RNAS: ROBUST NETWORK ARCHITECTURE SEARCH BEYOND DARTS

Anonymous authors
Paper under double-blind review

ABSTRACT

The vulnerability of Deep Neural Networks (DNNs) (i.e., susceptibility to adversarial attacks) severely limits the application of DNNs. Most of the existing methods improve the robustness of the model from weights optimization, such as adversarial training and regularization. However, the architecture of DNNs is also a key factor to robustness, which is often neglected or underestimated. We propose a Robust Network Architecture Search (RNAS) to address this problem. In our method, we define a network vulnerability metric based on the features' deviation between clean examples and adversarial examples. Through constraining this vulnerability, we search the robust architecture and solve it by iterative optimization. The extensive experiments conducted on CIFAR-10/100 and SVHN show that our model achieves the best performance under various adversarial attacks compared with extensive baselines and state-of-the-art methods.

1 INTRODUCTION

In recent years, Deep Neural Networks (DNNs) have shown excellent performance in various applications, such as image classification (Krizhevsky et al., 2012; Huang et al., 2017a), target detection (Girshick et al., 2014), and semantic segmentation (Chen et al., 2018). However, many investigations (Szegedy et al., 2013; Goodfellow et al., 2015) show that DNNs are vulnerable to adversarial examples, i.e., images added by some elaborately designed perturbations that are imperceptible to human eyes may lead to the model's misclassification. At present, various techniques (Goodfellow et al., 2015; Madry et al., 2017; Carlini & Wagner, 2017) have been proposed to generate adversarial examples. Meanwhile, countermeasures (Madry et al., 2017; Zhang et al., 2019; Wong et al., 2020) have been proposed to defend against adversarial examples. However, most of the methods focus on weight optimization, while neglecting the influence of network structures. For example, adversarial training (AT) (Madry et al., 2017), as one of the most effective defensive methods, only optimizes the weights by adversarial examples. Nevertheless, recent studies reveal that robustness is highly related to a structure (Guo et al., 2020). We believe that a fixed structure may limit the further improvement of robustness.

Network Architecture Search (NAS) (Zoph & Le, 2017; Liu et al., 2019) is a high-performance automatic network structure design technology. In this paper, we propose Robust Network Architecture Search (RNAS) based on Differentiable Architecture Search (DARTS) (Liu et al., 2019), which can stably search for the network structure with high robustness. We define the network vulnerability specifically for the cell-based search process (Liu et al., 2019; Xu et al., 2019). First, we define channel vulnerability by the KL divergence of the output distribution of the adversarial examples and their corresponding clean examples in a channel. Then, we define cell vulnerability as the sum of the channel vulnerabilities in the cell's output layer. At last, we define network vulnerability by the cell vulnerability. With the above definitions, we constrain the search to get a more robust structure. Inspired by DARTS, we transform the original bilevel single-objective optimization into a bilevel multi-objective optimization by imposing network vulnerability constraints. Then, we simplify this optimization into a constrained single-objective optimization that can be solved by our proposed iterative algorithm. We evaluate its robustness on CIFAR-10/100, and SVHN based on extensive comparisons.

Our contributions are summarized as follows:

- We propose a network vulnerability metric based on the training deviation between clean examples and adversarial examples.
- We propose RNAS based on the vulnerability metric to obtain a robust network architecture. Meanwhile, we propose a noval algorithm to solve the constrained multi-objective optimization of RNAS.
- Extensive experiments show that RNAS achieves state-of-the-art performance in robust accuracy on several public datasets.

2 Related Works

Adversarial Attacks and Defends. Adversarial attack refers to a process of deceiving the target model by applying a tiny perturbation to the original input, i.e., adversarial examples. According to the available information, adversarial attacks are divided into white-box attacks (Zugner et al., 2018; Moosavi-Dezfooli et al., 2017) and black-box attacks (Papernot et al., 2017; Tu et al., 2019). Currently, the most classic white-box attack methods contain: Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015), Projected Gradient Descent (PGD) (Madry et al., 2017) and Carlini & Wagner (C&W) (Carlini & Wagner, 2017). Recently, Croce & Hein (2020a) proposed a reliable and stable attack method: AutoAttack (AA). It is an automatic parameter-free method for robustness evaluation. Four attack methods are integrated in AutoAttack, including three white-box attacks: APGD (Croce & Hein, 2020a) with cross entropy loss, targeted APGD with difference-of-logits-ratio loss and targeted FAB (Croce & Hein, 2020b), and a black-box attack: SquareAttack (Andriushchenko et al., 2020). Various adversarial defense methods have been proposed to improve the robustness of DNN against adversarial attacks, such as random smoothing (Lecuyer et al., 2019), defensive distillation (Papernot et al., 2016), and adversarial training (Madry et al., 2017; Zhang et al., 2019).

