ \delta_{i_1}, \delta_{i_2}, ..., \delta_{i_{N_2}} \} \) using the AN-Estimator
13: Evaluate \( H_{i} \) over \( \{ x, ..., x + \delta_{i_{N_1}}, ..., x + \delta_{i_{N_2}} \} \)
end for
14: Update \( M_{v} \) according to evaluation results over \( D_v \)
end for
15: Train VLE in EWSS over \( M_{v} \) for \( K_{v} \) times
16: for \( t = 1 \rightarrow T \) do
17: Sample an architecture with \( H_{i} \)
18: Adversarially train arch. \( H_{i} \) on \( D_t \) for 1 epoch
19: Update \( M_{a} \), AN-Estimator and \( M_{v} \) as in line 9 - 16
20: Update VLE in EWSS over \( M_{v} \)
21: Update a using the EWSS
end for
<table>
<thead>
<tr>
<th>NAS Type</th>
<th>Model Name</th>
<th>Train Type</th>
<th>Params (M)</th>
<th>Natural (%)</th>
<th>Avg. Rob. (%)</th>
<th>FGSM Acc. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>WRN-34-10* (Zagoruyko and Komodakis 2017)</td>
<td>Standard</td>
<td>46.2</td>
<td>95.01</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>Adv. Train</td>
<td>ResNet-50*(He et al. 2015)</td>
<td>Multi-Adv</td>
<td>23.5</td>
<td>20.2</td>
<td>14.8</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>ResNet-18(He et al. 2015)</td>
<td>PGD-7</td>
<td>11.2</td>
<td>20.2</td>
<td>78.4</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>ResNet-30(He et al. 2015)</td>
<td>PGD-7</td>
<td>23.5</td>
<td>20.2</td>
<td>79.2</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>WRN-34-10* (Zagoruyko and Komodakis 2017)</td>
<td>TRADES-1</td>
<td>46.2</td>
<td>88.6</td>
<td>20.8</td>
<td>-</td>
</tr>
<tr>
<td>Standard NAS</td>
<td>AmoebaNet(Real et al. 2019)</td>
<td>PGD-7</td>
<td>3.2</td>
<td>20.2</td>
<td>83.4</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>PVLL*(Li et al. 2020)</td>
<td>PGD-7</td>
<td>3.5</td>
<td>20.2</td>
<td>81.3</td>
<td>18.6</td>
</tr>
<tr>
<td></td>
<td>NAS-Net*(Zoph et al. 2018)</td>
<td>TRADES-1</td>
<td>3.3</td>
<td>20.2</td>
<td>91.8</td>
<td>17.4</td>
</tr>
<tr>
<td></td>
<td>DARTS* (Liu, Simonyan, and Yang 2018)</td>
<td>TRADES-1</td>
<td>3.2</td>
<td>20.2</td>
<td>91.5</td>
<td>14.4</td>
</tr>
<tr>
<td>Robust NAS</td>
<td>RobNet-Sr(Guo et al. 2020)</td>
<td>PGD-7</td>
<td>4.4</td>
<td>20.2</td>
<td>78.1</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>RobNet-M(Guo et al. 2020)</td>
<td>PGD-7</td>
<td>5.7</td>
<td>20.2</td>
<td>78.3</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>RobNet-L(Guo et al. 2020)</td>
<td>PGD-7</td>
<td>6.9</td>
<td>20.2</td>
<td>78.6</td>
<td>13.8</td>
</tr>
<tr>
<td></td>
<td>RAXCL *(Dong et al. 2020)</td>
<td>PGD-7</td>
<td>3.6</td>
<td>20.2</td>
<td>80.7</td>
<td>25.3</td>
</tr>
<tr>
<td></td>
<td>AdvRush*(Mok et al. 2021)</td>
<td>PGD-7</td>
<td>4.2</td>
<td>20.2</td>
<td>82.0</td>
<td>21.7</td>
</tr>
<tr>
<td></td>
<td>RobNet-Sr(Guo et al. 2020)</td>
<td>TRADES-1</td>
<td>4.4</td>
<td>20.2</td>
<td>90.5</td>
<td>15.8</td>
</tr>
<tr>
<td></td>
<td>RAXCL *(Dong et al. 2020)</td>
<td>TRADES-1</td>
<td>3.6</td>
<td>20.2</td>
<td>91.0</td>
<td>15.4</td>
</tr>
<tr>
<td></td>
<td>AdvRush*(Mok et al. 2021)</td>
<td>TRADES-1</td>
<td>4.2</td>
<td>20.2</td>
<td>91.6</td>
<td>15.8</td>
</tr>
<tr>
<td></td>
<td>E2R-NAS *(Yue et al. 2020)</td>
<td>TRADES-1</td>
<td>4.0</td>
<td>20.2</td>
<td>90.1</td>
<td>18.0</td>
</tr>
<tr>
<td></td>
<td>NAS-OOOD*(Bai et al. 2021)</td>
<td>TRADES-1</td>
<td>4.4</td>
<td>20.2</td>
<td>91.2</td>
<td>21.3</td>
</tr>
<tr>
<td>Wsr-NAS (Naive)</td>
<td>WsrNet-Naive-1</td>
<td>TRADES-1</td>
<td>4.4</td>
<td>20.2</td>
<td>89.6</td>
<td>21.7</td>
</tr>
<tr>
<td></td>
<td>WsrNet-Naive-2</td>
<td>TRADES-1</td>
<td>4.5</td>
<td>20.2</td>
<td>89.6</td>
<td>22.1</td>
</tr>
<tr>
<td></td>
<td>WsrNet-Naive-6</td>
<td>TRADES-1</td>
<td>4.4</td>
<td>20.2</td>
<td>91.2</td>
<td>22.9</td>
</tr>
<tr>
<td>Wsr-NAS (Full)</td>
<td>WsrNet-Basic</td>
<td>PGD-7</td>
<td>4.5</td>
<td>20.2</td>
<td>80.8</td>
<td>24.8</td>
</tr>
<tr>
<td></td>
<td>WsrNet-Robust</td>
<td>PGD-7</td>
<td>4.5</td>
<td>20.2</td>
<td>78.3</td>
<td>27.9</td>
</tr>
<tr>
<td></td>
<td>WsrNet-Plus</td>
<td>PGD-7</td>
<td>4.7</td>
<td>20.2</td>
<td>80.7</td>
<td>25.7</td>
</tr>
<tr>
<td></td>
<td>WsrNet-Plus</td>
<td>TRADES-1</td>
<td>4.5</td>
<td>20.2</td>
<td>90.4</td>
<td>22.4</td>
</tr>
</tbody>
</table>

