Robust Network Architecture Search via Feature Distortion Restraining

Yaguan Qian[1[0000−0003−4056−9755]], Shenghui Huang[1[0000−0001−9838−7095]], Bin Wang[2⋆], Xiang Ling[3], Xiaohui Guan[4], Zhaoquan Gu[5], Shaoning Zeng[6], Wujie Zhou[1], and Haijiang Wang[1]

[1] Zhejiang University of Science and Technology, Hangzhou Zhejiang 310023, China qianyaguan@zust.edu.cn
[2] Zhejiang Key Laboratory of Multidimensional Perception Technology, Application and Cybersecurity, Hangzhou Zhejiang 310052, China wbin2006@gmail.com
[3] Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
[4] Zhejiang University of Water Resources and Electric Power, Hangzhou Zhejiang 310023, China
[5] Cyberspace Institute of Advanced Technology (CIAT), Guang Zhou University, Guangzhou Guangdong 510006, China
[6] Yangtze Delta Region Institute, University of Electronic Science and Technology of China, Huzhou Zhejiang 313000, China

Abstract. The vulnerability of Deep Neural Networks, i.e., susceptibility to adversarial attacks, severely limits the application of DNNs in security-sensitive domains. Most of existing methods improve model robustness from weight optimization, such as adversarial training. However, the architecture of DNNs is also a key factor to robustness, which is often neglected or underestimated. We propose Robust Network Architecture Search (RNAS) to obtain a robust network against adversarial attacks. We observe that an adversarial perturbation distorting the non-robust features in latent feature space can further aggravate misclassification. Based on this observation, we search the robust architecture through restricting feature distortion in the search process. Specifically, we define a network vulnerability metric based on feature distortion as a constraint in the search process. This process is modeled as a multi-objective bilevel optimization problem and a novel algorithm is proposed to solve this optimization. Extensive experiments conducted on CIFAR-10/100 and SVHN show that RNAS achieves the best robustness under various adversarial attacks compared with extensive baselines and SOTA methods.

Keywords: Adversarial examples; Network architecture search; Roubst architecture

1 Introduction

In recent years, Deep Neural Networks (DNNs) have shown excellent performance in various applications, such as image classification [20, 15], objective de-

⋆ Corresponding author

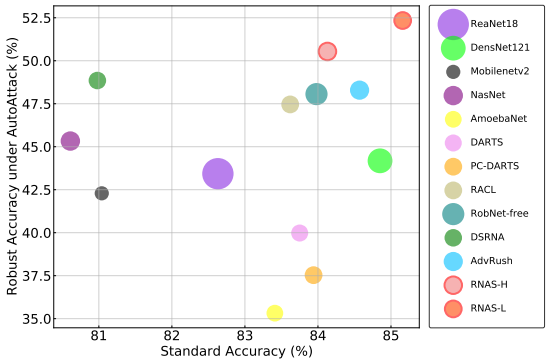


Fig. 1: Adversarial robustness, Standard accuracy, and parameter numbers for various architectures on CIFAR-10. All architectures are adversarially trained using 7-step PGD and evaluated by AutoAttack. The bubbles' size reflects the model parameters. RNAS's robustness and standard accuracy outperform other architectures, even with fewer parameters.

tection [9], and semantic segmentation [3]. However, many investigations [35, 10] show that DNNs are vulnerable to adversarial examples, i.e., images added by some elaborately designed imperceptible perturbations may lead to the model's misclassification. At present, various techniques [10, 26, 2] have been proposed to generate adversarial examples. Meanwhile, countermeasures [26, 40, 37] have been proposed to defend against adversarial examples. However, most of the methods focus on weight optimization, while neglects the influence of network structures, *e.g.*, adversarial training (AT) [26]. Nevertheless, recent studies reveal that robustness is highly related to the network structure [11]. A fixed structure may limit the further improvement of robustness. Therefore, in this paper, our work focuses on searching for a robust network architecture.

To achieve higher robustness, we attempt to explore the relationship between network architecture and adversarial examples. Ilyas et al. [18] claimed that the existence of adversarial examples is due to the non-robust features of data, i.e., the intrinsic property of data. However, we suggest that adversarial perturbation misleads the network through distorting the non-robust features in latent feature space, which is also a property of networks. In other words, different network architectures and their weights have different defensive abilities to this distortion. Hence, we propose Robust Network Architecture Search (RNAS) based on Differentiable Architecture Search (DARTS) [24] to obtain a robust architecture through restraining the latent feature distortion. Specifically, we measure the latent feature distortion by the difference of feature distribution between clean and adversarial examples. This difference is quantified by KL divergence [21]. Based on the feature distortion of each cell (a basic component of a network in DARTS), we define network vulnerability. Specifically, we first define channel vulnerability by the KL divergence between the channel output distribution of adversarial

examples and their corresponding clean examples. Then, we define cell vulnerability as the mean of channel vulnerabilities in the cell's output layer. Finally, we define network vulnerability as the mean of all cells' vulnerabilities. With the above definitions, we can obtain a more robust architecture by constraining network vulnerability in a search process. Inspired by DARTS, we transform an original single-objective bilevel optimization into a multi-objective bilevel optimization by imposing network vulnerability constraints. Then, we simplify this optimization and solve it with our proposed iterative algorithm. We evaluate the robustness of RNAS on CIFAR-10/100 [19], SVHN [29], and Tiny-ImageNet based on extensive comparisons. Our contributions are summarized as follows:

