DSRNA: Differentiable Search of Robust Neural Architectures

Ramtin Hosseini, Xingyi Yang, Pengtao Xie
UC San Diego
{rhossein, x3yang, p1xie}@eng.ucsd.edu

Abstract

In deep learning applications, the architectures of deep neural networks are crucial in achieving high accuracy. Many methods have been proposed to search for high-performance neural architectures automatically. However, these searched architectures are prone to adversarial attacks. A small perturbation of the input data can render the architecture to change prediction outcomes significantly. To address this problem, we propose methods to perform differentiable search of robust neural architectures. In our methods, two differentiable metrics are defined to measure architectures' robustness, based on certified lower bound and Jacobian norm bound. Then we search for robust architectures by maximizing the robustness metrics. Different from previous approaches which aim to improve architectures' robustness in an implicit way: performing adversarial training and injecting random noise, our methods explicitly and directly maximize robustness metrics to harvest robust architectures. On CIFAR-10, ImageNet, and MNIST, we perform game-based evaluation and verification-based evaluation on the robustness of our methods. The experimental results show that our methods 1) are more robust to various norm-bound attacks than several robust NAS baselines; 2) are more accurate than baselines when there are no attacks; 3) have significantly higher certified lower bounds than baselines.

1. Introduction

In deep learning applications, the architectures of neural models play a crucial role in improving performance. For example, on the ImageNet [15] benchmark, the image classification error is reduced from 16.4% to 3.57%, when the architecture is evolved from AlexNet [26] to ResNet [20]. Previously, neural architectures are mostly designed by humans, which is time-consuming to obtain a highly-performant architecture. Recently, automated neural architecture search [49, 50, 36, 37, 41, 42] which develops algorithms to find out the optimal architecture that yields the best performance on the validation datasets, has raised much attention and achieved promising results. For example, on the CIFAR-10 dataset, an automatically searched architecture [32] achieves an image classification error rate of 2.76% while the error achieved by state-of-the-art human-designed architecture is 3.46%.

As we will show in the experiments, the architectures searched by existing methods are prone to adversarial attacks. A small perturbation (which is not perceivable by humans) of the input data can render the architecture to change prediction outcomes significantly. Many approaches [18, 4, 33, 12, 28] have been proposed to improve the robustness of DNNs. In these approaches, the architecture of a DNN is provided by humans, and the defense method focuses on training the weights in this architecture in a robust way. However, the robustness of a DNN is not only relevant to its weight parameters, but also determined by the architecture. It is important to search for architectures that are robust to adversarial attacks as well.

In this paper, we develop a novel approach for robust NAS. We define two differentiable metrics to measure the robustness of architectures and formulate robust NAS as an optimization problem that aims to find out an optimal architecture by maximizing the robustness metrics. The first metric is defined based on certified lower bound [2]. Linear bounding methods are applied to individual building blocks in the differentiable architecture search space and these individual bounds are composed to obtain global bounds for the entire neural architecture. The second metric is based on the Jacobian norm bound [21], where the robustness is measured by how much the output shifts as the input is perturbed. The shift is upper bounded by the norms of row vectors in the Jacobian matrix of the neural architecture. Our approach is applicable to various forms of differentiable architecture search methods (e.g., DARTS [32], PC-DARTS [46], P-DARTS [9], etc. and is robust against adversarial attacks in various norm choices. Previously, robust NAS has been investigated in [19, 8], based on adversarial training of randomly sampled sub-architectures [19] and differentiable architecture variables [8]. Unlike these methods that achieve robustness implicitly via adversarial training, our method explicitly defines robustness metrics and directly

optimizes these metrics to obtain robust architectures.

On CIFAR-10, ImageNet, and MNIST, we perform game-based evaluation and verification-based evaluation on the robustness of our methods. The experimental results show that our methods 1) are more robust to various norm-bound attacks than several robust NAS baselines; 2) are more accurate than baselines when there are no attacks; 3) have significantly higher certified lower bounds than baselines.

The major contributions of this paper include:

- We propose a novel robust NAS method, which searches robust architectures by maximizing differentiable robustness metrics, defined based on certified lower bound and Jacobian norm bound. Our methods have strong guarantees in obtaining robust architectures by explicitly and directly maximizing robustness measures. In contrast, previous approaches perform implicit robustification of architectures via adversarial training, which is not guaranteed to yield robust architectures. Besides, our methods can be applied to robustify any differentiable NAS methods, in a principled and unified way.

- Experiments on ImageNet, CIFAR-10, and MNIST show that the architectures searched by our methods are robust to various forms of adversarial attacks and are as accurate as state-of-the-art NAS methods when there are no attacks. Our methods are consistently more robust than previous approaches against various attacks. In contrast, previous approaches are effective for certain types of attacks, but ineffective for other types.

The rest of the paper is organized as follows. Section 2 reviews related works. Section 3 and 4 present the method and experiments. Section 5 concludes the paper.

2. Related Works

2.1. Neural Architecture Search

In general, there are three paradigms of methods in NAS: reinforcement learning (RL) approaches [49, 35, 50], evolutionary learning approaches [31, 36], and differentiable approaches [3, 32, 45]. In RL-based approaches, a policy is learned to iteratively generate new architectures by maximizing a reward, which is the accuracy on the validation set. Evolutionary learning approaches represent the architectures as individuals in a population. Individuals with high fitness scores (validation accuracy) have the privilege to generate offspring, which replace individuals with low fitness scores. Differentiable NAS approaches adopt a network pruning strategy. On top of an over-parameterized network, the weights of connections between nodes are learned using gradient descent. Then weights close to zero are later pruned. There have been many efforts devoted to improving differentiable NAS methods. In P-DARTS [9], the depth of searched architectures is allowed to grow progressively during the training process. Search space approximation and regularization approaches are developed to reduce computational overheads and improve search stability. PC-DARTS [46] reduces the redundancy in exploring the search space by sampling a small portion of a super network. Operation search is performed in a subset of channels with the held out part bypassed in a shortcut. DARTS+ [29] leverages early stopping to avoid the collapse of DARTS' performance.