NAS For Robustness Network. NAS is proposed to automatically design the network architecture to replace the traditional manually-designed method. Representative techniques include reinforcement learning (Zoph & Le, 2017), evolutionary algorithms (Real et al., 2019), and differentiable approaches (Liu et al., 2019). One of the most representative differentiable methods is DARTS (Liu et al., 2019), which conducts search and evaluation at the same time. Though NAS achieves excellent performance by automatically searching the network architecture, the robust accuracy of the obtained model is neglected (Devaguptapu et al., 2020).

At present, researchers begin to focus on searching a more robust network architecture through NAS (Guo et al., 2020). They proved that robustness has a strong correlation with structure. Dong et al. (2020) discussed the relationship between robustness, Lipschitz constant and architecture parameters. They proved that proper constraints of the architecture parameter can reduce Lipschitz constant, thereby improve robustness. Hosseini et al. (2021) defined two differentiable metrics to measure the architecture robustness based on verifiable lower bounds and Jacobian norm bounds. The robust architecture is obtaind by maximizing the robustness metrics.

3 Preliminary

In our work, DARTS is used as the basic framework, which is a differentiable search framework, and its search space is defined based on the cell. The cell is defined as a directed acyclic graph (DAG) with N nodes $\{x_0, x_1, \ldots, x_{N-1}\}$, where each node represents a layer in the network. We define an operation space $\mathcal{O}$, where each element $o(\cdot) \in \mathcal{O}$ represents an operation in the layer (3×3 convolution, pooling, zero operation, etc.). Within a cell, the goal is to select one operation in $\mathcal{O}$ to connect each pair of nodes. The information flow between node i and node j is represented as edge $f_{(i,j)}$, which is composed of operations weighted by the architecture parameter $\alpha^{(i,j)}$ and denoted as:

$$f_{i,j}(x_i) = \sum\nolimits_{o \in \mathcal{O}_{i,j}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}_{i,j}} \exp(\alpha_{o'}^{(i,j)})} \cdot o(x_i) \tag{1}$$

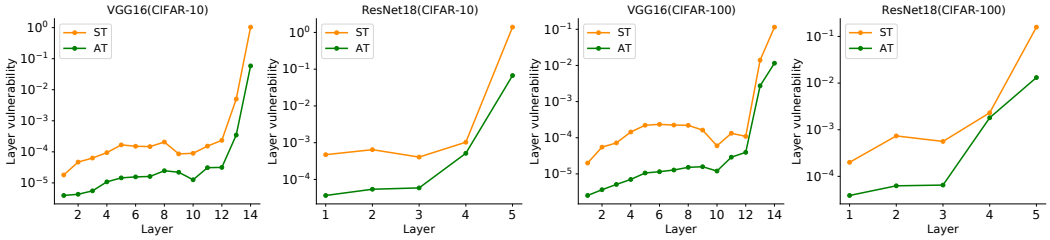


Figure 1: The layer vulnerability of VGG16 and ResNet18 on CIFAR-10/100.

where x_i is the output of the i-th node and $\alpha_0^{(i,j)}$ is the weight of operation $o(x_i)$. A node's input is the sum of all outputs of its previous nodes, i.e., $x_j = \sum_{i<j} f_{i,j}(x_i)$. The output of the entire cell is $x_{N-1} = concat(x_0, x_1, \dots x_{N-2})$, where $concat(\cdot)$ represents concatenating all channels. A proxy network on the search process is constructed by m cells.

The parameter of the operations on the edges is called as operation parameter θ. The search space of DARTS is differentiable so that θ and α can be updated with gradient alternately during the search in an end-to-end manner. When the search converges, we retain the operation with the largest α in each edge $f_{(i,j)}$ to compose the final cell structure. The obtained cell is taken as the basic unit to form the target network by stacking multiple cells together. To obtain the optimal structure parameter α, we define the optimization problem of DARTS as follows (Liu et al., 2019):

$$\begin{aligned} &\min_{\alpha} \mathcal{L}_{val}(\theta^*(\alpha), \alpha) \\ &\text{s.t.} \theta^*(\alpha) = \arg\min_{\theta} \mathcal{L}_{train}(\theta, \alpha) \end{aligned} \tag{2}$$

where $\mathcal{L}_{train}$ and $\mathcal{L}_{val}$ represent training loss and validation loss, respectively. More technical details can be referenced to (Liu et al., 2019).

