Adversarial Robustness in Graph-Based Neural Architecture Search for Edge AI Transportation Systems

Peng Xu, Ke Wang, Mohammad Mehedi Hassan, *Senior Member, IEEE*, Chien-Ming Chen, *Senior Member, IEEE*, Weiguo Lin, Md. Rafiul Hassan, and Giancarlo Fortino

***Abstract*— Edge AI technologies have been used for many Intelligent Transportation Systems, such as road traffic monitor systems. Neural Architecture Search (NAS) is a typcial way to search high-performance models for edge devices with limited computing resources. However, NAS is also vulnerable to adversarial attacks. In this paper, A One-Shot NAS is employed to realize derivative models with different scales. In order to study the relation between adversarial robustness and model scales, a graph-based method is designed to select best sub models generated from One-Shot NAS. Besides, an evaluation method is proposed to assess robustness of deep learning models under various scales of models. Experimental results shows an interesting phenomenon about the correlations between network sizes and model robustness, reducing model parameters will increase model robustness under maximum adversarial attacks, while, increasing model paremters will increase model robustness under minimum adversarial attacks. The phenomenon is analyzed, that is able to help understand the adversarial robustness of models with different scales for edge AI transportation systems.**

***Index Terms*— Adversarial robustness, adversarial example, model compression and neural architecture search.**

Manuscript received 26 September 2021; revised 16 April 2022 and 17 June 2022; accepted 20 July 2022. Date of publication 26 August 2022; date of current version 2 August 2023. This work was supported in part by the Open Research Project of the State Key Laboratory of Media Convergence and Communication, Communication University of China, China, under Grant SKLMCC2021KF008; and in part by King Saud University, Riyadh, Saudi Arabia, through the Researchers Supporting Project, under Grant RSP2023R18. The Associate Editor for this article was W. Wei. *(Corresponding authors: Ke Wang; Mohammad Mehedi Hassan.)*

Peng Xu is with the Department of Computer Science, Harbin Institute of Technology (Shenzhen), Shenzhen 518055, China (e-mail: 19s051059@stu.hit.edu.cn).

Ke Wang is with the College of Information Science and Technology, Jinan University, Guangzhou 510632, China (e-mail: wangkehku@gmail.com).

Mohammad Mehedi Hassan is with the Information Systems Department, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia (e-mail: mmhassan@ksu.edu.sa).

Chien-Ming Chen is with the College of Computer Science and Engineering, Shandong University of Science and Technology, Shandong 266590, China (e-mail: chienmingchen@ieee.org).

Weiguo Lin is with the State Key Laboratory of Media Convergence and Communication, Communication University of China, Beijing 100024, China (e-mail: linwei@cuc.edu.cn).

Md. Rafiul Hassan is with the College of Arts and Sciences, University of Maine at Presque Isle, Presque Isle, ME 04769 USA (e-mail: md.hassan@maine.edu).

Giancarlo Fortino is with the Department of Informatics, Modeling, Electronics, and Systems, University of Calabria, 87036 Rende, Italy (e-mail: giancarlo.fortino@unical.it).

Digital Object Identifier 10.1109/TITS.2022.3197713

I. Introduction

ONE of the main applications of artificial intelligence technology in the field of transportation is machine vision tasks. For example, accessing road monitor video streams, and using deep-learning-based video analysis technology to screen out abnormal traffic behaviors in massive data,including motor vehicles. Non-motor vehicle violations, traffic incident detection models, etc. At present, many deep learning models have begun to run directly on the edge side. For instance, a single 1080P camera can generate about 80G data a day. If all the data is sent back to the cloud server for video analysis, it costs too much of the bandwidth and storage resources, while the valuable part of the data is far less than 15%. Therefore, many specific and practical solutions are to deploy deep learning models at the edge for anomaly detection. In comparison, edge intelligence is closer to the terminal side, and has significant advantages of reducing latency, reducing data transmission bandwidth, relieving pressure on cloud computing centers, and protecting data security and privacy. From traditional centralized data processing to edge-based data processing, integrating cloud with edge and complementing each other is the general trend.