- We suggest that the distortion of non-robust features in latent feature space plays a key role in misclassification caused by adversarial examples. From this observation, we propose a network vulnerability metric based on the difference between latent feature distribution of clean examples and adversarial examples.
- We propose Robust Network Architecture Search (RNAS) using the network vulnerability metric as a constraint to obtain a robust network architecture. Meanwhile, we design an effective algorithm to solve this constrained multi-objective optimization.
- Extensive experiments conduct on several public datasets show that RNAS achieves SOTA performance in robust accuracy compared with RobNet-free, DSRNA, and AdvRush.

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 [44, 28] and black-box attacks [30, 36]. Currently, the most classic white-box attack methods contain: Fast Gradient Sign Method (FGSM) [10], Projected Gradient Descent (PGD) [26] and Carlini & Wagner (C&W) [2]. Recently, Croce and Hein [6] 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 [6] with cross entropy loss, targeted APGD with difference-of-logits-ratio loss and targeted FAB [5], and a black-box attack: SquareAttack [1]. Various adversarial defense methods have been proposed to improve the robustness of DNN against adversarial attacks, such as random smoothing [22], defensive distillation [31], and adversarial training [26, 40].

NAS for robustness network. NAS is proposed to automatically design the network architecture to replace traditional manually-designed methods. Representative techniques include reinforcement learning [42], evolutionary algorithms [32], and differentiable approaches [24]. One of the most representative

differentiable methods is DARTS [24], which conducts search and evaluation at the same time. Though NAS achieves excellent performance by automatically searching the network architecture, it neglects the robust accuracy of the obtained model [7].

At present, researchers focus on searching a more robust network architecture through NAS [11]. They proved that robustness has a strong correlation with structure. Dong et al.[8] 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.[14] defined two differentiable metrics to measure the architecture robustness based on verifiable lower bounds and Jacobian norm bounds. The search process is based on the maximization of the robustness metrics.

3 Preliminary

In our work, we use DARTS as our basic framework. DARTS is a differentiable search framework and its search space is defined on cells. A 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. In an operation space $\mathcal{O}$, each element $o(\cdot) \in \mathcal{O}$ represents an operation in a layer (3×3 convolution, pooling, zero operation, etc.). Within a cell, the goal of DARTS is to select an operation in $\mathcal{O}$ to connect each node-pair. The information flow between node i and node j is represented as an edge $f_{(i,j)}$, which is composed of operations weighted by an architecture parameter $\alpha^{(i,j)}$, i.e.,

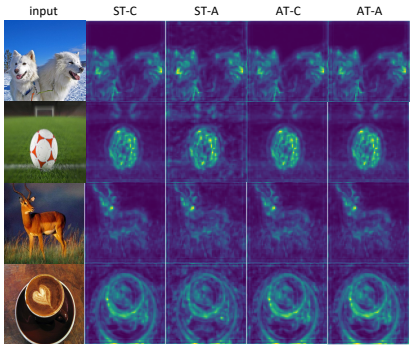
$$f_{i,j}(x_i) = \sum\nolimits_{o \in \mathcal{O}_{i,j}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}_{i,j}} \exp(\alpha_{o'}^{(i,j)})} \cdot o(x_i) \tag{1}$$

where x_i is the output of the i-th node and $\alpha_o^{(i,j)}$ is the weight of an 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 cell x_{N-1} is $concat(x_0, x_1, \ldots x_{N-2})$, where $concat(\cdot)$ represents concatenating all channels. A proxy network on the search process is constructed by m cells.

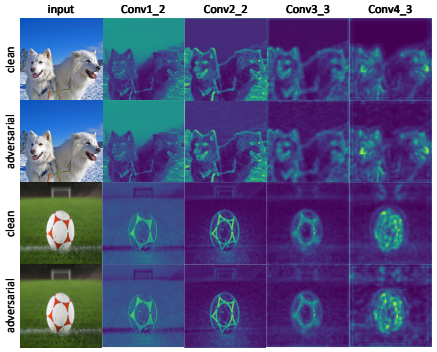
The operation parameter of $o(\cdot)$ is denoted by θ. The search space of DARTS is differentiable, so that θ and α can be alternately updated with gradients in the search process. When the search process converges, we retain the operation with the largest α in each edge $f_{(i,j)}$ to compose the final cell structure. The obtained cell is taken as a basic unit to form the target network by stacking multiple cells together. The optimization of α and θ are defined as follows [24]:

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

where $\mathcal{L}_{train}$ and $\mathcal{L}_{val}$ denote training loss and validation loss, respectively.