4 Methodology

We propose Robust Network Architecture Search (RNAS) to find a robust network architecture. First, we define the network vulnerability to guide the search process. Then, based on the search process of DARTS, we apply the network vulnerability as a constraint to the architecture parameter α to obtain a robust structure. We describe RNAS as a multi-objective bilevel optimization problem and propose an iterative optimization algorithm to solve the optimal α.

4.1 Network Vulnerability Constraint

We use the expectation of the KL divergence (Kullback & Leibler, 1951) to measure the deviation between adversarial examples and clean examples on the feature map. We train two kinds of models: adversarial training (AT) and standard training (ST). The deviation between adversarial examples and clean examples on AT and ST is demonstrated in Figure 1. Compared with standard training, the deviation of the model with adversarial training becomes significantly smaller, which indicates that adversarial training improves the robust accuracy by reducing the deviation. Inspired by this observation, our main idea is to reduce the deviation to weaken the influence of adversarial examples on the model. To clarify the above idea, we formally define the vulnerability of channel, layer, cell, and network as follows.

Channel Vulnerability: The vulnerability of the k-th channel of the l-th layer is defined as:

$$F(z^{(l,k)}, \tilde{z}^{(l,k)}) = \mathbb{E}_{(x,y)\sim\mathcal{D}} KL(z^{(l,k)}, \tilde{z}^{(l,k)}) \tag{3}$$

where $z^{(l,k)}$ represents the feature value of the k-th feature map in the l-th layer of the clean example. Similarly, $\tilde{z}^{(l,k)}$ represents the adversarial case.

Layer Vulnerability: We define the layer vulnerability as the mean of the *channel vulnerability* of one layer, the l-th layer vulnerability is defined as:

$$f_l = \frac{1}{N^{(l)}} \sum_{k=1}^{N^{(l)}} F(z_i^{(l,k)}, \tilde{z}_i^{(l,k)}) \tag{4}$$

where $N^{(l)}$ is the number of the feature maps of the l-th layer.

Since the search space is defined based on cells, we define cell vulnerability, and further define network vulnerability based on cell vulnerability.

Cell Vulnerability: We define cell vulnerability as *layer vulnerability* of the output layers of the corresponding cell. The i-th cell vulnerability is defined as:

$$f_i^{(o)} = \frac{1}{N_i^{(o)}} \sum_{i=1}^{N_i^{(o)}} F(z_i^{(o,k)}, \tilde{z}_i^{(o,k)}) \tag{5}$$

where $z_i^{(o,k)}$ is the feature value of the k-th feature map in the output layer of the i-th cell of the clean example, similarly $\tilde{z}_i^{(o,k)}$ represents the adversarial case and $N_i^{(o)}$ is the number of the feature map in the output layer of the i-th cell.

Network Vulnerability: The mean of all *cell vulnerabilities* is denoted as the network vulnerability:

$$F(f_\theta(x), f_\theta(\tilde{x})) = \frac{1}{M} \sum_{i=1}^{M} f_i^{(o)} \tag{6}$$

where M is the number of the cell in the whole network.

4.2 Robust Network Architecture Search (RNAS)

In this paper, we take DARTS (Liu et al., 2019) as our basic framework. The original objective function of DARTS is presented in Equation 2, which only focuses on improving clean accuracy (Zhang et al., 2019). Our goal is to find robust cells and use them to construct a robust network. Thus, we need to add another object to the original objective function of DARTS, which achieves the minimal network vulnerability when updating architecture parameter α during the search process. If α is determined, we further use adversarial training to update the operation parameter θ of the new network architecture (corresponding to α). Briefly, the goal of RNAS is to minimize the validation loss and network vulnerability under the adversarial attack by searching for the architecture parameter α, and to obtain robust operation parameter θ through adversarial training. We formalize RNAS as a multi-objective bilevel optimization problem:

$$\begin{aligned} &\min_{\alpha} \left(\mathcal{L}_{val}(\theta^*(\alpha), \alpha) + \mathcal{L}_{val}^{adv}(\theta^*(\alpha), \alpha), \mathcal{F}(\alpha) \right) \\ &\text{s.t.} \theta^*(\alpha) = \arg\min_{\theta} \mathcal{L}_{train}^{adv}(\theta, \alpha) \end{aligned} \tag{7}$$

where $\mathcal{L}_{train}^{adv}$ and $\mathcal{L}_{val}^{adv}$ respectively represent adversarial training loss and adversarial validation loss and $\mathcal{F}(\alpha)$ represents the network vulnerability. In this multi-objective bilevel optimization, α is a upper-level variable and θ is a lower-level variable. However, this problem is non-trivial. We turn the network vulnerability into a constraint and set an upbound $H \in [0, +\infty)$ of the network vulnerability. Thus, Problem 7 is transformed into a single-objective optimization problem with two constraints, as shown in Problem 8. The operation parameter θ and the architecture parameter α are alternately updated until they converge.