The increasing terminal and edge devices are desired to embed deep learning technology, however, these devices have limited computing power, thus common deep learning models usually are not applicable. Therefore, some small models such as compressed models from common deep models are employed. However, obtaining small high-performance models is not easy. The current manual design of deep neural networks requires a huge amount of work in terms of architecture and structural hyperparameter tuning, and the design of model structures that perform well for specific tasks largely relies on the experience and inspiration of experts. In recent years, neural architecture search (NAS) [1] has emerged as an effective automatic machine learning method, freeing experts from tedious work by automatically searching for excellent architectures for specific tasks. However, most of the current neural structure search methods focus on basic tasks such as image classification. The NAS method comprehensively considers model operation efficiency indicators (Parameters, FLOPs, Latency) and search efficiency by balancing model

efficiency and accuracy performance, and finally obtains a series of efficient and compact Pareto-optimal neural networks. The architecture significantly surpasses the current artificially designed SOTA neural network. The NAS in the case of limited resources will make it easier to obtain the network structure for the deployment scenarios of edge computing devices and terminal embedded devices, which is conducive to the landing of intelligent applications and promotes the development of edge computing.

However, we found that the models generated from NAS are vulnerable to various adversarial attacks. First, the search space of NAS often comes from the same set of network structures. The differences in various settings lead to differences in their number of parameters. We find that adversarial samples generated from other network structures in the same search space can be effectively migrated to attack the targeted network. Second, we find that adversarial samples generated using network structures with close number of parameters tend to be more successful in migrating attacks on the targeted models. This implies that in the edge scenario powered by NAS system, the attack can be successfully applied to the system built by the target network structure, through the network with close number of parameters from the same search space. Actually, adversarial attacks have gradually become an important threat to NAS-enabled deep learning models for Intelligent Transportation System(ITS).

In this paper, it mainly targets the Graph-based NAS-enabled Edge AI for ITS. Fristly, one-shot NAS is used to generate sub models with different sizes. Secondly, graph-based method is used to select the sub models with the best performance. Finally, the robustness performance with different model scales is evaluated, and we find out a mechanism of adversarial robustness of NAS cross different model sizes. Besides, some popular adverarial attacks have been employed to generate advanced Adversarial Examples.

Our contribution lies in the following:

- Describe three adversarial attacking scenarios of one-shot NAS;
- Propose a graph-based method for one-shot NAS to select the best robust and accurate submodels under different scales;
- Two experimental findings: one is the balance compression ratio between accuracy and robustness, the other the correlation between model compression ratio and robustness.

II. Related Work

In many application scenarios, only detecting the Adversarial Examples is not enough, it needs to know the correct prediction of the Adversarial Examples. For example, in intelligent driving, if the safety system only detects the speed limit signs on the roadside as Adversarial Examples, it is very dangerous to park blindly. At this time, if the correct prediction of Adversarial Examples can be identified, many accidents can be avoided.

Traditional techniques making neural network models more robust, such as weight decay, randomly dropping neurons or neuron connection edges, are often ineffective against Adversarial Examples. Some researchers propose to enhance the robustness by changing the non-linear transformation used by the model, and [3] distilled a large network into a small network [4]. Some other works used another statistical model to detect Adversarial Examples [5], [6], [7]. However, these methods have all been proved to be unsuccessful [8].

Later, some researchers used regularization methods [9], [10], [11] and adversarial training methods to improve robustness [12], [14], [15], [34]. However, after adversarial training, the performance of neural networks on normal samples tends to decline, and the essential reason for the compromise between accuracy and robustness has not yet been found out. Adversarial training is indeed an effective means of regularization, but it does so by inappropriately reducing the neural network hypothesis space. Furthermore, local errors still occur when going beyond the bounds designed for adversarial training [16]. Florian Tramèr1 [17] pointed out that the model trained with a single adversarial is still vulnerable, because the discriminative hyperplane of the model varies significantly near the data points. The discriminative hyperplane hinders the first-order approximation attack based on the model loss, but it cannot reject samples from Adversarial Examples Migration black box attack.