2.2. Adversarial Attacks and Defenses

Adversarial attacks aim to perturb input data examples by adding imperceptible noises so that the prediction results are altered significantly. In white-box attack [38, 6, 11, 51], the adversary has full access to the target model, while in the black-box attack [7, 22, 39, 10], the target model is unknown to the adversary. In targeted attacks, the adversary aims to change the prediction outcome in certain classes, while untargeted attacks are not class-specific. Arguably, the most popular and effective white-box untargeted attacks with various norm-bounds are: fast gradient sign method (FGSM) [18], projected gradient descent (PGD) [33], and Carlini & Wagner (C&W) [4]. FGSM is a single step attack algorithm that aims to increase the adversarial loss by updating its gradient sign. PGD is a more general version of FGSM that runs over several iterations to increase the adversarial loss. The attacks of FGSM and PGD are based on l_∞-norm bound, while those in C&W are based on l_0, l_2, and l_∞ norms. C&W is particularly effective for l_2-norm attacks. Additionally, a recent work AutoAttack [14] proposes a reliable and robust attack method using an ensemble of stepsize-free versions of PGD attacks, a white-box attack – Fast Adaptive Boundary (FAB) [13], and a black-box attack – Square Attack [1] to create parameter-free attacks. To improve the robustness of neural networks against adversarial attacks, many adversarial defense methods have been proposed, such as random smoothing [28, 12], adversarial training [18, 33, 4], and Jacobian regularization [23, 21, 5]. Jacobian regularization aims to minimize the change of network outputs when inputs are perturbed. Mathematically, this amounts to minimizing the Frobenius norm of a Jacobian matrix.

Most of these defense methods assume the neural architectures are manually designed by humans and focus on improving the robustness of network weights. Automatically searching for robust architectures is largely under-explored. In [16], experiments show that architectures searched by existing NAS methods such as DARTS, PC-DARTS, and P-DARTS are vulnerable to various forms of adversarial attacks. To address this issue, studies have been conducted to

robustify NAS methods. RobNet [19] used one-shot NAS to obtain a large number of networks and then studied the patterns of architectures that are robust against adversarial attacks. They discovered that using dense connectivity and adding convolution operations to direct connection edges help to improve robustness. Chen et al. [8] proposed performing adversarial training and random smoothing on architecture variables, which can improve the robustness of DARTS-based methods. Our work takes a different approach for robustifying architectures, where we explicitly define differentiable metrics to measure architectures' robustness and search for robust architectures by maximizing these metrics.

2.3. Robustness Verification of Neural Networks

Robustness verification aims to provide certified defense against any possible attacks under a threat model. A robustness certificate ϵ means the prediction outcome cannot be changed if the strength of the attack is smaller than ϵ. Many verification approaches [43, 40, 48, 17] have been proposed, which focus on achieving tighter lower bounds of the robustness certificate, computing bounds for various complex building blocks in neural networks, and improving the efficiency in computing the bounds. Dvijotham et al. [17] formulate verification as an optimization problem and seek bounds of the certificate by solving a Lagrangian relaxation of the optimization problem. Weng et al. [40] propose methods to verify the robustness of Rectified Linear Unit (ReLU) networks by bounding the ReLU units with linear functions or local Lipschitz constant. CNN-Cert [2] applies linear bounding techniques to provide certified lower bounds for various operations including convolution, pooling, batch normalization, residual blocks, activation functions, etc.

3. Methods

We begin with defining differentiable metrics to measure the robustness of neural architectures. Then we propose a robust NAS framework that performs optimization in the architecture search space to maximize the robustness metrics. The objective function explores a tradeoff between predictive accuracy and robustness and can be efficiently optimized using gradient-based methods.

3.1. Defining Differentiable Robustness Metrics

In this section, we define two differentiable metrics to measure the robustness of neural architectures. The first one is based on robustness certification methods [2]. Specifically, given an architecture, we seek to obtain a certified lower bound of this architecture and use the bound to measure robustness. The architecture with a larger lower bound is more robust against different attacks. The second metric is based on upper-bounding the shift of the model's prediction when the inputs are perturbed, and the bound is based on the norm of the Jacobian matrix [21] of the architecture. The smaller the upper bound is, the more robust the network is. Previous works [2, 21] have utilized certified bounds and Jacobian regularization to measure or improve the robustness of neural networks that have human-designed and fixed architectures. Different from these works, our work defines certified bounds and Jacobian regularizers on neural architecture variables and leverage them to search for robust architectures.

3.1.1 Measuring Robustness Based on Certified Bound

One way to measure the robustness of a neural network is to use the verified robustness certificate. A certificate with value $\epsilon(\mathbf{x})$ means that model prediction on the input data $\mathbf{x}$ cannot be changed if the attack strength is smaller than $\epsilon(\mathbf{x})$. A larger $\epsilon(\mathbf{x})$ indicates more robustness. In practice, it is infeasible to obtain the exact robustness certificate of a model. Instead, one can derive lower bounds of $\epsilon(\mathbf{x})$ and use these lower bounds as surrogates for measuring robustness. Given an architecture search space comprised of various building blocks such as ReLU-Conv-BN, (dilated) separable convolutions, pooling operations, etc., we perform linear bounding [2] on these building blocks and compose the individual bounds to obtain a certified lower bound for each architecture in the search space. These bounds are differentiable functions of architecture variables and are amenable for gradient-based optimization. In the sequel, we discuss how to derive the certified upper and lower bounds for each type of building blocks.