(a) Feature map visualization of the Conv4_3 layer of VGG16. ST and AT represent standard and adversarial training respectively; C and A represent clean and adversarial examples respectively.



(b) Feature map visualization of different layers of VGG16. A pair of clean example and adversarial example is compared by visualizing their feature maps in different layers.

Fig. 2: Feature map visualization.

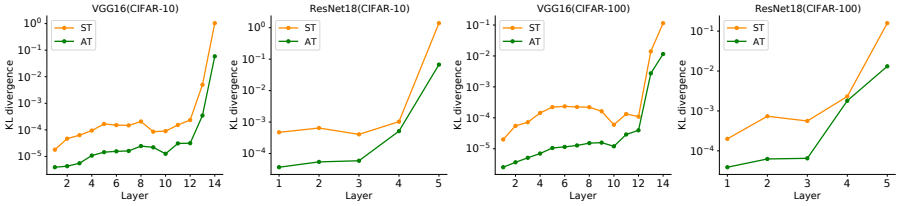


Fig. 3: The latent feature distortion of VGG16 and ResNet18 on CIFAR-10/100.

4 Method

First, we analyze the mechanism of adversarial examples from the perspective of feature distortion. Based on feature distortion, we define a network vulnerability metric to guide the search process. Taking DARTS as the basic framework of RNAS, we apply network vulnerability as a constraint to an architecture parameter α. We formulize RNAS as a multi-objective bilevel optimization. Through an iterative optimization algorithm, we can obtain a more robust network architecture than other methods.

4.1 Network vulnerability constraint

Ilyas et al. [18] claimed that in classification tasks, the network relies on non-robust features, which are highly predictive and imperceptible, to achieve high accuracy. However, it leads to adversarial perturbations exploiting this dependence. Non-robust features are highly sensitive to the feature distortion and only

a slight distortion will lead to misclassification. We speculate that such adversarial perturbations further aggravate a distortion in the latent feature space. To observe the process of latent feature distortion caused by the adversarial perturbation, we visualize the feature maps in Fig. 2. As illustrated in Fig. 2a, for a standard training model, the latent features of adversarial examples have an obvious distortion compared with that of clean examples. The worst distortion regions mainly exist in the non-object parts that contain many imperceptible non-robust features. On the contrary, for an adversarial training model, the distortion caused by adversarial examples is significantly weakened. In addition, Fig. 2b shows that the distortion becomes more obvious as layers become deeper. Hence, we believe that an adversarial example fools a model by enlarging the distortion of non-robust features.

To further quantify this distortion, i.e., to measure the latent feature difference between clean examples and adversarial examples in deep networks, we introduce the KL divergence [21] that is widely used to measure the difference between two distributions, e.g., the difference of feature distribution of different networks' outputs in knowledge distilling [13] and deep mutual learning [41]. In practice, we represent the distortion as a feature maps' KL divergence between clean examples and adversarial examples. We quantitatively analyze the distortion change in adversarial training models and standard training models, as shown in Fig. 3. As layers become deeper, the distortion increases by an order of magnitude. Meanwhile, the distortion of adversarial training models is significantly smaller than that of standard training models, which is consistent with the visualization in Fig. 2.

Based on the above analysis, the main idea of our method is to restrain feature distortion. If a network lacks resistance to feature distortion, the network will be vulnerable to adversarial examples. Hence, we define a model vulnerability metric based on feature distortion. Considering this metric as a constraint, we can search for a more robust network architecture. Then, we define the network vulnerability metric as follows.

Channel vulnerability: The vulnerability of the k-th channel in 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 $z^{(l,k)}$ denotes the feature value of the k-th feature map in the l-th layer of clean examples. Similarly, $\tilde{z}^{(l,k)}$ represents the adversarial case.

Layer vulnerability: We define the layer vulnerability f_l as the mean of all *channel vulnerabilities* in the l-th layer, i.e.,

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

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

From the observation in Fig. 2 and Fig. 3, the feature distortion increases as the network becomes deeper. Therefore, we should not only focus on the final

output distortion, but also concern the distortion changes in hidden layers of the network. Since the search space is defined based on cells, we first define cell vulnerability, and further define network vulnerability based on cell vulnerability.

Cell vulnerability: We define cell vulnerability $f_i^{(o)}$ as the mean of all *layer vulnerability* of the output layers in the i-th cell, i.e.,

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

where $N_i^{(o)}$ is the number of feature maps, $z_i^{(o,k)}$ is the feature value of the k-th feature map from clean examples, similarly, $\tilde{z}_i^{(o,k)}$ represents the adversarial case.