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 naïvely applying the One-Shot-NAS framework, a major source of time consumption is the process of calculating gradient to update architecture weights $H$ 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 $H$ 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 $H$ and appropriate configurations in the Robust Search Objective, gradient direction to update the architecture weight $H$ can be calculated, significantly reducing the cost of updating architecture weight during the search.

More formally, we describe the VLE in EWSS as:

$$
\tilde{L} = \{\tilde{L}_{\text{natural}}, \tilde{L}_{e_1}, \ldots, \tilde{L}_{e_{N_1+N_2}}\} = \psi(H).
$$

Within Eqn.8, $\tilde{L} \in \mathbb{R}^{N_1+N_2+1}$ 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 $H$; 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 $M_e = \{(H_i, L_i), \forall 1 \leq t \leq T\}$ 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

$$
L_e(\psi) = \frac{1}{T} \sum_{t=1}^{T} ||\psi(H_t) - L_t||_2^2
$$

over the memory $M_e$, where $t$ represents the index in the memory, $L_t$ is the validation loss values evaluated on $H_t$. Leveraging the VLE defined, we propose the Robust Search Objective in EWSS as:

$$
\min_{A} \alpha \tilde{L}_{\text{natural}} + \beta \tilde{L}_{e_i}
$$

s.t. $H = \text{GumbelSoftmax}(A, \xi, \tau)$,

$$
\sum_{i=1}^{n} \beta_i = 1, \alpha + \beta = 1, \alpha > 0, \beta > 0, \beta_i > 0 \forall i
$$

where $\xi$ is an i.i.d sample from Gumbel(0,1) and $\tau$ is temperature. With the value of $\alpha$, $\beta$ and different $\beta_i$ selected, the
architecture weight $\mathbf{A}$ in the super-network can be adjusted according to the gradient of Eqn. 10 by $\mathbf{A} \leftarrow \mathbf{A} + \epsilon \nabla \mathbf{A}$.

Within Eqn. 10, by having the $\alpha$ and $\beta$ terms, we allow customized priority over accuracy and overall robustness. With different $\beta_i$ being summed up to 1, by adjusting certain $\beta_i$ 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.

**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 $D_i$ and $D_o$ to be different halves of the CIFAR-10 training set, let $M_n$ 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. $\alpha$ and $\beta$ 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 $\epsilon_i$ as in (Madry et al. 2019). Also, we used $L_{\infty}$ 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-