The distillation method [18] is an excellent method in recent years to imitate a large model with a small model. But then Carlini and Wagner [19] prove that the distillation mechanism is insufficient against adversarial sample attacks by modifying the distillation mechanism on the standard attack.

Recently, researchers have proposed a quantum-classical hybrid convolutional neural network H-QCNN [33] based on quantum computing, which provides a new solution for the study of adversarial robustness in deep learning. However, it is found in the experiment that H-QCNN is prone to overfitting during training, which leads to the problem of the model's validation accuracy and adversarial robustness being reduced. In addition, the researchers designed a "retinal fixation point" model based on the biological vision mechanism, and non-uniformly sampled the picture under different fixation points, and then sent it into the neural network; and designed a biologically inspired mechanism model. The cortical fixation point model divides the standard ResNet [2] into different branches, each branch handles a receptive field of one scale, and then the results of the different branches are spliced together. However, its improved robustness is lower than that of adversarial training models on some datasets, and its effectiveness and universality need to be further explored.

In addition, some scholars have studied the classification rejection method [20], [21], that is, the system can choose not to classify for some of the observed samples. For example, the reject option is chosen when the class-conditional posterior probability is close. Hendrycks [22] pointed out that correctly classified samples tend to have larger maximum class-conditional posteriors than samples that are misclassified and not in the probability distribution.

Model compression is widely used in the mobile deployment of deep learning [24]. However, the compressed model also suffer from the threat of Adversarial Examples.

Zhao *et al.* [25] found that Adversarial Examples produced by the uncompressed model remain highly transferable for the pruned models. While, Gui *et al.* [26] proposed a adversarially trained model compression framework to improve the robustness of compressed models. They integrated pruning, factorization, quantization into constraints to construct an optimization formulation. They have proved their algorithm obtained more favorable trade-off among model size, accuracy and robustness.

III. Methodologies

A. Preliminary

In this subsection, we first briefly introduce the concept of weight sharing mechanism in NAS, and then we formally define the concept of adversarial robustness under different model scales.

1) Weight Sharing NAS: The main goal of One-Shot is to optimize a SuperNet $\mathcal{N}$ containing all structures in a given compression model space $\mathcal{A}$. The weights θ of the SuperNet $\mathcal{N}$ are shared by all its subnetworks (called architectures or paths), i.e., $a \in \mathcal{A}$. In a trained SuperNet, the optimal network structure results can be obtained by searching and evaluating them. A usual One-Shot NAS problem can be constructed as a two-stage optimization problem, according to the definition in the work of Guo *et al.* [28] and Su *et al.* [29].

$$a^* = \arg\max_{a \in A} Acc_{val}(a, \theta^*_{\mathcal{A}}(a)) \tag{1}$$

$$\text{s.t. } \theta^*_{\mathcal{A}} = \arg\min_{\theta} \mathcal{L}_{train}(\mathcal{A}, \theta) \tag{2}$$

where Acc_{val} defines the accuracy of each structure a on the validation data set D_{val} and $\mathcal{L}_{train}(\mathcal{A}, \theta) = \mathbb{E}_{a \in \mathcal{A}}[L_{train}(a, \theta)]$ denotes the expectation value of the loss generated by the SuperNet on the training data set D_{train} by randomly sampling a path and optimizing its corresponding weights corresponding to it. Also, since it is computationally infeasible to traverse all the structures in this large compression model space, the search problem for the optimal structure in Equation (2) is often solved using an efficient search, such as an evolutionary algorithm (Deb *et al.* [32]).