Network vulnerability: The network vulnerability is defined as the mean of all *cell vulnerabilities*, i.e.,

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

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

4.2 Robust Network Architecture Search (RNAS)

The DARTS only focuses on clean accuracy [40]. Our goal is to find robust cells and then use them to construct a robust network. Thus, we add network vulnerability to the original DARTS objective function. This add-on guarantees a minimal network vulnerability during the update of architecture parameter α in the search process. Once α is determined, we use adversarial training to update the operation parameter θ for a new architecture (corresponding to α). Briefly, the objective of RNAS is to minimize the validation loss and network vulnerability under adversarial attacks. The robust operation parameter θ is updated through adversarial training. We formalize RNAS as a multi-objective bilevel optimization problem:

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

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 an upper-level variable and θ is a lower-level variable. However, addressing this problem is non-trivial. We turn the network vulnerability into a constraint and set an upper bound $H \in [0, +\infty)$ of the network vulnerability. Thus, Problem 7 is transformed into a single-objective optimization problem with two constraints, as shown in Eq. 8. θ and α are al-

Algorithm 1 Robust Network Architecture Search (RNAS)

Input: Dataset $\mathcal{D}$, training epochs E, training iteration T.
Output: Learned architecture parameter α_p.
//*Phase I*: θ-warm up
Randomly initialized operation parameters θ and weights α in mixed operation set $\mathcal{O}$
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*: Robust Architecture Search
while *not converged* **or** $15 < e \leq E$ **do**
  //*Step*1: unconstrained searching
  for $t = 1$ to $T/2$ **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 $\nabla_\alpha \mathcal{L}_{val}^{adv}\left(\theta^{t+1}, \alpha^t\right)$
  end for
  //*Step*2: vulnerability constrained for α.
  for $T/2$ to T **do**
    $\alpha_p^t \leftarrow \Pi_{proj} \alpha^t$
    Update α_p^t
  end for
  $\alpha^t \leftarrow \alpha_p^t$
  $e \leftarrow e + 1$
end while

ternately updated until they converge.

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

Since Eq. 8 is a constrained optimization and the softmax function of α is non-convex, it is hard to obtain a closed-form solution. So we introduce a projection method to optimize the constraining function $\mathcal{F}(\alpha) \leq H$. We project α to the nearest point α_p in the feasible region that satisfies the network vulnerability constraint, as shown in Eq. 9, which can be solved by a Lagrangian method.

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

The whole process is divided into two *phases*. In *Phase I: warm-up*, we only update θ since it is randomly initialized at the beginning, which contains little valuable knowledge to guide the search process. *Phase II: search* is divided

into two *steps*, *Step* 1 is an unconstrained search, in which α can disobey the vulnerability constraint and 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, where θ and α are alternately updated by gradient descent. In *Step* 2, we project α to the nearest point α_p in the feasible set, where the objective of projection in Eq. 9 should satisfy the vulnerability constraint. At the end of *Step* 2, we assign α_p to α to make the algorithm return to *Step* 1. The algorithm of RNAS is presented as Algorithm 1.

The advantages of RNAS 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 search 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. In addition, the upper bound H can adjust the vulnerability constraint to make the search more flexible. The detailed discussion of H is in the Section 5.4

5 Experiment

We first use RNAS to search on CIFAR-10, then transfer the obtained architecture to SVHN, CIFAR-100 and Tiny-ImageNet. We conduct extensive experiments on CIFAR-10/100, SVHN and Tiny-ImageNet under various adversarial attacks to evaluate the effectiveness of RNAS. Our model significantly outperforms the baselines and achieves the highest robustness.

5.1 Experimental setup

Searching: When searching on CIFAR-10, we divide the training set into two equal parts. 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 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. Architecture parameters α are updated through Adam with a learning rate of 0.0006 and a weight decay of 0.001. $H = 0.0001$ in RNAS-H and $H = 0.00001$ in RNAS-L.

Training: After obtaining the normal cell and the reduction cell (as shown in Fig. 4) in the search process, we adversarially train the target network on the entire dataset. The adversarial examples are generated by PGD and the total perturbation size $\epsilon = 8/255$, the number of attack iterations is 7 with a step size

Table 1: Size and robust accuracy (%) of different architecture with PGD adversarial training on CIFAR-10. PGD^{20} and PGD^{100} refer to PGD attack with 20 and 100 iterations. The best result in each column is in bold, and the second best result is underlined.