ReLU-Conv-BN Block The ReLU-Conv-BN building block consists of three consecutive operations including rectified linear unit (ReLU) as a nonlinear activation operation, convolution, and batch normalization (BN). Let Φ^r and Φ^{r-2} be the output and input of an ReLU-Conv-BN block r, then we have

$$\Phi^{r-1} = W^{r-1} * \sigma(\Phi^{r-2}) + b^{r-1} \tag{1}$$

$$\Phi^{r} = \gamma_{bn} \frac{\Phi^{r-1} - \mu_{bn}}{\sqrt{\sigma_{bn}^2 + \epsilon_{bn}}} + \beta_{bn} \tag{2}$$

where $\sigma(\cdot)$ is the ReLU function. W^{r-1} and b^{r-1} are the weight parameters and bias parameters in the convolution operation. μ_{bn} and σ_{bn}^2 are the mean and variance of a batch of Φ^{r-1} in batch normalization. γ_{bn}, ϵ_{bn}, and β_{bn} are hyperparameters in BN.

By applying linear bounds to these equations, we get these upper and lower bounds:

$$A_{L,bn}^r * \Phi^{r-1} + B_{L,bn}^r \leq \Phi^r \leq A_{U,bn}^r * \Phi^{r-1} + B_{U,bn}^r \tag{3}$$

$$A^r_{L,bn}\Phi^{r-1} + B^r_{L,bn} \geq A^r_{L,bn}(A^{r-1}_{L,conv}\Phi^{r-2} + B^{r-1}_{L,conv}) + B^r_{L,bn} \quad (4)$$

$$A^r_{U,bn}\Phi^{r-1} + B^r_{U,bn} \leq A^r_{U,bn}(A^{r-1}_{U,conv}\Phi^{r-2} + B^{r-1}_{U,conv}) + B^r_{U,bn} \quad (5)$$

where $A_{L,bn}$, $A_{U,bn}$, $B_{L,bn}$, and $B_{U,bn}$ are constants that can be computed as in [2]:

$$A^r_{L,bn} = A^r_{U,bn} = \frac{\gamma_{bn}}{\sqrt{\sigma^2_{bn} + \epsilon_{bn}}} \quad (6)$$

$$B^r_{L,bn} = B^r_{U,bn} = \frac{-\gamma_{bn}\mu_{bn}}{\sqrt{\sigma^2_{bn} + \epsilon_{bn}}} + \beta_{bn} \quad (7)$$

and $A_{L,conv}$, $A_{U,conv}$, $B_{L,conv}$, $B_{U,conv}$ are constant tensors.

(Dilated) Separable Convolutions Another two types of building blocks in our search space are separable convolutions and dilated separable convolutions. Dilated separable convolutions consist of four consecutive operations: ReLU, convolution, convolution, and batch normalization (BN). Separable convolutions consist of two consecutive dilated separable convolutions. Let Φ^{r-3} and Φ^r denote the input and output of a dilated separable convolution, then:

$$\Phi^{r-1} = W^{r-1} * (W^{r-2} * \sigma(\Phi^{r-3}) + b^{r-2}) + b^{r-1} \quad (8)$$

where W^{r-1} and W^{r-2} are weights of convolutions; b^{r-2} and b^{r-1} are bias parameters in convolutions. The calculation of Φ^r is the same as that in Eq.(2). We can again use Eq.(3) to find the upper and lower bound of Φ^r, which are:

$$A^r_{L,bn} * \Phi^{r-1} + B^r_{L,bn} \geq A^r_{L,bn} * (A^{r-1}_{L,conv} * (W^{r-2} \Phi^{r-3} + b^{r-2}) + B^{r-1}_{L,conv}) + B^r_{L,bn} \quad (9)$$

$$A^r_{U,bn} * \Phi^{r-1} + B^r_{U,bn} \leq A^r_{U,bn} * (A^{r-1}_{U,conv} * (W^{r-2} \Phi^{r-3} + b^{r-2}) + B^{r-1}_{U,conv}) + B^r_{U,bn} \quad (10)$$

The upper and lower bound for separable convolution operations can be derived in a similar way.

Pooling Operations Let Φ^{r-1} and Φ^r denote the input and output of a pooling operation r. We have the following lower and upper bound of Φ^r:

$$A^r_{L,pool} * \Phi^{r-1} + B^r_{L,pool} \leq \Phi^r \leq A^r_{U,pool} * \Phi^{r-1} + B^r_{U,pool} \quad (11)$$

Robustness Metric Given the lower and upper bounds of individual building blocks, we are ready to derive a certified lower bound for the entire network as a measure of the robustness of its architecture. In differentiable architecture search [32], the neural network is overparameterized with many building blocks that are organized into a directed acyclic graph (DAG). The output of each block is multiplied with a positive scalar. The larger the scalar is, the more critical the block is. After learning, a subset of blocks with the largest scalars are selected to form the final architecture of this network. Therefore, these scalars (called architecture variables) represent the architecture. Given a block with lower bound L and upper bound U, after multiplying with an architecture variable α, this block has a lower bound of αL and αU. Following the topological order of blocks in the DAG, we recursively compose the lower and upper bounds (multiplied with architecture variables) of blocks and get a global lower and upper bound for the entire network. These two bounds are functions of architecture variables and the input data example. The lower bound is used as the robustness metric.

3.1.2 Measuring Robustness with Jacobian Regularization

When the architecture search space is large, computing gradients of the certified lower bound with respect to architecture variables is time-consuming. To address this problem, we investigate another measure of robustness, which is computationally efficient. Let $f(\mathbf{x})$ denote the neural network which takes a data example $\mathbf{x} \in \mathbb{R}^D$ as input and outputs a K-dimensional vector. Similar to the robustness metric defined in Section 3.1.1, the architecture search space is differentiable, where continuous architecture variables are multiplied to the outputs of building blocks. Therefore, $f(\mathbf{x})$ is a continuous function of the architecture variables. Let $\mathbf{x} + \boldsymbol{\epsilon}$ be an adversarial example where $\boldsymbol{\epsilon}$ is a small perturbation vector. We assume the p-norm of $\boldsymbol{\epsilon}$ is less equal to a small scalar δ: $\|\boldsymbol{\epsilon}\|_p \leq \delta$. The robustness of the network can be measured using the following quantity [21]:

$$S = -\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\boldsymbol{\epsilon}}\left[\frac{1}{K}\sum_{k=1}^{K}|f_k(\mathbf{x}+\boldsymbol{\epsilon}) - f_k(\mathbf{x})|\right] \quad (12)$$

where $a = 1/K\sum_{k=1}^{K}|f_k(\mathbf{x}+\boldsymbol{\epsilon}) - f_k(\mathbf{x})|$ is the average change of the output across all dimensions when $\mathbf{x}$ is perturbed with $\boldsymbol{\epsilon}$ and S is the expectation of a defined with respect to the distributions of $\mathbf{x}$ and $\boldsymbol{\epsilon}$. The smaller this quantity is, the more robust the network is: intuitively, a network is robust if for every input data example, no matter how it is perturbed, the change of network output is small.