<table>
<thead>
<tr>
<th>Model</th>
<th>Train Type</th>
<th>Para. (M)</th>
<th>Acc. (%)</th>
<th>Acc. (%)</th>
<th>Acc. (%)</th>
<th>Acc. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRN-34</td>
<td>S</td>
<td>46.2</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>MA</td>
<td>23.5</td>
<td>37.0</td>
<td>12.5</td>
<td>6.3</td>
<td>3.5</td>
</tr>
<tr>
<td>ResNet-18</td>
<td>P7</td>
<td>11.2</td>
<td>45.6</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>P7</td>
<td>23.5</td>
<td>45.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>WRN-34</td>
<td>T1</td>
<td>46.2</td>
<td>46.1</td>
<td>18.1</td>
<td>11.2</td>
<td>7.9</td>
</tr>
<tr>
<td>AmoebaNet</td>
<td>P7</td>
<td>3.2</td>
<td>39.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PVVLL</td>
<td>P7</td>
<td>3.5</td>
<td>47.4</td>
<td>15.7</td>
<td>7.3</td>
<td>3.8</td>
</tr>
<tr>
<td>NAS-Net*</td>
<td>T1</td>
<td>3.3</td>
<td>49.1</td>
<td>15.5</td>
<td>4.4</td>
<td>0.5</td>
</tr>
<tr>
<td>DARTS*</td>
<td>T1</td>
<td>3.2</td>
<td>46.9</td>
<td>9.3</td>
<td>1.23</td>
<td>0.1</td>
</tr>
<tr>
<td>RobNet-S</td>
<td>P7</td>
<td>4.4</td>
<td>48.3</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RobNet-M</td>
<td>P7</td>
<td>5.7</td>
<td>49.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RobNet-L</td>
<td>P7</td>
<td>6.9</td>
<td>49.4</td>
<td>3.1</td>
<td>1.8</td>
<td>0.7</td>
</tr>
<tr>
<td>RACL</td>
<td>P7</td>
<td>3.6</td>
<td>49.7</td>
<td>22.4</td>
<td>14.0</td>
<td>9.4</td>
</tr>
<tr>
<td>AdvRush*</td>
<td>T1</td>
<td>4.2</td>
<td>48.7</td>
<td>22.3</td>
<td>10.8</td>
<td>4.9</td>
</tr>
<tr>
<td>RobNet-S</td>
<td>T1</td>
<td>4.4</td>
<td>46.9</td>
<td>11.6</td>
<td>4.11</td>
<td>1.27</td>
</tr>
<tr>
<td>RACL*</td>
<td>T1</td>
<td>3.6</td>
<td>46.9</td>
<td>11.6</td>
<td>2.8</td>
<td>0.3</td>
</tr>
<tr>
<td>AdvRush*</td>
<td>T1</td>
<td>4.2</td>
<td>48.2</td>
<td>11.8</td>
<td>3.1</td>
<td>0.3</td>
</tr>
<tr>
<td>E2R</td>
<td>T1</td>
<td>4.0</td>
<td>40.5</td>
<td>14.5</td>
<td>10.2</td>
<td>6.9</td>
</tr>
<tr>
<td>NAS</td>
<td>T1</td>
<td>4.4</td>
<td>50.3</td>
<td>20.6</td>
<td>10.5</td>
<td>3.8</td>
</tr>
<tr>
<td>OOD*</td>
<td>T1</td>
<td>4.4</td>
<td>45.8</td>
<td>19.1</td>
<td>13.3</td>
<td>8.6</td>
</tr>
<tr>
<td>WsrNet-N1</td>
<td>T1</td>
<td>4.5</td>
<td>45.9</td>
<td>20.3</td>