Model	Params	AT						ST	
		Clean	FGSM	PGD^{20}	PGD^{100}	C&W	AA	Clean	FGSM
VGG16[34]	14.7M	80.08	52.85	47.50	46.66	41.80	42.10	92.64	46.35
ResNet18[12]	11.2M	82.63	54.12	48.81	48.50	42.48	43.43	94.64	49.72
DenseNet121[16]	7.0M	84.85	54.35	48.31	47.81	35.68	44.18	95.97	47.11
NasNet[43]	4.3M	80.61	54.19	50.25	49.63	42.97	45.33	97.33	50.03
AmoebaNet[32]	3.2M	83.41	54.44	42.95	42.80	39.21	35.32	<u>97.45</u>	41.60
PNAS[23]	4.5M	<u>85.08</u>	58.79	47.70	47.51	40.15	43.03	96.60	49.32
SNAS[38]	2.7M	82.56	54.39	46.03	45.97	40.36	43.55	97.18	50.01
DARTS[24]	3.3M	83.75	55.75	44.91	45.00	41.25	39.98	97.41	50.56
P-DARTS[4]	3.4M	82.65	53.27	42.72	42.77	41.03	37.22	97.40	54.51
PC-DARTS[39]	3.6M	83.94	52.67	41.92	42.50	39.25	37.53	**97.50**	52.75
MobileNetv2[33]	2.3M	81.04	53.66	47.40	46.79	41.26	42.29	94.23	47.32
ShuffleNetv2[25]	1.3M	80.25	49.10	42.10	40.25	40.34	36.78	91.48	44.58
SqueezeNet[17]	0.7M	78.65	51.21	44.22	40.56	39.66	28.58	86.72	31.90
RACL[8]	3.6M	83.62	57.25	50.02	49.86	44.13	47.64	96.42	49.29
RobNet-free[11]	5.6M	83.98	58.44	51.68	51.47	<u>46.07</u>	48.06	96.46	35.32
DSRNA[14]	3.5M	80.98	59.41	51.34	51.28	38.92	48.85	97.02	52.24
AdvRush[27]	4.2M	84.57	60.21	52.32	52.20	45.13	48.29	97.28	54.72
RNAS-H	3.5M	84.13	<u>61.90</u>	<u>53.48</u>	<u>53.35</u>	**50.74**	<u>50.54</u>	96.65	<u>63.23</u>
RNAS-L	3.2M	**85.16**	**62.61**	**54.85**	**53.70**	45.57	**52.34**	95.26	**65.32**

of 2/255. The training phase has 600 epochs with a batch size of 128. We use SGD with momentum, where the initial learning rate is 0.1 with a momentum of 0.9, weight decay is 0.0003 and a cosine learning rate decay is used.

Evaluation: All models are fully trained for 600 epochs and the setting is consistent with RNAS in the training phase. Adversarial examples used for evaluation are generated from FGSM [10], PGD [26], C&W [2] and AutoAttack (AA) [6]. The attack settings are as follows: 1) FGSM attack with $\epsilon = 0.031$ (8 / 255); 2) PGD attack with $\epsilon = 0.031$ (8 / 255), attack iterations of 20 and 100, and a step size of 2 / 255; 3) C&W attack with $c = 0.5$ and attack iterations of 100; 4) AA with $\epsilon = 0.031$ (8/255). All attacks are l_∞-bounded.

5.2 Results on CIFAR-10

Fig. 4 illustrates the architecture of the normal cell and the reduction cell obtained on CIFAR-10. We obtain two architectures through different H values: RNAS-H (high) and RNAS-L (low). We observe that the operations between

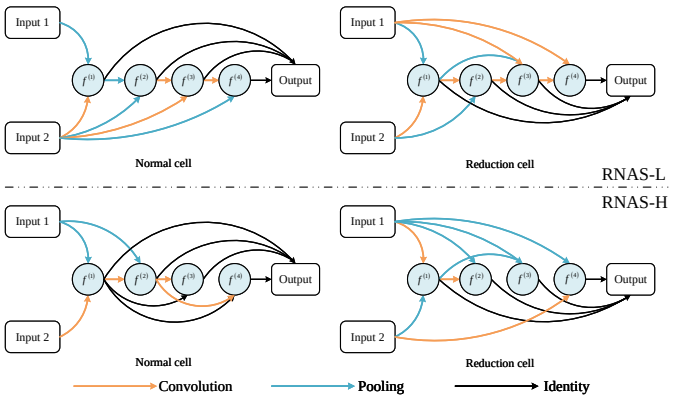


Fig. 4: Architecture of the cells obtained on CIFAR-10.

nodes in RNAS-H and RNAS-L are intensive, which is consistent with the property of the robust architecture in [11]. We use 20 cells to construct the target network, train the network through standard training and adversarial training respectively and evaluate it under various adversarial attacks. The comparison results are summarized in Table. 1.