According to Taylor expansion, we have:

$$f_k(\mathbf{x}+\boldsymbol{\epsilon}) - f_k(\mathbf{x}) \approx \left[\frac{\partial f_k(\mathbf{x})}{\partial \mathbf{x}}\right]^\top \boldsymbol{\epsilon} \tag{13}$$

Let $\mathbf{J}(\mathbf{x})$ denote the Jacobian matrix at $\mathbf{x}$ where $J_{kj} = \partial f_k(\mathbf{x})/\partial x_j$. Then $\partial f_k(\mathbf{x})/\partial \mathbf{x} = \mathbf{J}_k(\mathbf{x})$ where $\mathbf{J}_k(\mathbf{x})$ is the k-th row vector of $\mathbf{J}(\mathbf{x})$. According to Hölder's inequality, we have:

$$\left|\mathbf{J}_k(\mathbf{x})^\top \boldsymbol{\epsilon}\right| \leq \left\|\mathbf{J}_k(\mathbf{x})\right\|_q \|\epsilon\|_p \leq \left\|\mathbf{J}_k(\mathbf{x})\right\|_q \delta \tag{14}$$

where $\frac{1}{p} + \frac{1}{q} = 1$.

Putting these pieces together, we have:

$$\begin{aligned}
&-\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\boldsymbol{\epsilon}}\left[\tfrac{1}{K}\textstyle\sum_{k=1}^K |f_k(\mathbf{x}+\boldsymbol{\epsilon}) - f_k(\mathbf{x})|\right] \\
&\approx -\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\boldsymbol{\epsilon}}\left[\tfrac{1}{K}\textstyle\sum_{k=1}^K |\mathbf{J}_k(\mathbf{x})^\top \boldsymbol{\epsilon}|\right] \\
&\geq -\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\boldsymbol{\epsilon}}\left[\tfrac{1}{K}\textstyle\sum_{k=1}^K \left\|\mathbf{J}_k(\mathbf{x})\right\|_q \delta\right] \\
&= -\delta\mathbb{E}_{\mathbf{x}}\left[\tfrac{1}{K}\textstyle\sum_{k=1}^K \left\|\mathbf{J}_k(\mathbf{x})\right\|_q\right] \\
&\approx -\tfrac{\delta}{N}\textstyle\sum_{i=1}^N\left[\tfrac{1}{K}\sum_{k=1}^K \left\|\mathbf{J}_k(\mathbf{x}_i)\right\|_q\right]
\end{aligned} \tag{15}$$

where in the last step, the expectation is approximated by the mean on a set of data examples $\{\mathbf{x}_i\}_{i=1}^N$. To maximize S for achieving robustness, we can maximize its approximated lower bound $-\delta/N\sum_{i=1}^N\left[1/K\sum_{k=1}^K \left\|\mathbf{J}_k(\mathbf{x}_i)\right\|_q\right]$. This bound is referred to as the Jacobian norm bound. It is a function of the architecture variables. For l_2 and l_∞ norm bound attacks, $\sum_{k=1}^K \left\|\mathbf{J}_k(\mathbf{x})\right\|_q$ is the Frobenius norm and l_1 norm of the Jacobian matrix, respectively. We use the method in [21] to compute the Jacobian matrix efficiently based on random projection.

3.2. Differentiable Search of Robust Neural Architectures

Given the robustness metrics defined based on certified lower bound and Jacobian norm bound, which are increasing functions of the architecture variables (i.e., larger values of the metrics indicate that the architecture is more robust), we search for robust architectures by maximizing these robustness metrics. The formulation is as follows:

$$\begin{aligned}
&\min_{\alpha} \quad \sum_{i=1}^M L(w^*(\alpha), \alpha, x_i^{(\text{val})}) - \gamma R(w^*(\alpha), \alpha, x_i^{(\text{val})}) \\
&s.t. \quad w^*(\alpha) = \operatorname*{argmin}_{w} \quad \sum_{i=1}^N L(w, \alpha, x_i^{(\text{tr})})
\end{aligned} \tag{16}$$

where α denotes the set of architecture variables, and w denotes the weight parameters of blocks. R denotes the robustness metric (either based on certified lower bound or Jacobian norm bound). M is the number of validation examples, and N is the number of training examples. On each validation example $x_i^{(\text{val})}$, we measure the robustness R and predictive loss L of the architecture α and aim to search for an optimal architecture that yields the largest robustness and smallest predictive loss on the validation set. γ is a tradeoff parameter balancing these two objectives. Similar to [32], this is a bi-level optimization problem. In the inner optimization problem, given an architecture configuration α, an optimal set of weights $w^*(\alpha)$ is learned by minimizing the training loss $\sum_{i=1}^N L(w, \alpha, x_i^{(\text{tr})})$. Note that $w^*(\alpha)$ is a function of α: each architecture configuration α corresponds to a set of optimal weights $w^*(\alpha)$. $w^*(\alpha)$ and α are both used to measure the robustness and predictive loss on the validation set. In the outer optimization problem, we learn the architecture variables by minimizing the validation loss and maximizing the robustness metric, i.e., searching for an architecture that is accurate and robust. When R is the metric based on certified bound (CB), our method is denoted as DSRNA-CB; when R is the metric based on Jacobian norm bound, our method is denoted as DSRNA-Jacobian. The two metrics can be summed together as a single metric, leading to a DSRNA-Combined method. The algorithm for solving the optimization problem in Eq.(16) can be derived in a similar way to that in DARTS [32]. We approximate $w^*(\alpha)$ using one step gradient descent update of w with respect to the training loss. Then we plug in this approximation into the validation loss and robustness metric, and perform gradient descent update of α with respect to the approximated objective in the first line in Eq.(16). The detailed algorithm is deferred to the supplements.

