

Anti-Bandit Neural Architecture Search for Model Defense

Hanlin Chen¹, Baochang Zhang^{1*}, Song Xue¹, Xuan Gong^{2*}, Hong Liu³,
Rongrong Ji³, and David Doermann²

¹ Beihang University, Beijing, China

² University at Buffalo, Buffalo, USA

³ Xiamen University, Fujian, China,
{hlchen,bczhang}@buaa.edu.cn

Abstract. Deep convolutional neural networks (DCNNs) have dominated as the best performers in machine learning, but can be challenged by adversarial attacks. In this paper, we defend against adversarial attacks using neural architecture search (NAS) which is based on a comprehensive search of denoising blocks, weight-free operations, Gabor filters and convolutions. The resulting anti-bandit NAS (ABanditNAS) incorporates a new operation evaluation measure and search process based on the lower and upper confidence bounds (LCB and UCB). Unlike the conventional bandit algorithm using UCB for evaluation only, we use UCB to abandon arms for search efficiency and LCB for a fair competition between arms. Extensive experiments demonstrate that ABanditNAS is about twice as fast as the state-of-the-art NAS method, while achieving an 8.73% improvement over prior arts on CIFAR-10 under PGD-7.

Keywords: Neural Architecture Search (NAS), Bandit, Adversarial Defense

1 Introduction

The success of deep learning models [4] have been demonstrated on various computer vision tasks such as image classification [18], instance segmentation [25] and object detection [36]. However, existing deep models are sensitive to adversarial attacks [6, 16, 37], where adding an imperceptible perturbation to input images can cause the models to perform incorrectly. Szegedy et. al [37] also observe that these adversarial examples are transferable across multiple models such that adversarial examples generated for one model might mislead other models as well. Therefore, models deployed in the real world scenarios are susceptible to adversarial attacks [24]. While many methods have been proposed to defend against these attacks [37, 7], improving the network training process proves to be one of the most popular. These methods inject adversarial examples into the training data to retrain the network [16, 21, 1]. Similarly, pre-processing

* Corresponding author.

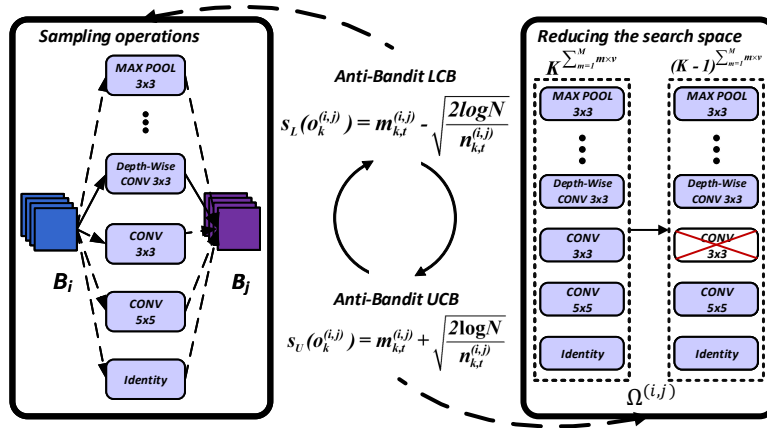


Fig. 1. ABanditNAS is mainly divided into two steps: sampling using LCB and abandoning based on UCB.

defense methods modify adversarial inputs to resemble clean inputs [35, 22] by transforming the adversarial images into clean images before they are fed into the classifier.

Overall, however, finding adversarially robust architectures using neural architecture search (NAS) shows even more promising results [7, 11, 27, 29]. NAS has attracted a great attention with remarkable performance in various deep learning tasks. In [7] the researchers investigate the dependence of adversarial robustness on the network architecture via NAS. A neural architecture search framework for adversarial medical image segmentation is proposed by [29]. [27] leverages one-shot NAS [3] to understand the influence of network architectures against adversarial attacks. Although promising performance is achieved in existing NAS based methods, this direction still remains largely unexplored.

In this paper, we consider NAS for model defense by treating it as a multi-armed bandit problem and introduce a new anti-bandit algorithm into adversarially robust network architecture search. To improve the robustness to adversarial attacks, a comprehensive search space is designed by including diverse operations, such as denoising blocks, weight-free operations, Gabor filters and convolutions. However, searching a robust network architecture is more challenging than traditional NAS, due to the complicated search space, and learning inefficiency caused by adversarial training. We develop an anti-bandit algorithm based on both the upper confidence bound (UCB) and the lower confidence bound (LCB) to handle the huge and complicated search space, where the number of operations that define the space can be 9^{60} ! Our anti-bandit algorithm uses UCB to reduce the search space, and LCB to guarantee that every arm is fairly tested before being abandoned.

Making use of the LCB, operations which have poor performance early, such as parameterized operations, will be given more chances but they are thrown

away once they are confirmed to be bad. Meanwhile, weight-free operations will be compared with parameterized operations only when they are well trained. Based on the observation that the early optimal operation is not necessarily the optimal one in the end, and the worst operations in the early stage usually has a worse performance at the end [45], we exploit UCB to prune the worst operations earlier, after a fair performance evaluation via LCB. This means that the operations we finally reserve are certainly a near optimal solution. On the other hand, with the operation pruning process, the search space becomes smaller and smaller, leading to an efficient search process. Our framework shown in Fig. 1 highlights the anti-bandit NAS (ABanditNAS) for finding a robust architecture from a very complicated search space. The contributions of our paper are as follows:

- ABanditNAS is developed to solve the adversarially robust optimization and architecture search in a unified framework. We introduce an anti-bandit algorithm based on a specific operation search strategy with a lower and an upper bound, which can learn a robust architecture based on a comprehensive operation space.
- The search space is greatly reduced by our anti-bandit pruning method which abandons operations with less potential, and significantly reduces the search complexity from exponential to polynomial, i.e., $\mathcal{O}(K^{|\mathcal{E}_{\mathcal{M}}| \times v})$ to $\mathcal{O}(K^2 \times T)$ (see Section 3.4 for details).
- Extensive experiments demonstrate that the proposed algorithm achieves better performance than other adversarially robust models on commonly used MNIST and CIFAR-10.