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 a better robust accuracy. (2) Compared with NAS baselines, our model achieves the best performance under various adversarial attacks. The main reason is that we restrian the feature distortion in our search process. Hence, the obtained network is insensitive to the distortion caused by adversarial examples. (3) Compared with existing NAS-based robust search methods, RNAS-L and RNAS-H both achieve better performance.

In Table. 1, even only with standard training, RNAS is still more robust than other network architectures. This result indicates that the architecture obtained by RNAS has a natural robust property. In addition, the clean accuracy of RNAS is yet closed to that of the optimal NAS.

5.3 Results on CIFAR-100, SVHN and Tiny-ImageNet

To further evaluate the effectiveness of RNAS, we transfer the model to CIFAR-100, SVHN and Tiny-ImageNet, specifically. We use 20 cells obtained on CIFAR-10 to construct a network and retrain it on CIFAR-100, SVHN and Tiny-ImageNet by adversarial training respectively. In Table. 2, Table. 3 and Table. 4 compared with most of the baselines, RNAS obtained on CIFAR-10 can still improve the robustness after adversarial training on CIFAR-100, SVHN and Tiny-ImageNet. The results indicate that the RNAS architecture is highly transferable.

Table 2: Evaluation of different architectures on CIFAR-100.

Model	Clean(%)	FGSM(%)	PGD^{20}(%)	PGD^{100}(%)	C&W(%)	AA(%)
ResNet18[12]	55.12	25.65	21.08	19.98	17.04	18.02
DenseNet121[16]	**61.71**	34.28	27.30	27.07	20.18	24.55
Mobilnetv2[33]	53.81	28.45	23.80	23.76	16.04	20.30
NasNet[43]	59.64	29.83	24.41	24.38	20.41	22.06
DARTS[24]	59.14	30.35	25.66	25.40	20.72	22.65
PC-DARTS[39]	59.44	20.29	24.10	24.02	19.45	22.06
RobNet-free[11]	59.64	34.23	27.21	27.18	21.92	24.82
DSRNA[14]	57.44	35.03	28.11	27.97	21.52	25.20
AdvRush[27]	58.73	39.01	30.16	29.67	20.08	26.46
RNAS-H	59.80	**42.44**	**32.05**	**31.99**	**29.84**	**28.29**
RNAS-L	60.24	40.52	31.11	31.06	20.84	27.37

Table 3: Evaluation of different architectures on SVHN.

Model	Clean(%)	FGSM(%)	PGD^{20}(%)	PGD^{100}(%)	C&W(%)	AA(%)
ResNet18[12]	92.06	85.70	68.50	68.18	58.31	63.37
DenseNet121[16]	93.72	89.68	72.62	72.32	60.61	65.92
Mobilnetv2[33]	89.94	83.28	66.17	65.78	54.63	59.22
NasNet[43]	94.55	87.54	69.52	68.88	42.44	64.80
DARTS[24]	**94.90**	90.01	77.58	77.15	60.22	66.24
PC-DARTS[39]	94.78	88.81	76.07	76.01	56.49	67.56
RobNet-free[11]	92.45	89.33	85.30	84.68	61.28	72.15
DSRNA[14]	91.58	91.27	84.94	84.03	40.50	74.41
AdvRush[27]	94.80	91.16	89.89	88.79	60.05	75.52
RNAS-H	94.58	**93.07**	**91.46**	**89.80**	**63.64**	**77.26**
RNAS-L	93.88	92.08	89.67	88.92	61.44	75.89

In addition, we observe that on all three datasets, the robustness improvement of RNAS-L is not as high as that on CIFAR-10. The reason is that the architecture search process is dataset-dependent, and RNAS-L is obtained under a stricter network vulnerability constraint, which may *overfit* the original dataset. After transferred to other dataset, the performance of the architecture obtained on CIFAR-10 decreases a little. Despite this, RNAS-L still achieves a comparative or even better performance compared with SOTA methods. The differences in performance between RNAS-H and RNAS-L are due to the intensity of the constraints on the vulnerability, which will be discussed in Section 5.4.

To verify the effectiveness of RNAS on other datasets, we also directly search the architecture on Tiny-Imagenet. which evaluation results is presented in the Table. 5. RNAS can still outperform other models.

Table 4: Evaluation of different architectures on Tiny-ImageNet.