4. Experiments

4.1. Dataset

We used three datasets in the experiments: CIFAR-10 [25], ImageNet [15], and MNIST [27]. CIFAR-10 contains 60K images with a size of 32×32. The train, validation, and test sets in CIFAR-10 contain 25K, 25K, 10K images, respectively. ImageNet has 1.3M training images and 50K validation images. MNIST has a training set of 60,000 examples and a test set of 10,000 examples, which are 28×28 gray-scale images of handwritten single digits between 0 and 9.

4.2. Experimental Settings

4.2.1 Baselines

We compare our proposed methods with the following baselines: 1) RobNet [19] which searches robust architectures based on adversarial training in one-shot NAS; 2) SDARTS-ADV and PC-DARTS-ADV [8], which performs adversarial training on architecture variables in DARTS-based NAS. During architecture evaluation, DSRNA-CB, DSRNA-Jacobian, DSRNA-Combined, SDARTS-ADV, and PC-DARTS-ADV are trained with Jacobian regularization,

Method	PGD (10)	PGD (20)	PGD (100)	FGSM	C&W	AutoAttack (l_∞)	AutoAttack (l_2)
RobNet-large [19]	49.49	49.44	49.24	54.98	47.19	48.93	46.38
RobNet-free [19]	52.80	52.74	52.57	58.38	46.95	50.13	46.33
SDARTS-ADV [8]*	56.94 ± 0.02	56.90 ± 0.04	56.77 ± 0.17	63.84 ± 0.02	42.67 ± 0.09	55.04 ± 0.07	40.98± 0.19
PC-DARTS-ADV [8]*	57.15 ± 0.02	57.11 ± 0.05	56.83 ± 0.21	65.29 ± 0.03	42.58 ± 0.04	55.29± 0.05	40.57± 0.21
DSRNA-CB (ours)*	60.31 ± 0.07	60.22 ± 0.11	59.93 ± 0.24	69.88 ± 0.09	63.01 ± 0.07	59.24± 0.04	61.87 ± 0.15
DSRNA-Jacobian (Ours)*	59.81 ± 0.02	59.77 ± 0.04	59.47 ± 0.14	68.92 ± 0.02	62.87 ± 0.04	59.11 ± 0.04	62.09 ± 0.10
DSRNA-Combined (Ours)* †	**61.12 ± 0.03**	**61.06 ± 0.04**	**60.71 ± 0.15**	**70.32 ± 0.04**	**64.76± 0.06**	**59.83 ± 0.05**	**64.51 ± 0.12**

Table 1. Accuracy (%) (mean and standard deviation) of different methods under various norm-bound attacks on CIFAR-10. *Average of five runs. † Using early stopping. The best method is boldfaced and the second best is underlined.

Method	Test Acc. (%)	Params (M)	Search Cost (GPU days)	Search Method
NASNet-A [49]	97.35	3.3	1800	RL
AmoebaNet-B [36]	97.45	2.8	3150	evolution
PNAS [30]†	96.59	3.2	255	SMBO
ENAS [35]	97.11	4.6	0.5	RL
DARTS (1st) [32]	97.00 ± 0.14	3.3	1.5	gradient
DARTS (2nd) [32]	97.26 ± 0.09	3.3	4.0	gradient
SNAS (moderate) [45]	97.15	2.8	1.5	gradient
ProxylessNAS [3]*	97.92	–	4.0	gradient
ASAP [34]	**98.01**	2.5	0.2	gradient
R-DARTS (L2) [47]	97.05 ± 0.21	–	1.6	gradient
DARTS+ [29]	97.68	3.7	0.4	gradient
P-DARTS [9]	97.50	3.4	0.3	gradient
PC-DARTS [46]	97.43 ± 0.07	3.6	0.1	gradient
RobNet-large [19]	78.57	6.9	–	one shot
RobNet-free [19]	82.79	5.5	–	one shot
SDARTS-RS [8]	97.33 ± 0.03	3.4	0.4	gradient
SDARTS-ADV [8]	97.39 ± 0.02	3.3	1.3	gradient
PC-DARTS-ADV [8]	97.51 ± 0.04	3.5	0.4	gradient
DSRNA-CB (ours)‡	97.42 ± 0.07	3.5	4.0	gradient
DSRNA-Jacobian (ours)‡	97.50 ± 0.03	3.5	0.4	gradient
DSRNA-Combined (Ours)‡ *	97.51 ± 0.04	3.5	0.6	gradient

Table 2. Accuracy (%) (mean and standard deviation) of different NAS methods when there are no attacks. ‡Average of five runs. †Training without cutout augmentation. *Using a different search space. ⋆Using early stopping.

while RobNet-Free and RobNet-large are trained with adversarial training. We select four popular adversarial attack methods to evaluate the robustness of our methods: fast gradient sign method (FGSM) [18] , projected gradient descent (PGD) [33], Carlini & Wagner (C&W) [4], and AutoAttack [14].

4.2.2 Hyperparameter Settings

The search space of our methods is the same as that of PC-DARTS, which is composed of 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated separable convolutions, 3×3 max pooling, 3×3 average pooling, identity, and zero. The convolutional cell consists of 6 nodes, which has 2 input nodes, 3 intermediate nodes, and 1 output node. For CIFAR-10 and MNIST, our methods search the architectures from scratch. In the searching phase, a small network of 8 cells was trained for 50 epochs with an initial number of channels of 16.