2 Related Work

Neural architecture search (NAS). NAS becomes one of the most promising technologies in the deep learning paradigm. Reinforcement learning (RL) based methods [47, 46] train and evaluate more than 20,000 neural networks across 500 GPUs over 4 days. The recent differentiable architecture search (DARTS) reduces the search time by formulating the task in a differentiable manner [23]. However, DARTS and its variants [23, 41] might be less efficient for a complicated search space. To speed up the search process, a *one-shot* strategy is introduced to do NAS within a few GPU days [23, 31]. In this one-shot architecture search, each architecture in the search space is considered as a sub-graph sampled from a super-graph, and the search process can be accelerated by parameter sharing [31]. Though [7] uses NAS with reinforcement learning to find adversarially robust architectures that achieve good results, it is insignificant compared to the search time. Those methods also seldom consider high diversity in operations closely related to model defense in the search strategy.

Adversarial attacks. Recent research has shown that neural networks exhibit significant vulnerability to adversarial examples. After the discovery of adversarial examples by [37], [16] proposes the Fast Gradient Sign Method (FGSM) to generate adversarial examples with a single gradient step. Later, in [21], the

researchers propose the Basic Iterative Method (BIM), which takes multiple and smaller FGSM steps to improve FGSM, but renders the adversarial training very slow. This iterative adversarial attack is further strengthened by adding multiple random restarts, and is also incorporated into the adversarial training procedure. In addition, projected gradient descent (PGD) [26] adversary attack, a variant of BIM with a uniform random noise as initialization, is recognized to be one of the most powerful first-order attacks [1]. Other popular attacks include the Carlini and Wagner Attack [6] and Momentum Iterative Attack [10]. Among them, [6] devises state-of-the-art attacks under various pixel-space l_p norm-ball constraints by proposing multiple adversarial loss functions.

Model defense. In order to resist attacks, various methods have been proposed. A category of defense methods improve network’s training regime to counter adversarial attacks. The most common method is adversarial training [21, 28] with adversarial examples added to the training data. In [26], a defense method called Min-Max optimization is introduced to augment the training data with first-order attack samples. [38] investigates fast training of adversarially robust models to perturb both the images and the labels during training. There are also some model defense methods that target at removing adversarial perturbation by transforming the input images before feeding them to the network [22, 1, 17]. In [12, 8], the effect of JPEG compression is investigated for removing adversarial noise. In [30], the authors apply a set of filters such as median filters and averaging filters to remove perturbation. In [42], a ME-Net method is introduced to destroy the adversarial noise and re-enforce the global structure of the original images. With the development of NAS, finding adversarially robust architectures using NAS is another promising direction [7], which is worth in-depth exploration. Recently, [29] designs three types of primitive operation set in the search space to automatically find two-cell architectures for semantic image segmentation, especially medical image segmentation, leading to a NAS-Unet backbone network.

In this paper, an anti-bandit algorithm is introduced into NAS, and we develop a new optimization framework to generate adversarially robust networks. Unlike [19] using bandits to produce black-box adversarial samples, we propose an anti-bandit algorithm to obtain a robust network architecture. In addition, existing NAS-based model defense methods either target at different applications from ours or are less efficient for object classification [7, 11, 27, 29].

3 Anti-Bandit Neural Architecture Search

3.1 Search Space

Following [47, 23, 45], we search for computation cells as the building blocks of the final architecture. Different from these approaches, we search for v ($v > 2$) kinds of cells instead of only normal and reduction cells. Although it increases the search space, our search space reduction in ABanditNAS can make the search efficient enough. A cell is a fully-connected directed acyclic graph (DAG) of M

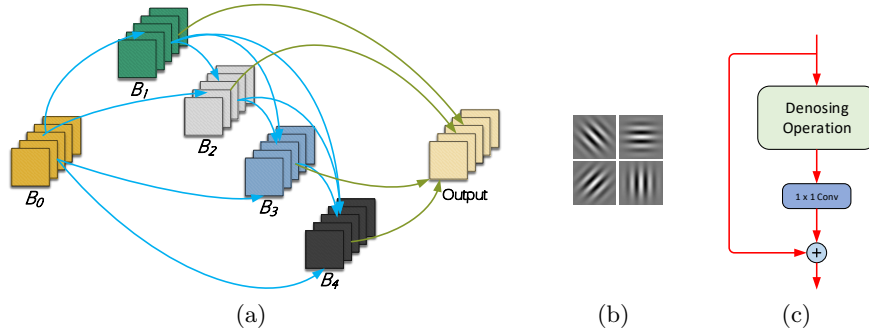


Fig. 2. (a) A cell containing four intermediate nodes B_1, B_2, B_3, B_4 that apply sampled operations on the input node B_0 . B_0 is from the output of the previous cell. The output node concatenates the outputs of the four intermediate nodes. (b) Gabor Filter. (c) A generic denoising block. Following [40], it wraps the denoising operation with a 1×1 convolution and an identity skip connection [18].