$$\begin{aligned} &\min_{\alpha} \mathcal{L}_{val}(\theta^*(\alpha), \alpha) + \mathcal{L}_{val}^{adv}(\theta^*(\alpha), \alpha) \\ &\text{s.t.} \theta^*(\alpha) = \arg\min_{\theta} \mathcal{L}_{train}^{adv}(\theta, \alpha) \\ &\mathcal{F}(\alpha) \leq H \end{aligned} \tag{8}$$

We project the architecture parameter α to the nearest point α_p in the feasible region that satisfies the network vulnerability constraint, then Problem 9 can be solved by a Lagrangian method.

$$\begin{aligned} &\min_{\alpha_p} \frac{1}{2} \left\| \alpha - \alpha_p \right\|_2^2 \\ &\text{s.t.} \mathcal{F}(\alpha_p) \leq H \end{aligned} \tag{9}$$

Algorithm 1 Search Process of RNAS.

Input: Dataset $\mathcal{D}$, training epochs E, training iteration T, randomly initialized operation parameters θ and randomly initialized weights in mixed operation set $\mathcal{O}$.
Output: Learned architecture parameter α_p.
Phase I: θ-warm up

while $e \leq 15$ **do**
  for $t = 1$ to T **do**
    keep α fixed, and obtain θ^{t+1} by gradient descent with $\nabla_\theta \mathcal{L}_{train}^{adv}\left(\theta^t, \alpha\right)$
  end for
  $e \leftarrow e + 1$
end while

Phase II: Search for Robust Architectures

while *not converged* **or** $15 < e \leq E$ **do**
  $Step1$: unconstrained searching
  for $i = 1$ to u **do**
    keep α^t fixed, and obtain θ^{t+1} by gradient descent with $\nabla_\theta \mathcal{L}_{train}^{adv}\left(\theta^t, \alpha^t\right)$
    keep θ^{t+1} fixed, and obtain α^{t+1} by gradient descent with $\nabla_\theta \mathcal{L}_{val}^{adv}\left(\theta^{t+1}, \alpha^t\right)$
  end for
  $Step2$: vulnerability constrained for α.
  for $t = u$ to T **do**
    Get the projection of α outputting α_p at the end of *Phase II*
    Update α_p
  end for
  $e \leftarrow e + 1$
end while

The whole process is divided into two *phases*. In *the first warm-up phase*, we only update the operation parameters θ since it is randomly initialized at the beginning, which does not contain much valuable knowledge to guide the search process. *The second search phase* is divided into two *steps*, $Step$ 1 is unconstrained search, in which a better architecture is searched freely in a larger parameter space. In this step, the objective function of RNAS is the same as that of DARTS. According to the first-order approximation method in DARTS, θ and α are alternately updated by gradient descent. In $Step$ 2, we apply network vulnerability constraint to the architecture parameter α. In practice, we project α to the nearest point α_p of the feasible set, as shown in Equation 9. The algorithm is presented as Algorithm 1.

The advantages of Algorithm 1 are as follows: in *Phase I*, the operation parameters θ are warmed up to provide a stable network for further searching. In $Step$ 1 of *Phase II*, the weight and architecture are jointly optimized by adversarial examples to determine a reasonable projection starting point of α in $Step$ 2. In $Step$ 2, we apply network vulnerability constraint to α to achieve a "low-feature-distortion" network architecture. When the inputs are adversarially perturbed, the network vulnerability constraint can restrain the distortion by minimizing the deviation between the latent features of clean examples and adversarial examples. After $Step$ 2, the algorithm will return to $Step$ 1 to search the architecture in a larger parameter space. Therefore, even if a sub-optimal architecture is obtained in the projection step, the unconstrained search of $Step$ 1 can still learn a better one. In addition, the constraint of vulnerability can be adjusted by the upper bound H to make the search more flexible.

5 EXPERIMENT

We first search the network architecture by RNAS on CIFAR-10, then transfer the obtained architecture to SVHN and CIFAR-100 datasets. Extensive experiments are conducted on CIFAR-10, CIFAR-100, and SVHN to evaluate the effectiveness of RNAS under various adversarial attacks. Our model significantly outperforms the baselines and achieves the highest robustness.