Model	Clean(%)	FGSM(%)	PGD^{20}(%)	PGD^{100}(%)	C&W(%)	AA(%)
ResNet18[12]	38.36	18.80	14.92	14.85	**25.61**	12.20
DenseNet121[16]	46.26	22.88	19.11	19.05	22.13	17.78
Mobilnetv2[33]	39.48	17.64	15.33	15.22	16.66	14.80
NasNet[43]	44.52	23.69	20.60	20.48	19.87	19.57
DARTS[24]	**45.94**	24.36	21.74	21.68	20.67	20.09
PC-DARTS[39]	45.42	25.38	22.94	21.88	19.99	20.45
RobNet-free[11]	44.24	25.44	23.85	23.65	20.11	22.54
DSRNA[14]	44.42	**28.52**	24.32	24.09	14.56	21.55
AdvRush[27]	45.45	25.20	23.58	23.38	18.68	22.78
RNAS-H	45.92	28.30	**26.84**	**26.49**	24.27	**24.22**
RNAS-L	44.82	26.28	24.04	23.89	22.83	23.02

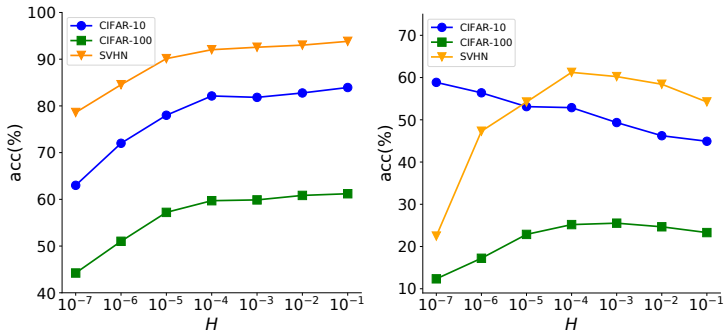
Table 5: Evaluation of the networks obtained on Tiny-ImageNet. Except for ResNet18 and DenseNet121, the rest models are directly obtained by searching on Tiny-ImageNet.

Model	Clean(%)	FGSM(%)	PGD^{20}(%)	PGD^{100}(%)	C&W(%)	AA(%)
ResNet18[12]	38.36	18.80	14.92	14.85	25.61	12.20
DenseNet121[16]	46.26	22.88	19.11	19.05	22.13	17.78
DARTS[24]	46.85	24.85	22.14	21.86	22.07	20.33
PC-DARTS[39]	46.22	26.40	21.63	21.49	23.74	21.32
RobNet-free[11]	45.43	28.61	24.85	24.46	21.21	21.52
DSRNA[14]	46.50	28.67	25.33	25.09	18.66	23.56
AdvRush[27]	45.98	27.31	25.85	25.40	15.70	23.81
RNAS-H	45.92	28.53	26.92	26.79	**26.47**	24.82
RNAS-L	**47.62**	**29.24**	**27.34**	**27.02**	20.85	**25.74**

5.4 Upper bound H of network vulnerability

Recall that H is the upper bound of the constraint on the network vulnerability, which controls the constraint's intensity. The larger the H value, the looser the constraint, vice versa. $H = 0$ means no distortion in the latent features. Intuitively, a smaller H ensures a more robust architecture in the search process. However, experiments show that too small an H may lead to an overfit to the original data, thus reduce the model's generalization ability.

Network architecture search is a time-consuming process, and the adversarial training introduced into RNAS further overload such computation costs. In practice, we search the architecture on a small proxy dataset first, then transfer this architecture to the target dataset. Thus, transferability is a valuable property of the obtained models. In RNAS, the value of H directly influences



(a) Clean accuracy (b) Robust accuracy by PGD^{20}

Fig. 5: The clean accuracy and robust accuracy of RNAS with different H value on different datasets. The models are obtained on CIFAR-10 and adversarially trained on CIFAR-10/100 and SVHN.

the model's transferability. Fig. 5 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 SVHN, the PGD robust accuracy falls rapidly after rising. This indicates that too small an H makes the model overfit CIFAR-10 and reduces its generalization ability; too large an H is not effective on constraining the network vulnerability. In conclusion, the advantage of RNAS is that the transferability can be improved by adjusting the upper bound of network vulnerability H for different datasets.

6 Conclusion

In this paper, we empirically verify that the distortion of non-robust features in the latent feature space plays a vital role in misclassification caused by adversarial examples. Experimental result shows on various datasets, RNAS outperforms other classic and SOTA models in robustness. Under both PGD adversarial training, RNAS has a higher robustness than other methods. Even without adversarial training, RNAS still shows some robustness. In addition, we find that the transferability of RNAS can be improved by adjusting the upper bound of network vulnerability H. In the future, we will focus on extending our work to other tasks, such as semantic segmentation and object detection.

Acknowledgment. This work is sponsored by the Zhejiang Provincial Natural Science Foundation of China (LZ22F020007, LGF20F020007), Major Research Plan of the National Natural Science Foundation of China (92167203), National Key R&D Program of China (2018YFB2100400), Natural Science Foundation of China (61902082, 61972357), and project funded by China Postdoctoral Science Foundation under No.2022M713253.