nodes, *i.e.*, $\{B_1, B_2, \dots, B_M\}$ as shown in Fig. 2(a). Each node B_i takes its dependent nodes as input, and generates an output through the selected operation $B_j = o^{(i,j)}(B_i)$. Here each node is a specific tensor (*e.g.*, a feature map in convolutional neural networks) and each directed edge (i, j) between B_i and B_j denotes an operation $o^{(i,j)}(\cdot)$, which is sampled from $\Omega^{(i,j)} = \{o_1^{(i,j)}, \dots, o_K^{(i,j)}\}$. Note that the constraint $i < j$ ensures that there are no cycles in a cell. Each cell takes the output of the previous cell as input, and we define this node belonging to the previous cell as the input node B_0 of the current cell for easy description. The set of the operations Ω consists of $K = 9$ operations. Following [23], there are seven normal operations that are the 3×3 max pooling, 3×3 average pooling, skip connection (identity), 3×3 convolution with rate 2, 5×5 convolution with rate 2, 3×3 depth-wise separable convolution, and 5×5 depth-wise separable convolution. The other two are 3×3 Gabor filter and denoising block. Therefore, the size of the whole search space is $K^{|\mathcal{E}_{\mathcal{M}}| \times v}$, where $\mathcal{E}_{\mathcal{M}}$ is the set of possible edges with M intermediate nodes in the fully-connected DAG. The search space of a cell is constructed by the operations of all the edges, denoted as $\{\Omega^{(i,j)}\}$. In our case with $M = 4$ and $v = 6$, together with the input node, the total number of cell structures in the search space is $9^{(1+2+3+4) \times 6} = 9^{10 \times 6}$.

Gabor filter. Gabor wavelets [15, 14] were invented by Dennis Gabor using complex functions to serve as a basis for Fourier transform in information theory applications. The Gabor wavelets (kernels or filters) in Fig. 2(b) are defined as: $\exp(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}) \cos(2\pi \frac{x'}{\lambda} + \psi)$, where $x' = x \cos \theta + y \sin \theta$ and $y' = -x \sin \theta + y \cos \theta$. We set $\sigma, \gamma, \lambda, \psi$ and θ to be learnable parameters. Note that the symbols used here apply only to the Gabor filter and are different from the symbols used in the rest of this paper. An important property of the wavelets is that the product of its standard deviations is minimized in both time and frequency domains. Also, robustness is another important property which we use here [32].

Denoising block. In [40], the researchers suggest that adversarial perturbations on images can result in noise in the features. Thus, a denoising block Fig. 2(c) is used to improve adversarial robustness via feature denoising. Similarly, we add the non-local mean denoising block [5] to the search space to denoise the features. It computes a denoised feature map z of an input feature map x by taking a weighted mean of the features over all spatial locations \mathcal{L} as $z_p = \frac{1}{C(x)} \sum_{\forall q \in \mathcal{L}} f(x_p, x_q) \cdot x_q$, where $f(x_p, x_q)$ is a feature-dependent weighting function and $C(x)$ is a normalization function. Also, the denoising block needs huge computations because of the matrix multiplication between features.

It is known that adversarial training is more challenging than that of natural training [33], which adds an additional burden to NAS. For example, adversarial training using the F -step PGD attack needs roughly $F + 1$ times more computation. Also, more operations added to the search space are another burden. To solve these problems, we introduce operation space reduction based on the UCB bandit algorithm into NAS, to significantly reduce the cost of GPU hours, leading to our efficient ABanditNAS.

3.2 Adversarial Optimization for ABanditNAS

Adversarial training [26] is a method for learning networks so that they are robust to adversarial attacks. Given a network f_θ parameterized by θ , a dataset (x_e, y_e) , a loss function l and a threat model Δ , the learning problem is typically cast as the following optimization problem: $\min_\theta \sum_e \max_{\delta \in \Delta} l(f_\theta(x_e + \delta), y_e)$, where δ is the adversarial perturbation. A typical choice for a threat model is to take $\Delta = \{\delta : \|\delta\|_\infty \leq \epsilon\}$ for some $\epsilon > 0$, where $\|\cdot\|_\infty$ is some l_∞ -norm distance metric and ϵ is the adversarial manipulation budget. This is the l_∞ threat model used by [26] and what we consider in this paper. The procedure for adversarial training is to use attacks to approximate the inner maximization over Δ , followed by some variation of gradient descent on the model parameters θ . For example, one of the earliest versions of adversarial training uses the Fast Gradient Sign Method (FGSM) [16] to approximate the inner maximization. This could be seen as a relatively inaccurate approximation of the inner maximization for l_∞ perturbations, and has the closed form solution: $\theta = \epsilon \cdot \text{sign}(\nabla_x l(f(x), y))$.

A better approximation of the inner maximization is to take multiple, smaller FGSM steps of size α instead. However, the number of gradient computations caused by the multiple steps is proportional to $\mathcal{O}(EF)$ in a single epoch, where E is the size of the dataset and F is the number of steps taken by the PGD adversary. This is F times greater than the standard training which has $\mathcal{O}(E)$ gradient computations per epoch, and so the adversarial training is typically F times slower. To speed up the adversarial training, we combine the FGSM with random initialization [39].