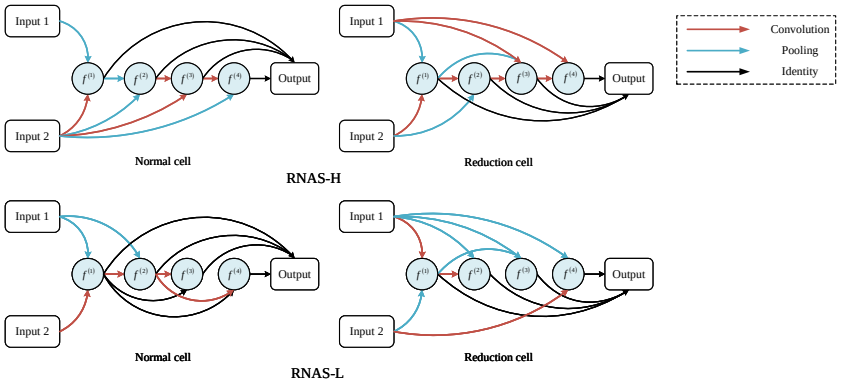


Figure 2: Architecture of the cells obtained on CIFAR-10.

5.1 Experimental Setup

Datasets: (1) CIFAR-10/100: Each dataset consists of 60K images, all of which are of a spatial resolution of 32 × 32. These images are equally distributed over 10/100 classes, with 50K training and 10K testing images (Krizhevsky & Hinton, 2009); (2) SVHN: This dataset (SVHN, Street View House Number Dataset) is derived from Google Street View door numbers, which consists of 73,257 images for training and 26,032 images for testing. These images are distributed over 10 classes of digits, and all of which size are 32 × 32 (Netzer et al., 2011).

Searching: We search the robust architecture on CIFAR-10. We divide the training set into two parts with the same size and respectively optimize the architecture parameters and the operation parameters. The search space contains 8 candidate operations: 3 × 3 and 5 × 5 separable convolutions, 3 × 3 and 5 × 5 dilated separable convolutions, 3 × 3 max pooling, 3 × 3 average pooling, skip connection, and zero operation. The proxy network consists of 8 cells: 6 normal cells and 2 reduction cells. Each cell has 6 nodes. We use SGD with momentum to train the model for 60 epochs with a batch size of 128. The initial learning rate is 0.01 with a momentum of 0.9, weight decay is 0.0003, and a cosine learning rate decay is used to update the proxy network weights. Architecture parameters are updated through Adam with a learning rate of 0.0006 and a weight decay of 0.001.The H is 0.0001

Training: After the search phase, we obtain the normal cell and the reduction cell in Figure 2. Then we adversarially train the target network on the entire dataset. The adversarial examples are generated by PGD and set the total perturbation size $\epsilon = 8/255$, the number of attack iteration is 7 with a step size of 2/255. The training phase took 300 epochs with a batch size of 128. We used SGD with momentum, where the initial learning rate is 0.01 with a momentum of 0.9, weight decay is 0.0003 and a cosine learning rate decay is used to update the network weights.

Evaluation: All models are fully trained for 300 epochs to be evaluated and all parameters are consistent with those of RNAS. We use various adversarial attack to generate adversarial examples to evaluate the models, including FGSM (Goodfellow et al., 2015), PGD (Madry et al., 2017), and C&W (Carlini & Wagner, 2017). For a more comprehensive evaluation, we introduce AutoAttack (AA) (Croce & Hein, 2020a). The attack settings are as follows: 1) FGSM attack with $\epsilon = 0.03$ (8 / 255); 2) PGD attack with $\epsilon = 0.03$ (8 / 255), attack iterations of 20, and a step size of 2 / 255; 3) C&W attack with $c = 0.5$ and attack iterations of 100; 4) AA with l_∞-norm and $\epsilon = 0.03$ (8/255).

5.2 Results on CIFAR-10

Figure 2 illustrates the architecture of the normal cell and the reduction cell obtained on CIFAR-10. Through setting different H, we obtain two architecture (RNAS-H and RNAS-L). We observe that the operations between nodes in RNAS-H and RNAS-L are intensive, which is consistent with the property of the robust architecture obtained in (Guo et al., 2020). We use 20 cells to construct the target network and evaluate it through standard training and adversarial training respectively under various adversarial attacks. The comparison results are summarized in Table 1 and Table 2.

Table 1: Comparison results of different models with adversarial training on CIFAR-10.