In DSRNA-CB, we used SGD for optimizing the network weights w with a learning rate of 0.1, a batch size of 256, a momentum of 0.9, and a weight decay of $3e-4$. We used the Adam optimizer [24] for optimizing architecture variables α, with a fixed learning rate of $6e-4$, $\beta_1 = 0.5$, $\beta_2 = 0.999$, and a weight decay of $3e-4$. In DSRNA-Jacobian, the network weights w were optimized via SGD with a learning rate of 0.025, a batch size of 128, a momentum of 0.9, and a weight decay of $3e-4$. The architecture variables α were optimized using Adam [24] with a learning rate of $3e-4$, $\beta_1 = 0.5$, $\beta_2 = 0.999$, and a weight decay of $1e-3$.

Given the searched cell, we stack 20 copies of them into a larger network and train this network from scratch on CIFAR-10 or MNIST. The network was trained for 600 epochs from scratch with a batch size of 128, an initial learning rate of 0.025, norm gradient clipping of 5, drop-path with a rate of 0.3, and an initial number of channels of 36. For ImageNet, the architecture is transferred from CIFAR-10: given the optimal cell searched on CIFAR-10, we stack 14 copies of them into a larger network with 48 initial channels and train this network on ImageNet. The training was performed for 250 epochs using an SGD optimizer with an annealing learning rate of 0.5, a momentum of 0.9, and a weight decay of $3e-5$. The tradeoff parameter γ in both DSRNA-CB and DSRNA-Jacobian was set to 0.01. In DSRNA-CB, we initialized ϵ as 0.03, and then linearly increased or decreased it based on the global difference between the certified upper bound and lower bound. The hyperparameters of baseline methods are deferred to the supplements. A single NVIDIA GTX 1080Ti GPU was used to perform the search.

4.3. Results

In this section, we perform game-based and verification-based evaluations of the adversarial robustness of our proposed methods and compare with state-of-the-art baselines.

4.3.1 Game-based Evaluation

Game-based evaluation estimates the success rate of defending against adversarial attacks with various forms of norm-bounds, such as l_2, l_∞, etc. FGSM [18, 44] and PGD [33] are two effective l_∞ attack methods. C&W [4] is an effective l_2 attack method. On CIFAR-10, ImageNet, and MNIST, we evaluate our proposed methods against 1) PGD

Method	Without attack	PGD (100)	FGSM	C&W	AutoAttack (l_∞)	AutoAttack (l_2)	Params (M)
RobNet-large [19]	61.26	37.14	39.74	25.73	32.96	23.90	11.6
SDARTS-ADV [8] *	74.85 ± 0.06	46.54 ± 0.13	48.09 ± 0.07	36.86 ± 0.10	41.58± 0.07	35.71± 0.15	4.7
PC-DARTS-ADV [8] *	75.73 ± 0.07	**46.59 ± 0.15**	48.25 ± 0.08	36.69 ± 0.09	41.79±0.06	35.86±0.11	5.3
DSRNA-CB (ours)*	75.84 ± 0.11	45.39 ± 0.18	**50.89 ± 0.07**	**43.64 ± 0.19**	**44.05±0.09**	**42.98±0.16**	5.4
DSRNA-Jacobian (ours)*	**75.88 ± 0.07**	43.79 ± 0.11	48.69 ± 0.04	43.17 ± 0.08	43.81±0.03	42.56± 0.11	5.3

Table 3. Accuracy (%) (mean and standard deviation) of different methods on ImageNet under various attacks and without attack. *Average of five runs. These architectures were searched on CIFAR-10. The best method is boldfaced.

Method	Without attack	PGD (100)	FGSM	C&W	AutoAttack (l_∞)	AutoAttack (l_2)
RobNet-large [19]	90.73	87.28	89.43	69.38	86.85	65.07
SDARTS-ADV [8] *	99.19 ± 0.01	97.31 ± 0.02	98.67 ± 0.02	78.94 ± 0.05	95.29±0.02	77.73±0.06
PC-DARTS-ADV [8] *	99.21 ± 0.01	97.33 ± 0.04	98.75 ± 0.01	78.93 ± 0.03	95.86±0.03	77.83±0.07
DSRNA-CB (ours)*	99.21 ± 0.03	97.34 ± 0.06	**98.85 ± 0.03**	94.02 ± 0.08	97.01±0.06	94.31±0.14
DSRNA-Jacobian (ours)*	99.36 ± 0.01	96.82 ± 0.02	98.79 ± 0.01	95.37 ± 0.02	96.28±0.04	94.91±0.08
DSRNA-Combined (ours)*	**99.40 ± 0.02**	**97.36 ± 0.04**	98.83 ± 0.04	**96.72 ± 0.02**	**96.31 ± 0.03**	**95.47± 0.09**

Table 4. Accuracy (%) (mean and standard deviation) of different methods on MNIST under various attacks and without attack. *Average of five runs. The best method is boldfaced and the second best is underlined.

attack with $\epsilon = 8/255$ on CIFAR-10, $\epsilon = 2/255$ on ImageNet, and $\epsilon = 0.3$ on MNIST, attack iterations of 10, 20, and 100, and a step size of $2/255$, 2) FGSM attack with $\epsilon = 2/255$, 3) C&W with 60 attack iterations, 4) AutoAttack (l_∞) with $\epsilon = 8/255$ on CIFAR-10, $\epsilon = 2/255$ on ImageNet, and $\epsilon = 0.3$ on MNIST, and 5) AutoAttack (l_2) with $\epsilon = 1$.