3.3 Anti-Bandit

In machine learning, the multi-armed bandit problem [2, 34] is a classic reinforcement learning (RL) problem that exemplifies the exploration-exploitation

trade-off dilemma: shall we stick to an arm that gave high reward so far (exploitation) or rather probe other arms further (exploration)? The Upper Confidence Bound (UCB) is widely used for dealing with the exploration-exploitation dilemma in the multi-armed bandit problem. For example, the idea of bandit is exploited to improve many classical RL methods such as Monte Carlo [20] and Q-learning [13]. The most famous one is AlphaGo [34], which uses the Monte Carlo Tree Search (MCTS) algorithm to play the board game Go but based on a very powerful computing platform unavailable to common researchers. Briefly, the UCB algorithm chooses at trial the arm k that maximizes

$$\hat{r}_k + \sqrt{\frac{2 \log N}{n_k}}, \quad (1)$$

where \hat{r}_k is the average reward obtained from arm k , and n_k is the number of times arm k has been played up to trial N . The first term in Eq. 1 is the value term which favors actions that look good historically, and the second is the exploration term which makes actions get an exploration bonus that grows with $\log N$. The total value can be interpreted as the upper bound of a confidence interval, so that the true mean reward of each arm k with a high probability is below this UCB.

The UCB in bandit is not applicable in NAS, because it is too time-consuming to choose an arm from a huge search space (a huge number of arms, e.g., 9^{60}), particularly when limited computational resources are available. To solve the problem, we introduce an anti-bandit algorithm to reduce the arms for the huge-armed problem by incorporating both the upper confidence bound (UCB) and the lower confidence bound (LCB) into the conventional bandit algorithm. We first define LCB as

$$\hat{r}_k - \sqrt{\frac{2 \log N}{n_k}}. \quad (2)$$

LCB is designed to sample an arm from a huge number of arms for one more trial (later in Eq. 3). A smaller LCB means that the less played arm (a smaller n_k) is given a bigger chance to be sampled for a trial. Unlike the conventional bandit based on the maximum UCB (Eq. 1) to choose an arm, our UCB (Eq. 6) is used to abandon the arm operation with the minimum value, which is why we call our algorithm anti-bandit.

Our anti-bandit algorithm is specifically designed for the huge-armed bandit problem by reducing the number of arms based on the UCB. Together with the LCB, it can guarantee every arm is fairly tested before being abandoned.

3.4 Anti-Bandit Strategy for ABanditNAS

As described in [43, 45], the validation accuracy ranking of different network architectures is not a reliable indicator of the final architecture quality. However, the experimental results actually suggest a nice property that if an architecture performs poorly in the beginning of training, there is little hope that it can be part of the final optimal model [45]. As the training progresses, this observation

Algorithm 1: ABanditNAS

Input: Training data, validation data, searching hyper-graph, adversarial perturbation δ , adversarial manipulation budget ϵ , $K = 9$, hyper-parameters α , $\lambda = 0.7$, $T = 3$.

Output: The remaining optimal structure;

- 1 $t = 0$; $c = 0$;
- 2 Get initial performance $m_{k,0}^{(i,j)}$;
- 3 **while** ($K > 1$) **do**
- 4 $c \leftarrow c + 1$;
- 5 $t \leftarrow t + 1$;
- 6 Calculate $s_L(o_k^{(i,j)})$ using Eq. 3;
- 7 Calculate $p(o_k^{(i,j)})$ using Eq. 4;
- 8 Select an architecture by sampling one operation based on $p(o_k^{(i,j)})$ from $\Omega^{(i,j)}$ for every edge;
- 9 // Train adversarially the selected architecture
- 10 **for** $e = 1, \dots, E$ **do**
- 11 $\delta = \text{Uniform}(-\epsilon, \epsilon)$;
- 12 $\delta \leftarrow \delta + \alpha \cdot \text{sign}(\nabla_x l(f(x_e + \delta), y_e))$;
- 13 $\delta = \max(\min(\delta, \epsilon), -\epsilon)$;
- 14 $\theta \leftarrow \theta - \nabla_{\theta} l(f_{\theta}(x_e + \delta), y_e)$;
- 15 **end**
- 16 Get the accuracy a on the validation data;
- 17 Update the performance $m_{k,t}^{(i,j)}$ using Eq. 5;
- 18 **if** $c = K * T$ **then**
- 19 Calculate $s_U(o_k^{(i,j)})$ using Eq. 6;
- 20 Update the search space $\{\Omega^{(i,j)}\}$ using Eq. 7;
- 21 $c = 0$;
- 22 $K \leftarrow K - 1$;
- 23 **end**
- 24 **end**

is more and more certain. Based on this observation, we derive a simple yet effective operation abandoning method. During training, along with the increasing epochs, we progressively abandon the worst performing operation for each edge. Unlike [45] which just uses the performance as the evaluation metric to decide which operation should be pruned, we use the anti-bandit algorithm described next to make a decision about which one should be pruned.

Following UCB in the bandit algorithm, we obtain the initial performance for each operation in every edge. Specifically, we sample one from the K operations in $\Omega^{(i,j)}$ for every edge, then obtain the validation accuracy a which is the initial performance $m_{k,0}^{(i,j)}$ by training adversarially the sampled network for one epoch, and finally assigning this accuracy to all the sampled operations.