Model	Params (M)	Clean (%)	FGSM (%)	PGD (%)	C&W (%)	AA (%)
VGG16(Simonyan & Zisserman, 2015)	14.7	80.08	52.85	47.50	41.80	42.10
ResNet18(He et al., 2016)	11.2	82.63	55.12	49.81	42.48	44.43
DenseNet121(Huang et al., 2017b)	7.0	84.85	53.35	47.31	42.68	42.18
NasNet(Zoph et al., 2018)	4.3	80.61	54.19	50.25	42.97	45.33
AmoebaNet(Real et al., 2019)	3.2	83.41	54.44	42.95	39.21	35.32
PNAS(Liu et al., 2018)	4.5	85.08	58.79	47.70	40.15	43.03
SNAS(Xie et al., 2019)	2.7	82.56	54.39	46.03	40.36	43.55
DARTS(Liu et al., 2019)	3.3	83.75	55.75	44.91	41.25	39.98
P-DARTS(Chen et al., 2019)	3.4	82.65	53.27	42.72	41.03	37.22
PC-DARTS(Xu et al., 2019)	3.6	83.94	52.67	41.92	39.25	37.53
MobileNetv2(Sandler et al., 2018)	2.3	81.04	53.66	49.40	41.26	42.29
ShuffleNetv2(Ma et al., 2018)	1.3	81.25	49.10	42.10	40.34	36.78
SqueezeNet(Iandola et al., 2017)	0.7	77.65	51.21	44.22	39.66	38.58
RACL(Dong et al., 2020)	3.6	83.98	57.44	49.34	43.13	44.59
RobNet(Guo et al., 2020)	6.9	78.57	54.98	49.44	42.84	45.01
DSRNA(Hosseini et al., 2021)	5.4	83.84	59.89	50.39	43.94	46.78
RNAS-L	3.5	**85.13**	56.91	52.88	44.65	46.82
RNAS-H	3.2	81.68	**60.61**	**53.72**	**45.27**	**47.73**

Table 2: Comparison results of different models with standard training on CIFAR-10.

Model	Params (M)	Clean (%)	PGD (%)	Model	Params (M)	Clean (%)	PGD (%)
NasNet	4.3	97.37	0.52	VGG16	14.7	92.64	0
AmoebaNet	3.2	97.39	0.35	ResNet18	11.2	93.02	0
PNAS	4.5	96.70	0.28	DenseNet	7.0	95.04	0.11
SNAS	2.7	97.02	0.12	MobileNetv2	2.3	94.43	0
DARTS	3.3	**97.41**	0.16	ShuffleNetv2	1.3	93.45	0
PC-DARTS	3.6	97.32	0.18	**RNAS-L**	3.5	96.86	**3.91**

As shown in Table 1, through adversarial training, the architecture obtained by RNAS has a better robust performance than other models. (1) Compared with various manually designed baseline models, our model achieves better robust accuracy. (2) Compared with NAS baselines, our model achieves better performance under various adversarial attacks. The main reason is that RNAS applies network vulnerability to the cell-based search process. Therefore, RNAS can guarantee higher accuracy and robust architecture. (3) Compared with existing NAS-based robust search methods, RNAS-L and RNAS-H both achieve better performance. Under four adversarial attacks: FGSM, PGD, C&W, and AutoAttack, RNAS-L respectively improves the robust accuracy by -2.98, +2.49, +0.71, and +0.04 compared with the state-of-the-art DSRNA, while RNAS-H respectively improves by +0.72, +3.33, +1.33, and +0.95. This indicates a high correlation between network vulnerability and robustness. In addition, RNAS achieves higher clean accuracy by +1.15, +6.56, and +1.29 compared with RACL, RobNet, and DSRNA respectively.

Except for adversarial training, network architecture as well contributes to robustness, which is evaluated by standard training instead of adversarial training. The results are shown in Table 2. Even only with standard training, RNAS is still more robust than other network architecture, that is, even without adversarial training, RNAS still has a robust property. In addition, the clean accuracy of our method is yet closed to that of the optimal NAS.

Table 3: Comparison results of different models with adversarial training on CIFAR-100.