References

1. Andriushchenko, M., Croce, F., Flammarion, N., Hein, M.: Square attack: a query-efficient black-box adversarial attack via random search. In: European Conference on Computer Vision. pp. 484–501 (2020)
2. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy. pp. 39–57 (2017)
3. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European Conference on Computer Vision. pp. 801–818 (2018)
4. Chen, X., Xie, L., Wu, J., Tian, Q.: 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)
5. Croce, F., Hein, M.: Minimally distorted adversarial examples with a fast adaptive boundary attack. In: International Conference on Machine Learning. pp. 2196–2205 (2020)
6. Croce, F., Hein, M.: Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In: International Conference on Machine Learning. pp. 2206–2216 (2020)
7. Devaguptapu, C., Agarwal, D., Mittal, G., Balasubramanian, V.N.: An empirical study on the robustness of NAS based architectures (2020), *arXiv preprint arXiv:*2007.08428
8. Dong, M., Li, Y., Wang, Y., Xu, C.: Adversarially robust neural architectures (2020), *arXiv preprint arXiv:*2009.00902
9. Girshick, R., Donahue, J., Darrell, T., Malik, J.: 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)
10. Goodfellow, I., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (2015)
11. Guo, M., Yang, Y., Xu, R., Liu, Z., Lin, D.: 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)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
13. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. In: Advances in 28th Neural Information Processing Systems (2015)
14. Hosseini, R., Yang, X., Xie, P.: DSRNA: Differentiable Search of Robust Neural Architectures. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6196–6205 (2021)
15. Huang, G., Liu, Z., Maaten, L., Q, K.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017)
16. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)
17. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: SqueezeNet: AlexNet-level accuracy with 50× fewer parameters and <0.5MB model size. In: International Conference on Learning Representations (2017)

18. Ilyas, A., Santurkar, S., Engstrom, L., Tran, B., Madry, A.: Adversarial examples are not bugs, they are features. In: Advances in neural information processing systems. vol. 32 (2019)
19. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. In: Technical Report (2009)
20. Krizhevsky, A., Ilya, S., Geoffrey E, H.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems. pp. 1097–1105 (2012)
21. Kullback, S., Leibler, R.A.: On information and sufficiency. In: The annals of mathematical statistics. pp. 79–86 (1951)
22. Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., Jana, S.: Certified robustness to adversarial examples with differential privacy. In: 2019 IEEE Symposium on Security and Privacy. pp. 656–672 (2019)
23. Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.J., Li, F.F., Yuille, A., Huang, J., Murphy, K.: Progressive neural architecture search. In: Proceedings of the European Conference on Computer Vision. pp. 19–34 (2018)
24. Liu, H., Simonyan, K., Yang, Y.: DARTS: Differentiable Architecture Search. In: International Conference on Learning Representations (2019)
25. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. In: Proceedings of the European Conference on Computer Vision. pp. 116–131 (2018)
26. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks (2017)
27. Mok, J., Na, B., Choe, H., Yoon, S.: AdvRush: Searching for Adversarially Robust Neural Architectures. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12322–12332 (2021)
28. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: 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)
29. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Y, N.A.: Reading digits in natural images with unsupervised feature learning. In: NIPS Workshop on Deep Learning and Unsupervised Feature Learning (2011)
30. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A.: 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)
31. Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A.: Distillation as a defense to adversarial perturbations against deep neural networks. In: 2016 IEEE Symposium on Security and Privacy. pp. 582–597 (2016)
32. Real, E., Aggarwal, A., Huang, Y., V. Le, Q.: Regularized evolution for image classifier architecture search. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 4780–4789 (2019)
33. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4510–4520 (2018)
34. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large scale image recognition. In: International Conference on Learning Representations (2015)
35. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks (2013), *arXiv preprint arXiv:*1312.6199

36. Tu, C.C., Ting, P., Chen, P.Y., Liu, S., Zhang, H., Yi, J., Hsieh, C.J., Cheng, S.M.: 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)
37. Wong, E., Rice, L., Kolter, J.Z.: Fast is better than free: Revisiting adversarial training. In: International Conference on Learning Representations (2020)
38. Xie, S., Zheng, H., Liu, C., Lin, L.: SNAS: Stochastic neural architecture search. In: International Conference on Learning Representations (2019)
39. Xu, Y., Xie, L., Zhang, X., Chen, X.: PC-DARTS: Partial Channel Connections for Memory-Efficient Architecture Search. In: International Conference on Learning Representations (2019)
40. Zhang, H., Yu, Y., Jiao, J., Xing, E., Ghaoui, L.E., Jordan, M.I.: Theoretically principled trade-off between robustness and accuracy. In: International Conference on Machine Learning. pp. 7472–7482 (2019)
41. Zhang, Y., Xiang, T., Hospedales, T.M., Lu, H.: Deep mutual learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4320–4328 (2018)
42. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. In: International Conference on Learning Representations (2017)
43. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8697–8710 (2018)
44. Zugner, D., Akbarnejad, A., Gunnemann, S.: 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)