By considering the confidence of the k th operation with the UCB for every edge, the LCB is calculated by

$$s_L(o_k^{(i,j)}) = m_{k,t}^{(i,j)} - \sqrt{\frac{2 \log N}{n_{k,t}^{(i,j)}}}, \quad (3)$$

where N is to the total number of samples, $n_{k,t}^{(i,j)}$ refers to the number of times the k th operation of edge (i, j) has been selected, and t is the index of the epoch. The first item in Eq. 3 is the value term which favors the operations that look good historically and the second is the exploration term which allows operations to get an exploration bonus that grows with $\log N$. The selection probability for each operation is defined as

$$p(o_k^{(i,j)}) = \frac{\exp\{-s_L(o_k^{(i,j)})\}}{\sum_m \exp\{-s_L(o_m^{(i,j)})\}}. \quad (4)$$

The minus sign in Eq. 4 means that we prefer to sample operations with a smaller confidence. After sampling one operation for every edge based on $p(o_k^{(i,j)})$, we obtain the validation accuracy a by training adversarially the sampled network for one epoch, and then update the performance $m_{k,t}^{(i,j)}$ which historically indicates the validation accuracy of all the sampled operations $o_k^{(i,j)}$ as

$$m_{k,t}^{(i,j)} = (1 - \lambda)m_{k,t-1}^{(i,j)} + \lambda * a, \quad (5)$$

where λ is a hyper-parameter.

Finally, after $K * T$ samples where T is a hyper-parameter, we calculate the confidence with the UCB according to Eq. 1 as

$$s_U(o_k^{(i,j)}) = m_{k,t}^{(i,j)} + \sqrt{\frac{2 \log N}{n_{k,t}^{(i,j)}}}. \quad (6)$$

The operation with the minimal UCB for every edge is abandoned. This means that the operations that are given more opportunities, but result in poor performance, are removed. With this pruning strategy, the search space is significantly reduced from $|\Omega^{(i,j)}|^{10 \times 6}$ to $(|\Omega^{(i,j)}| - 1)^{10 \times 6}$, and the reduced space becomes

$$\Omega^{(i,j)} \leftarrow \Omega^{(i,j)} - \{\arg \min_{o_k^{(i,j)}} s_U(o_k^{(i,j)})\}, \forall (i, j). \quad (7)$$

The reduction procedure is carried out repeatedly until the optimal structure is obtained where there is only one operation left in each edge. Our anti-bandit search algorithm is summarized in Algorithm 1.

Complexity Analysis. There are $\mathcal{O}(K^{|\mathcal{E}_{\mathcal{M}}| \times v})$ combinations in the process of finding the optimal architecture in the search space with v kinds of different cells. In contrast, ABanditNAS reduces the search space for every $K * T$ epochs. Therefore, the complexity of the proposed method is

$$\mathcal{O}(T \times \sum_{k=2}^K k) = \mathcal{O}(TK^2). \quad (8)$$

4 Experiments

We demonstrate the robustness of our ABanditNAS on two benchmark datasets (MNIST and CIFAR-10) for the image classification task, and compare ABanditNAS with state-of-the-art robust models.

4.1 Experiment Protocol

In our experiments, we search architectures on an over-parameterized network on MNIST and CIFAR-10, and then evaluate the best architecture on corresponding datasets. Unlike previous NAS works [23, 41, 31], we learn six kinds of cells, instead of two, to increase the diversity of the network.

Search and Training Settings. In the search process, the over-parameterized network is constructed with six cells, where the 2^{nd} and 4^{th} cells are used to double the channels of the feature maps and halve the height and width of the feature maps, respectively. There are $M = 4$ intermediate nodes in each cell. The hyperparameter T which denotes the sampling times is set to 3, so the total number of epochs is $\sum_{k=2}^K k * T$. The hyperparameter λ is set to 0.7. The evaluation of the hyperparameters is provided in the supplementary file. A large batch size of 512 is used. And we use an additional regularization cutout [9] for CIFAR-10. The initial number of channels is 16. We employ FGSM adversarial training combined with random initialization and $\epsilon = 0.3$ for MNIST, and $\epsilon = 0.031$ for CIFAR-10. We use SGD with momentum to optimize the network weights, with an initial learning rate of 0.025 for MNIST and 0.1 for CIFAR-10 (annealed down to zero following a cosine schedule), a momentum of 0.9 and a weight decay of 3×10^{-4} for MNIST/CIFAR-10.

After search, the six cells are stacked to get the final networks. To adversarially train them, we employ FGSM combined with random initialization and $\epsilon = 0.3$ on MNIST, and use PGD-7 with $\epsilon = 0.031$ and step size of 0.0078 on CIFAR-10. Next, we use ABanditNAS- V to represent ABanditNAS with V cells in the training process. The number V can be different from the number v . The initial number of channels is 16 for MNIST, and 48 for CIFAR-10. We use a batch size of 96 and an additional regularization cutout [9] for CIFAR-10. We employ the SGD optimizer with an initial learning rate of 0.025 for MNIST and 0.1 for CIFAR-10 (annealed down to zero following a cosine schedule without restart), a momentum of 0.9, a weight decay of 3×10^{-4} , and a gradient clipping at 5. We train 150 epochs for MNIST and CIFAR-10.

White-Box vs. Black-Box Attack Settings. In an adversarial setting, there are two main threat models: white-box attacks where the adversary possesses complete knowledge of the target model, including its parameters, architecture and the training method, and black-box attacks where the adversary feeds perturbed images at test time, which are generated without any knowledge of the target model, and observes the output. We evaluate the robustness of our proposed defense against both settings. The perturbation size ϵ and step size are the same as those in the adversarial training for both the white-box and black-box attacks. The numbers of iterations for MI-FGSM and BIM are both

Architecture	Clean (%)	FGSM (%)	PGD-40 (%)	PGD-100 (%)	# Params (M)	Search Cost (GPU days)	Search Method
LeNet [26]	98.8	95.6	93.2	91.8	3.27	-	Manual
LeNet (Prep. + Adv. train [42])	97.4	-	94.0	91.8	0.06147	-	Manual
UCBNAS	99.5	98.67	96.94	95.4	0.082	0.13	Bandit
UCBNAS (pruning)	99.52	98.56	96.62	94.96	0.066	0.08	Bandit
ABanditNAS-6	99.52	98.94	97.01	95.7	0.089	0.08	Anti-Bandit

Table 1. Robustness of ABanditNAS under FGSM and PGD attacks on MNIST.