Model	Params (M)	Clean (%)	FGSM (%)	PGD (%)	C&W (%)	AA (%)
VGG16	34	47.50	21.17	15.85	14.06	14.88
ResNet18	11.2	55.02	24.56	18.89	16.04	17.82
NasNet	4.3	59.64	29.83	23.10	20.41	22.06
AmoebaNet	3.2	58.14	29.00	24.17	19.54	21.74
PNAS	4.5	58.24	29.75	24.33	19.47	22.09
SNAS	2.7	**61.59**	29.14	22.26	21.28	20.74
DARTS	3.3	61.14	28.79	24.72	19.45	21.54
PC-DARTS	3.6	60.71	29.29	23.10	20.72	21.60
MobileNetv2	2.3	48.51	22.89	19.78	15.83	16.97
ShuffleNetv2	1.3	51.10	22.16	17.64	15.13	16.26
RACL	3.6	60.32	31.46	24.23	21.73	21.78
RobNet	6.9	57.64	30.23	24.11	21.32	21.98
DSRNA	5.4	60.44	32.03	25.11	21.52	22.59
RNAS-L	3.5	59.71	**32.52**	**25.73**	**22.82**	**23.64**
RNAS-H	3.2	57.33	32.02	25.82	21.68	22.81

5.3 Results on CIFAR-100

To further evaluate the effectiveness of RNAS, we transfer the model to CIFAR-100. Specifically, we use 20 cells obtained on CIFAR-10 to construct a network and adversarially retrain the network on CIFAR-100. In Table 3, compared with most of the baselines, RNAS obtained on CIFAR-10 can still improve the robustness after adversarial training on CIFAR-100. In addition, we observe that on CIFAR-100, the robustness improvement of RNAS-H is not as high as it on CIFAR-10. The reason is that the architecture search process is dataset-dependent, and RNAS-H is obtained under a higher constraint of vulnerability, which may "overfit" the dataset. After being transferred to CIFAR-100, the performance of the architecture obtained on CIFAR-10 decreases a little. Despite this, RNAS-H still achieves a comparative or even better performance compared with state-of-the-art methods. The differences in performance between RNAS-L and RNAS-H are due to the intensity of the constraints on the vulnerability, which will be discussed in Subsection 5.5.

5.4 Results on SVHN

We transfer the model to the SVHN dataset, that is, we use 20 cells obtained on CIFAR-10 to construct a network and adversarially retrain the network on SVHN. In Table 4, similar results are shown as those on CIFAR-10/100, which as well demonstrates the overfit of RNAS-H to the data.

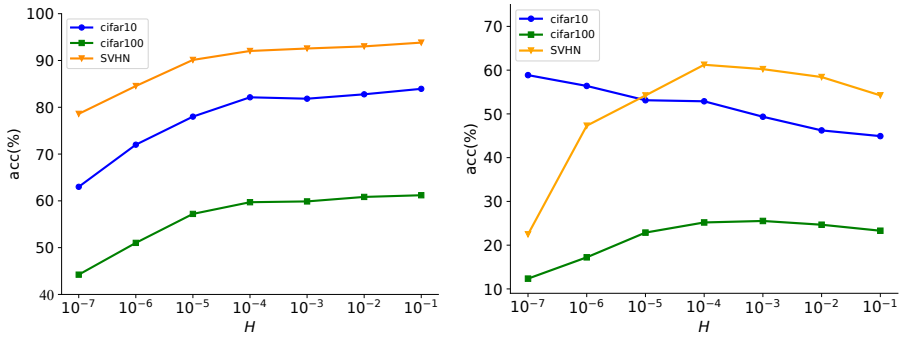
5.5 Upper Bound Parameter H of Vulnerability

Recall that H is the upper bound of the constraint on the network vulnerability, which is to control the intensity of the constraint. The larger the H value, the looser constraint on the vulnerability, vice versa. $H = 0$ means that it does not allow any deviation occurring in the cell output between the clean examples and their corresponding adversarial examples. Intuitively, the smaller H can ensure a more robust architecture in the search process. However, experiments show that too small an H may lead to an overfit to the training data, and further reduce the model's generalization ability.

The network architecture search is time-consuming, and the adversarial training introduced into the RNAS search process will further overload computation costs. In practice, the search is conducted on a small proxy dataset first, then the obtained architecture is transferred to the target dataset. Thus, transferability is a valuable property of the obtained models. Furthermore, the transferability is influenced by H. Figure 3 shows how H influences the model. As H decreases (i.e., the constraint strengthens), the PGD robust accuracy on CIFAR-10 gradually increases, while on CIFAR-100 and

Table 4: Comparison results of different models with adversarial training on SVHN.