<td>13.3</td>
<td>8.7</td>
</tr>
<tr>
<td>WsrNet-N3</td>
<td>T1</td>
<td>4.4</td>
<td>46.3</td>
<td>20.1</td>
<td>15.6</td>
<td>9.41</td>
</tr>
<tr>
<td>WsrNet-N6</td>
<td>T1</td>
<td>4.7</td>
<td>48.9</td>
<td>22.3</td>
<td>15.7</td>
<td>12.4</td>
</tr>
<tr>
<td>WsrNet-B</td>
<td>P7</td>
<td>4.5</td>
<td>48.4</td>
<td>27.2</td>
<td>20.3</td>
<td>15.5</td>
</tr>
<tr>
<td>WsrNet-R</td>
<td>P7</td>
<td>4.5</td>
<td>48.4</td>
<td>27.2</td>
<td>20.3</td>
<td>15.5</td>
</tr>
<tr>
<td>WsrNet-P</td>
<td>P7</td>
<td>4.7</td>
<td>48.9</td>
<td>23.1</td>
<td>17.0</td>
<td>13.7</td>
</tr>
<tr>
<td>WsrNet-B</td>
<td>T1</td>
<td>4.5</td>
<td>45.9</td>
<td>20.0</td>
<td>14.1</td>
<td>9.4</td>
</tr>
<tr>
<td>WsrNet-P</td>
<td>T1</td>
<td>4.7</td>
<td>47.5</td>
<td>22.1</td>
<td>16.0</td>
<td>11.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Search Config</th>
<th>WsrNet -Basic</th>
<th>WsrNet -Plus</th>
<th>WsrNet -Naive-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_1, N_2$</td>
<td>3,3</td>
<td>3,8</td>
<td>1,0</td>
</tr>
<tr>
<td>Total Time</td>
<td>3.6 days</td>
<td>4.0 days</td>
<td>2.6 days</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Search Config</th>
<th>WsrNet -Naive-3</th>
<th>WsrNet -Naive-6</th>
<th>WsrNet -Naive-11</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_1, N_2$</td>
<td>3,0</td>
<td>6,0</td>
<td>11,0</td>
</tr>
<tr>
<td>Total Time</td>
<td>3.4 days</td>
<td>5.2 days</td>
<td>7.4 days$^*$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Search Config</th>
<th>RobNet</th>
<th>MetaQNN</th>
<th>AmoNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_1, N_2$</td>
<td>1,1</td>
<td>1,1</td>
<td>1,1</td>
</tr>
<tr>
<td>Total Time</td>
<td>22.6 days$^*$</td>
<td>400 × 7 days</td>
<td>10 × 8 days$^*$</td>
</tr>
</tbody>
</table>

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

Table 3: total GPU time for different search algorithms or search configurations. * indicates estimated time.
To compare the performance of WsrNet-Basic with other models, we train WsrNet-Basic, WsrNet-Plus, RobNet-NAS, 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% respectively, 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 FGS. As in Tab. 1, by being attacked by the FGS, 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**