Structure	White-Box				Black-Box		
	Clean	MI-FGSM	BIM	PGD	MI-FGSM	BIM	PGD
MNIST ($\epsilon = 0.3$)							
LeNet [26] (copy)	98.8	-	-	93.2	-	-	96.0
ABanditNAS-6 (copy)	99.52	97.41	97.63	97.58	99.09	99.12	99.02
CIFAR-10 ($\epsilon = 0.031$)							
Wide-ResNet [26] (copy)	87.3	-	-	50.0	-	-	64.2
NASNet [7] (copy)	93.2	-	-	50.1	-	-	75.0
ABanditNAS-6 (copy)	87.16	48.77	47.59	50.0	74.94	75.78	76.13
ABanditNAS-6 (ResNet-18)	87.16	48.77	47.59	50.0	77.06	77.63	78.0
ABanditNAS-10 (ResNet-18)	90.64	54.19	55.31	58.74	80.25	80.8	81.26

Table 2. Robustness of our model in the white-box and black-box settings on MNIST and CIFAR-10. Here ϵ is the perturbation size. PGD means PGD-40 for MNIST and PGD-7 for CIFAR-10. ‘copy’ means we use a copied network to generate black-box adversarial examples, and ‘ResNet-18’ means using ResNet-18 to generate black-box adversarial examples.

set to 10 with a step size and a standard perturbation size the same as those in the white-box attacks. We evaluate ABanditNAS against transfer-based attack where a copy of the victim network is trained with the same training setting. We apply attacks similar to the white-box attacks on the copied network to generate black-box adversarial examples. We also generate adversarial samples using a ResNet-18 model, and feed them to the model obtained by ABanditNAS.

4.2 Results on Different Datasets

MNIST. Owing to the search space reduction by anti-bandit, the entire search process only requires 1.93 hours on a single NVIDIA Titan V GPU. For MNIST, the structure searched by ABanditNAS is directly used for training. We evaluate the trained network by 40 and 100 attack steps, and compare our method with LeNet [26] and MeNet [42] in Table 1. From these results, we can see that ABanditNAS using FGSM adversarial training with random initialization is more robust than LeNet with PGD-40 adversarial training, no matter which attack is used. Although MeNet uses matrix estimation (ME) as preprocessing to destroy the adversarial structure of the noise, our method still performs better. In addition, our method has the best performance (99.52%) on the clean images with a strong robustness. For the black-box attacks, Table 2 shows that

Architecture	Clean (%)	MI-FGSM (%)	PGD-7 (%)	PGD-20 (%)	# Params (M)	Search Cost (GPU days)	Search Method
VGG-16 [44]	85.16	-	46.04 (PGD-10)	-	-	-	Manual
ResNet [26]	79.4	-	47.1	43.7	0.46	-	Manual
Wide-ResNet [26]	87.3	-	50.0	45.8	45.9	-	Manual
NASNet [7]	93.2	-	50.1	-	-	$\sim 7 \times 2000$	RL
UCBNAS (pruning)	89.54	53.12	54.55	45.33	8.514	0.08	Bandit
ABanditNAS-6	87.16	48.77	50.0	45.9	2.892	0.08	Anti-Bandit
ABanditNAS-6 (larger)	87.31	52.01	51.24	45.79	12.467	0.08	Anti-Bandit
ABanditNAS-10	90.64	54.19	58.74	50.51	5.188	0.08	Anti-Bandit

Table 3. Validation accuracy and robustness of various models trained on CIFAR-10. Note that the search cost of NASNet which is unknown is estimated based on [7]. ‘PGD-10’ means the result of VGG-16 is under PGD-10 attack which comes from [44].

they barely affect the structures searched by ABanditNAS compared with other models, either manually designed or searched by NAS. As illustrated in Fig. 3(a), with the increase of the perturbation size, our network’s performance does not drop significantly, showing the robustness of our method.

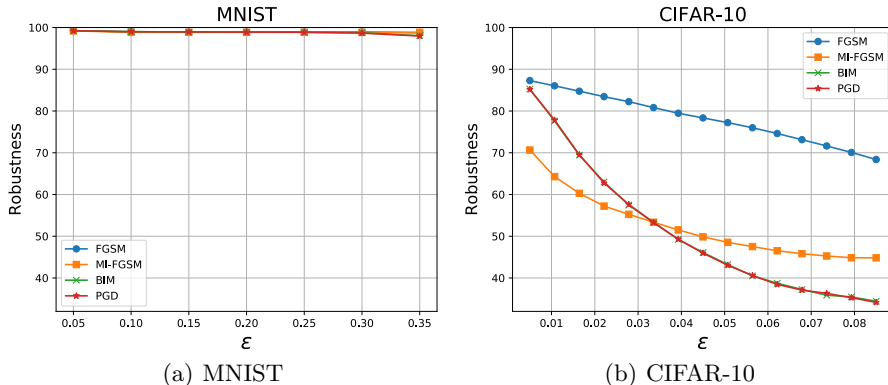


Fig. 3. Robustness of ABanditNAS against different white-box attacks for various perturbation budgets.