Model	Params (M)	Clean (%)	FGSM (%)	PGD (%)	C&W (%)	AA (%)
VGG16	14.7	90.33	60.25	51.70	48.48	39.41
ResNet18	11.2	91.64	61.32	53.01	49.31	41.82
NasNet	4.3	90.11	61.18	53.37	50.44	45.20
AmoebaNet	3.2	91.46	62.01	44.20	48.56	38.64
PNAS	4.5	92.13	62.84	50.91	49.80	39.65
SNAS	2.7	91.05	61.21	50.73	50.01	39.54
DARTS	3.3	91.33	61.97	48.30	49.10	43.01
PC-DARTS	3.6	91.51	60.01	46.26	49.47	40.12
MobileNetv2	2.3	88.94	59.82	50.67	48.86	43.90
ShuffleNetv2	1.3	89.26	57.26	46.50	47.57	39.24
RACL	3.6	91.80	62.46	52.02	51.02	46.54
RobNet	6.9	89.39	60.90	51.89	50.81	46.84
DSRNA	5.4	**94.01**	62.77	53.48	50.79	47.36
RNAS-L	3.5	92.03	**62.95**	**53.82**	**52.80**	**48.62**
RNAS-H	3.2	90.22	61.70	53.08	51.32	47.01



(a) Clean Accuracy (b) Robust Accuracy

Figure 3: The clean accuracy and robust accuracy of RNAS with different H value on different datasets. We use different H to search the architecture on CIFAR-10 and transfer the obtained model to CIFAR-100 and SVHN.

SVHN, the PGD robust accuracy falls rapidly after rising. This indicates that too small an H makes the model overfit CIFAR-10 and reduces the generalization ability of the model; too large an H is not effective in constraint the vulnerability of the network. In conclusion, the advantage of RNAS is that we can tune H to adapt different datasets.

6 Conclusion

Our proposed RNAS aims to improve both accuracy and robustness on the basis of network architecture search. We impose a network vulnerability metric on the search process. This network vulnerability metric is bound by a parameter H. The experiments show that through tuning H, RNAS can be adaptive to specific dataset. The experimental results also show that RNAS not only improve the robustness, but also achieve high accuracy close to standard training. Compared to existing classic methods, our model achieves state-of-the-art performance. In addition, even without adversarial training, our method still shows robustness.

ACKNOWLEDGMENTS

This work is supported by National Key R&D Program (No.2018YFB2100400), and the Scientific Project of Zhejiang Provincial Science and Technology Department (No.LGG19F030001, No.LGF20F020007).

REFERENCES

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*, pp. 484–501. 2020.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy*, pp. 39–57. 2017.

Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision*, pp. 801–818. 2018.

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

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning*, pp. 2206–2216. 2020a.

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

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.

Minjing Dong, Yanxi Li, Yunhe Wang, and Chang Xu. Adversarially robust neural architectures. *arXiv preprint arXiv:2009.00902*, 2020.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1097–1105. 2014.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*. 2015.

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 Conference on Computer Vision and Pattern Recognition*, pp. 628–637. 2020.

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*, pp. 770–778. 2016.

Ramtin Hosseini, Xingyi Yang, and Pengtao Xie. DSRNA: Differentiable Search of Robust Neural Architectures. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6196–6205. 2021.

Gao Huang, Zhuang Liu, Laurens Maaten, and Kilian Q. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017a.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017b.

Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50× fewer parameters and <0.5MB model size. In *International Conference on Learning Representations*. 2017.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. In *Technical Report*. 2009.

Alex Krizhevsky, Sutskever Ilya, and Hinton Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1097–1105. 2012.

S. Kullback and R. A. Leibler. On information and sufficiency. In *The annals of mathematical statistics*, pp. 79–86. 1951.

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*, pp. 656–672. 2019.

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

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable Architecture Search. In *International Conference on Learning Representations*. 2019.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. In *Proceedings of the European Conference on Computer Vision*, pp. 116–131. 2018.

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.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2574–2582. 2017.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Ng. Andrew Y. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*. 2011.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy*, pp. 582–597. 2016.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical Black-Box Attacks against Deep Learning Systems using Adversarial Examples. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp. 506–519. 2017.

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*, pp. 4780–4789. 2019.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520. 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large scale image recognition. In *International Conference on Learning Representations*. 2015.

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.

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*, pp. 742–749. 2019.

Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*. 2020.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: Stochastic neural architecture search. In *International Conference on Learning Representations*. 2019.

Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, and Xin Chen. PC-DARTS: Partial Channel Connections for Memory-Efficient Architecture Search. In *International Conference on Learning Representations*. 2019.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pp. 7472–7482. 2019.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*. 2017.

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*, pp. 8697–8710. 2018.

Daniel Zugner, Amir Akbarnejad, and Stephan Gunnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2847–2856. 2018.