Table 1 shows the accuracy of different methods under various norm-bound attacks on CIFAR-10. PGD (n) denotes the PGD attack with n iterations. From this table, we make the following observations. **First**, the accuracy of our proposed methods, including DSRNA-CB and DSRNA-Jacobian is much higher than that of other robust NAS baselines including RobNet-large, RobNet-free, SDARTS-ADV, and PC-DARTS-ADV, under PGD, FGSM, C&W attacks, AutoAttack (l_∞), and AutoAttack (l_2). This demonstrates that our methods are more robust against various attacks than these baselines. One major reason is that our methods search for robust architectures by explicitly and directly maximizing differentiable robustness metrics and therefore are guaranteed to obtain robust architectures. In contrast, the baseline methods try to improve the robustness of searched architectures implicitly and indirectly: performing adversarial training and injecting random noise. The implicitness and indirectness of these methods do not guarantee robustness. **Second**, among the baselines, there is no consistent winner: SDARTS-ADV and PC-DARTS-ADV perform better than the other baselines under PGD attack, FGSM attack, and AutoAttack (l_∞); RobNet-large and RobNet-free perform better than the other baselines on C&W attack and AutoAttack (l_2). None of these baselines consistently outperforms others across all these types of attacks. In contrast, our proposed methods are consistently more robust than these baselines under all types of attacks. **Third**, between our two proposed methods DSRNA-CB and DSRNA-Jacobian, DSRNA-CB is slightly more robust than DSRNA-Jacobian. This is probably because the first-order Taylor approximation in DSRNA-Jacobian incurs larger inexactness. However, DSRNA-Jacobian is much faster to train and more memory efficient than DSRNA-CB, as we will show later. **Fourth**, DSRNA-Combined, which utilizes CB and Jacobian norm bound simultaneously for regularization, performs better than DSRNA-CB and DSRNA-Jacobian. This shows that when used together, these two regularizers bring in a synergistic effect.

While our methods are robust against different attacks, we also would like them to be accurate when there are no attacks. To verify this, we compare the accuracy of our methods with state-of-the-art baselines under the attack-free setting. Table 2 shows the accuracy achieved by different methods on CIFAR-10 when there are no attacks. From this table, we make the following observations. **First**, the accuracy achieved by our methods is very close to the best accuracy achieved by ASAP. This demonstrates that not only being robust, our methods are also highly accurate when there are no attacks. **Second**, the accuracy of RobNet is much lower than that of ours. This shows that while our methods are not only more robust than RobNet when there are attacks, but also are much more accurate than RobNet when there are no attacks. **Third**, in general, the search cost of our methods is similar to that of other gradient-based baselines. This demonstrates that our methods gain robustness without significantly increasing search cost. Note that the search cost of DSRNA-CB is higher than SDARTS-RS, SDARTS-ADV, and PC-DARTS-ADV. One may wonder whether DSRNA-CB achieves higher robustness than

Dataset	RobNet-large [19]	SDARTS-ADV [8]	PC-DARTS-ADV [8]	DSRNA-CB (ours)	DSRNA-Jacobian (ours)
MNIST	0.0325	0.0471	0.0474	**0.0526**	0.0514
CIFAR-10	0.0024	0.0039	0.0040	**0.0049**	0.0048

Table 5. Comparison of averaged l_∞-norm certified lower bounds of architectures searched by various methods. Larger is better.

Dataset	RobNet-large [19]	SDARTS-ADV [8]	PC-DARTS-ADV [8]	DSRNA-CB (ours)	DSRNA-Jacobian (ours)
MNIST	0.1340	0.1767	0.1765	**0.4288**	0.4285
CIFAR-10	0.0167	0.0337	0.0336	**0.0412**	0.0409

Table 6. Comparison of averaged l_2-norm certified lower bounds of architectures searched by various methods. Larger is better.

the three baselines because it performs search for a longer time. To check this, in DSRNA-CB, we decrease the batch-size to 64 and use early stopping to reduce the search cost to 0.5 GPU days. The corresponding accuracy on CIFAR-10 is: 57.82% under PGD(100), 65.94% under FGSM, 62.35% under C&W, and 97.37% under no attack. Comparing these results with those in Table 1, we can see that our DSRNA-CB method is still more robust than SDARTS-RS, SDARTS-ADV, and PC-DARTS-ADV when their search costs are about the same. **Fourth**, while SDARTS-ADV and PC-DARTS-ADV can achieve high performance when there are no attacks, they are not as robust as our methods in the presence of attacks, as shown in Table 1.

To investigate our methods' transferability, we use the best cell structure searched on CIFAR-10 to compose a larger network and train it on ImageNet. Table 3 shows the accuracy of different methods achieved on ImageNet under various norm-bound attacks and without attack. From this table, we make the following observations. **First**, under all the attacks, our methods achieve much higher accuracy than RobNet. Under C&W attack and AutoAttack (l_2), our methods achieve substantially higher accuracy than SDARTS-ADV and PC-DARTS-ADV. Under PGD attack, FGSM attack, and AutoAttack (l_∞), our methods are on par with SDARTS-ADV and PC-DARTS-ADV: our methods are slightly better than SDARTS-ADV and PC-DARTS-ADV under FGSM attacks and AutoAttack (l_∞); SDARTS-ADV and PC-DARTS-ADV are slightly better than our methods under PGD attacks. These results further demonstrate that our methods are more robust against various types of attacks than the baselines. **Second**, when there are no attacks, the accuracy of our methods is much higher than that of RobNet. In addition to being more robust, our methods are also more accurate than RobNet under the attack-free setting. **Third**, DSRNA-CB is slightly more robust than DSRNA-Jacobian. Note that the search costs of methods in Table 3 are the same as those in Table 2 since the architectures were searched on CIFAR-10 and evaluated on ImageNet.

Table 4 shows the results on MNIST. Similarly, our methods are substantially more robust than RobNet-large under all types of attacks, and are substantially more robust than SDARTS-ADV and PC-DARTS-ADV under C&W attacks, AutoAttack (l_2), and AutoAttack (l_∞). Our methods are on par with SDARTS-ADV and PC-DARTS-ADV under PGD and FGSM attacks. When there is no attack, our methods achieve much higher accuracy than RobNet-large and are on par with SDARTS-ADV and PC-DARTS-ADV.

Runtime With a single GTX 1080Ti GPU, the runtime on CIFAR-10 for the search phase of DSRNA-CB is 4 GPU days, while that of DSRNA-Jacobian is 0.4 GPU days. On MNIST, DSRNA-CB takes 1 GPU day for architecture search while DSRNA-Jacobian takes 0.2 GPU days. DSRNA-Jacobian is more efficient than DSRNA-CB, but is less robust than DSRNA-CB as shown previously.