We also apply the conventional bandit which samples operations based on UCB to search the network, leading to UCBNAS. The main differences between UCBNAS and ABanditNAS lie in that UCBNAS only uses UCB as an evaluation measure to select an operation, and there is no operation pruning involved. Compared with UCBNAS, ABanditNAS can get better performance and use less search time under adversarial attacks as shown in Table 1. Also, to further demonstrate the effectiveness of our ABanditNAS, we use UCBNAS with pruning to search for a robust model, which not only uses UCB to select an operation,

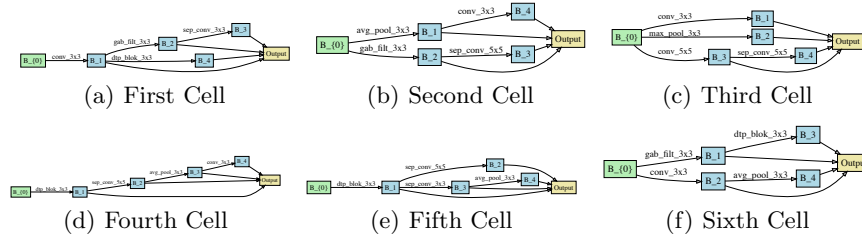


Fig. 4. Detailed structures of the best cells discovered on CIFAR-10 using FGSM with random initialization.

but also prune operation of less potential. Although UCBNAS (pruning) is as fast as ABanditNAS, it has worse performance than ABanditNAS because of unfair competitions between operations before pruning.

CIFAR-10. The results for different architectures on CIFAR-10 are summarized in Table 3. We use one Titan V GPU to search, and the batch size is 512. The entire search process takes about 1.94 hours. We consider $V = 6$ and $V = 10$ cells for training. In addition, we also train a larger network variant with 100 initial channels for $V = 6$. Compared with Wide-ResNet, ABanditNAS-10 achieves not only a better performance (50.0% vs. 58.74%) in PGD-7, but also fewer parameters (45.9M vs. 5.188M). Although the result of VGG-16 is under PGD-10, ABanditNAS-10 achieves a better performance under more serious attack PGD-20 (46.04% vs. 50.51%). When compared with NASNet⁴ which has a better performance on clean images, our method obtains better performance on adversarial examples with a much faster search speed ($\sim 7 \times 2000$ vs. 0.08). Note that the results in Table 3 are the best we got, which are unstable and need more trials to get the results. Table 2 shows the black-box attacks barely affect the networks obtained by ABanditNAS, much less than those by other methods. In addition, Fig. 3(b) illustrates ABanditNAS is still robust when the disturbance increases.

For the structure searched by ABanditNAS on CIFAR-10, we find that the robust structure prefers pooling operations, Gabor filters and denoising blocks (Fig. 4). The reasons lie in that the pooling can enhance the nonlinear modeling capacity, Gabor filters can extract robust features, and the denoising block and mean pooling act as smoothing filters for denoising. Gabor filters and denoising blocks are usually set in the front of cell by ABanditNAS to denoise feature encoded by the previous cell. The setting is consistent with [40], which demonstrates the rationality of ABanditNAS.

⁴ Results are from [7].

4.3 Ablation Study

The performances of the structures searched by ABanditNAS with different values of the λ are used to find the best λ . We train the structures under the same setting.

Effect on the hyperparameter λ : The hyperparameter λ is used to balance the performance between the past and the current. Different values of λ result in similar search costs. From Fig. 4.3, we can see that when $\lambda = 0.7$, ABanditNAS is most robust.

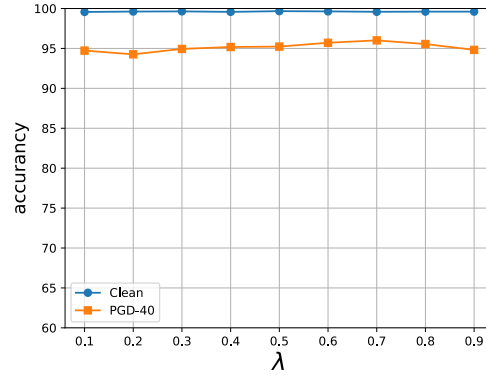


Fig. 5. The performances of the structures searched by ABanditNAS with different values of the hyperparameters T and λ .

5 Conclusion

We have proposed an ABanditNAS approach to design robust structures to defend adversarial attacks. To solve the challenging search problem caused by the complicated huge search space and the adversarial training process, we have introduced an anti-bandit algorithm to improve the search efficiency. We have investigated the relationship between our strategy and potential operations based on both lower and upper bounds. Extensive experiments have demonstrated that the proposed ABanditNAS is much faster than other state-of-the-art NAS methods with a better performance in accuracy. Under adversarial attacks, ABanditNAS achieves much better performance than other methods.

Acknowledgement

Baochang Zhang is also with Shenzhen Academy of Aerospace Technology, Shenzhen, China, and he is the corresponding author. He is in part Supported by National Natural Science Foundation of China under Grant 61672079, Shenzhen Science and Technology Program (No.KQTD2016112515134654)

References

1. Athalye, A., Carlini, N., Wagner, D.: Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In: ICML (2018)
2. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. *Machine learning* (2002)
3. Bender, G., Kindermans, P.J., Zoph, B., Vasudevan, V., Le, Q.V.: Understanding and simplifying one-shot architecture search. In: ICML (2018)
4. Bengio, Y., Goodfellow, I., Courville, A.: Deep learning. Citeseer (2017)
5. Buades, A., Coll, B., Morel, J.: A non-local algorithm for image denoising. In: CVPR (2005)
6. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (2017)
7. Cubuk, E.D., Zoph, B., Schoenholz, S.S., Le, Q.V.: Intriguing properties of adversarial examples. In: ICLR (2017)
8. Das, N., Shanbhogue, M., Chen, S., Hohman, F., Chen, L., Kounavis, M.E., Chau, D.H.: Keeping the bad guys out: Protecting and vaccinating deep learning with jpeg compression. arXiv (2017)
9. DeVries, T., Taylor, G.W.: Improved regularization of convolutional neural networks with cutout. arXiv (2017)
10. Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., Li, J.: Boosting adversarial attacks with momentum. In: CVPR (2018)
11. D.V.Vargas, S.Kotyan: Evolving robust neural architectures to defend from adversarial attacks. arXiv (2019)
12. Dziugaite, G.K., Ghahramani, Z., Roy, D.M.: A study of the effect of jpeg compression on adversarial images. arXiv (2016)
13. Even-Dar, E., Mannor, S., Mansour, Y.: Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of Machine Learning Research* (2006)
14. Gabor, D.: Electrical engineers part iii: Radio and communication engineering, j. *Journal of the Institution of Electrical Engineers - Part III: Radio and Communication Engineering 1945-1948* (1946)
15. Gabor, D.: Theory of communication. part 1: The analysis of information. *Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering* (1946)
16. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv (2014)
17. Gupta, P., Rahtu, E.: Ciidefence: Defeating adversarial attacks by fusing class-specific image inpainting and image denoising. In: ICCV (2019)
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
19. Ilyas, A., Engstrom, L., Madry, A.: Prior convictions: Black-box adversarial attacks with bandits and priors. In: ICLR (2018)
20. Kocsis, L., Szepesvari, C.: Bandit based monte-carlo planning. *Proceedings of the 17th European conference on Machine Learning* (2006)
21. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world. In: ICLR (2016)
22. Liao, F., Liang, M., Dong, Y., Pang, T., Hu, X., Zhu, J.: Defense against adversarial attacks using high-level representation guided denoiser. In: CVPR (2018)

23. Liu, H., Simonyan, K., Yang, Y.: Darts: Differentiable architecture search. In: ICLR (2018)
24. Liu, Y., Chen, X., Liu, C., Song, D.: Delving into transferable adversarial examples and black-box attacks. In: ICLR (2016)
25. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR (2015)
26. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. In: ICLR (2017)
27. M.Guo, Y.Yang, R.Xu, Z.Liu: When nas meets robustness: In search of robust architectures against adversarial attacks. CVPR (2020)
28. Na, T., Ko, J.H., Mukhopadhyay, S.: Cascade adversarial machine learning regularized with a unified embedding. In: ICLR (2017)
29. N.Dong, M.Xu, X.Liang, Y.Jiang, W.Dai, E.Xing: Neural architecture search for adversarial medical image segmentation. In: MICCAI (2019)
30. Osadchy, M., Hernandez-Castro, J., Gibson, S., Dunkelman, O., Pérez-Cabo, D.: No bot expects the deepcaptcha! introducing immutable adversarial examples, with applications to captcha generation. IEEE Transactions on Information Forensics and Security (2017)
31. Pham, H., Guan, M.Y., Zoph, B., Le, Q.V., Dean, J.: Efficient neural architecture search via parameter sharing. In: ICML (2018)
32. Pérez, J.C., Alfarra, M., Jeanneret, G., Bibi, A., Thabet, A.K., Ghanem, B., Arbeláez, P.: Robust gabor networks. arXiv (2019)
33. Shafahi, A., Najib, M., A, A.G., Xu, Z., Dickerson, J., Studer, C., Davis, L.S., Taylor, G., Goldstein, T.: Adversarial training for free! In: NIPS (2019)
34. Silver, D., S., J., S., K., etc., I.A.: Mastering the game of go without human knowledge. In: Nature (2017)
35. S.Pouya, K.Maya, C.Rama: Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In: ICLR (2018)
36. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: CVPR (2015)
37. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. In: ICLR (2013)
38. Wang, J., Zhang, H.: Bilateral adversarial training: Towards fast training of more robust models against adversarial attacks. In: ICCV (2019)
39. Wong, E., Rice, L., Kolter, J.Z.: Fast is better than free: Revisiting adversarial training. In: ICLR (2020)
40. Xie, C., Wu, Y., Maaten, L.V.D., Yuille, A.L., He, K.: Feature denoising for improving adversarial robustness. In: CVPR (2019)
41. Xu, Y., Xie, L., Zhang, X., Chen, X., Qi, G., Tian, Q., Xiong, H.: Pc-darts: Partial channel connections for memory-efficient differentiable architecture search. In: ICLR (2019)
42. Yang, Y., Zhang, G., Katabi, D., Xu, Z.: Me-net: Towards effective adversarial robustness with matrix estimation. In: ICML (2019)
43. Ying, C., Klein, A., Real, E., Christiansen, E., Murphy, K., Hutter, F.: Nas-bench-101: Towards reproducible neural architecture search. In: ICML (2019)
44. Zhang, C., Liu, A., Liu, X., Xu, Y., Yu, H., Ma, Y., Li, T.: Interpreting and improving adversarial robustness with neuron sensitivity. arXiv (2019)
45. Zheng, X., Ji, R., Tang, L., Wan, Y., Zhang, B., Wu, Y., Wu, Y., Shao, L.: Dynamic distribution pruning for efficient network architecture search. arXiv (2019)
46. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. In: ICLR (2016)

47. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: CVPR (2018)