4.3.2 Verification-based Evaluation

In this section, we use the certification method developed in Section 3.1.1 to find the certified lower bounds of the architectures searched by different methods. Larger lower bound indicates more robustness. Table 5 and Table 6 compare the averaged certified lower bounds of architectures searched by different methods on MNIST and CIFAR-10 under l_2 and l_∞ norms. As can be seen, the lower bounds achieved by our methods under various norms are larger than those achieved by baselines. This further demonstrates that our methods are more robust than these baseline methods.

5. Conclusion

To address the problem that existing neural architecture search (NAS) methods are vulnerable to adversarial attacks, we propose methods for differentiable search of robust architectures. We define two differentiable measures of architectures' robustness, based on certified robustness lower bound and Jacobian norm bound. Then we search for robust architectures by performing optimization in the architecture space with an objective of maximizing the robustness metrics. On various datasets, we demonstrate that our methods 1) are more robust to various norm-bound attacks than several robust NAS baselines; 2) are more accurate than baselines when there are no attacks; 3) have significantly higher certified lower bounds than baselines.

References

[1] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, pages 484–501. Springer, 2020.

[2] Akhilan Boopathy, Tsui-Wei Weng, Pin-Yu Chen, Sijia Liu, and Luca Daniel. Cnn-cert: An efficient framework for certifying robustness of convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3240–3247, 2019.

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

[4] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[5] Alvin Chan, Yi Tay, Yew Soon Ong, and Jie Fu. Jacobian adversarially regularized networks for robustness. *arXiv preprint arXiv:1912.10185*, 2019.

[6] Hongge Chen, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, and Cho-Jui Hsieh. Show-and-fool: Crafting adversarial examples for neural image captioning. *arXiv preprint arXiv:1712.02051*, 2017.

[7] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 15–26, 2017.

[8] Xiangning Chen and Cho-Jui Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization. *arXiv preprint arXiv:2002.05283*, 2020.

[9] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive darts: Bridging the optimization gap for nas in the wild. *arXiv preprint arXiv:1912.10952*, 2019.

[10] Minhao Cheng, Thong Le, Pin-Yu Chen, Jinfeng Yi, Huan Zhang, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. *arXiv preprint arXiv:1807.04457*, 2018.

[11] Minhao Cheng, Jinfeng Yi, Pin-Yu Chen, Huan Zhang, and Cho-Jui Hsieh. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. In *AAAI*, pages 3601–3608, 2020.

[12] Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. *arXiv preprint arXiv:1902.02918*, 2019.

[13] Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, pages 2196–2205. PMLR, 2020.

[14] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020.

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

[16] Chaitanya Devaguptapu, Devansh Agarwal, Gaurav Mittal, and Vineeth N Balasubramanian. An empirical study on the robustness of nas based architectures. *arXiv preprint arXiv:2007.08428*, 2020.

[17] Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy A Mann, and Pushmeet Kohli. A dual approach to scalable verification of deep networks. In *UAI*, volume 1, page 2, 2018.

[18] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[19] Minghao Guo, Yuzhe Yang, Rui Xu, Ziwei Liu, and Dahua Lin. When nas meets robustness: In search of robust architectures against adversarial attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 631–640, 2020.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[21] Judy Hoffman, Daniel A Roberts, and Sho Yaida. Robust learning with jacobian regularization. *arXiv preprint arXiv:1908.02729*, 2019.

[22] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. *arXiv preprint arXiv:1804.08598*, 2018.

[23] Daniel Jakubovitz and Raja Giryes. Improving dnn robustness to adversarial attacks using jacobian regularization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 514–529, 2018.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

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

[27] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[28] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 656–672. IEEE, 2019.

[29] Hanwen Liang, Shifeng Zhang, Jiacheng Sun, Xingqiu He, Weiran Huang, Kechen Zhuang, and Zhenguo Li. Darts+: Improved differentiable architecture search with early stopping. *arXiv preprint arXiv:1909.06035*, 2019.

[30] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018.

[31] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search, 2017. cite

arxiv:1711.00436Comment: Accepted as a conference paper at ICLR 2018.

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

[33] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[34] Asaf Noy, Niv Nayman, Tal Ridnik, Nadav Zamir, Sivan Doveh, Itamar Friedman, Raja Giryes, and Lihi Zelnik. Asap: Architecture search, anneal and prune. In *International Conference on Artificial Intelligence and Statistics*, pages 493–503. PMLR, 2020.

[35] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.

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

[37] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc Le, and Alex Kurakin. Large-scale evolution of image classifiers. *arXiv preprint arXiv:1703.01041*, 2017.

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

[39] Chun-Chen Tu, Paishun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 742–749, 2019.

[40] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Duane Boning, Inderjit S Dhillon, and Luca Daniel. Towards fast computation of certified robustness for relu networks. *arXiv preprint arXiv:1804.09699*, 2018.

[41] Yu Weng, Tianbao Zhou, Yujie Li, and Xiaoyu Qiu. Nas-unet: Neural architecture search for medical image segmentation. *IEEE Access*, 7:44247–44257, 2019.

[42] Yu Weng, Tianbao Zhou, Lei Liu, and Chunlei Xia. Automatic convolutional neural architecture search for image classification under different scenes. *IEEE Access*, 7:38495–38506, 2019.

[43] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5286–5295. PMLR, 2018.

[44] Eric Wong, Leslie Rice, and Zico J. Kolter. Fast is better than free: Revisiting adversarial training. *ICLR*, 2020.

[45] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018.

[46] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient differentiable architecture search. *arXiv preprint arXiv:1907.05737*, 2019.

[47] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. *arXiv preprint arXiv:1909.09656*, 2019.

[48] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in neural information processing systems*, pages 4939–4948, 2018.

[49] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[50] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.

[51] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2847–2856, 2018.