2) The Defination of Adversarial Robustness: Network Robustness refers to a degree of resistance of the network to Adversarial Examples, and network robustness under bounded adversarial perturbations can be defined as follows.

$$Acc_{adv} = \mathbb{E}_{x \in D}[Accuracy_{x' \in S}(a, \theta_a)] \tag{3}$$

where $S := x' : \|x - x'\|_p \leq \epsilon$ defines the allowed Adversarial Examples at l_p distance, a denotes the model corresponding structure, θ_a denotes the weight parameter corresponding to the model structure, and D denotes the data distribution of the model input, $Accuracy$ means to measure the accuracy on the given data.

A very effective attack is the use of Projected Gradient Descent (PGD) for adversarial perturbation generation, which can have a large impact on the performance of the network. In our study, we focus on PGD adversarial attacks with attack degree less than ϵ at n-step ($n = 7$) and use the PGD attack approach to obtain robust accuracy corresponding to networks of different sizes.

B. Three Scenarios of Adversarial Attacking Based on Model Compression

In this paper, we are interested in the correlation between adversarial attacks and model compression ratio, and we investigate three specific adversarial attack scenarios:

1) The Adversarial Examples generated from and applied on the same model.
2) The Adversarial Examples generated from uncompressed model, but applied on the fully compressed model and partly compressed model.
3) The Adversarial Examples generated from fully compressed model, but applied on the uncompressed model and partly compressed model.

In the first scenario, Adversarial Examples are generated from and applied on the same compressed model (or on the uncompressed model). Attackers have fully access to the compressed models with different compression ratios (or the uncompressed model). This is the situation where attackers hack into a single devices and use the compressed model (or the uncompressed model) to carry out adversarial attacking on the same devices or the same type of product.

In the second scenario, it is assumed that the uncompressed model is exposed to the attackers, and the information of compressed models is unknown to the attackers. Attackers can use the Adversarial Examples generated from the uncompressed model to attack the compressed models that derive from the uncompressed model, including the fully compressed model and partly compressed model. The key issue is the generalizability of Adversarial Examples, which genetated from the uncompressed model, over the partly compressed models and the fully compressed model.

The third scenario assumes that only the fully compressed models are accessible to attackers, and attackers generate Adversarial Examples using fully compressed models to attack other partly compressed models and the hidden uncompressed models. The key issue is the generalizability of Adversarial Examples genetated from the fully compressed model, over other partly compressed models and the uncompressed model.

Given a compression model size space $\mathcal{A}$, assume that the compression model space consists of multiple stages, and that the stage consists of multiple blocks. We define the network structure a as $a = (d_1, \ldots, d_n, w_1, \ldots, w_n, k_1, \ldots, k_n)$, where n is the number of stages contained in the network, $d_n \in D_n$, $D_n \subset \mathbb{N}^+$ denotes the number of blocks contained in the n-th stage, $w_n \in W_n$, $W_n \subset \mathbb{N}^+$ denotes the number of base channels of blocks in the n-th stage, $k_n \in K_n$, $K_n \subset \mathbb{N}^+$ denotes the size of the convolution kernel in the n-th stage, then $a \in \mathcal{A}$, $\mathcal{A} = D_1 \times \cdots \times D_n \times W_1 \cdots \times W_n \times K_1 \cdots \times K_n$.

For two different structures a and b ($a, b \in \mathcal{A}$) in the compression model space, a partial order relation on the compression model space $\mathcal{A}$ can be defined as follows: $a \leq b$, when and only when $a_1 \leq b_1$ or there exists an integer i such that $a_1 = b_1, a_2 = b_2, \ldots, a_i \leq b_i$.

Then the fully compressed model in the compression model space $\mathcal{A}$ can be defined as the minimum value of the network structure:

$$a_{\text{fullyComp}} = \min((x_1, \ldots, x_n, w_1, \ldots, w_n, k_1, \ldots, k_n)) \tag{4}$$

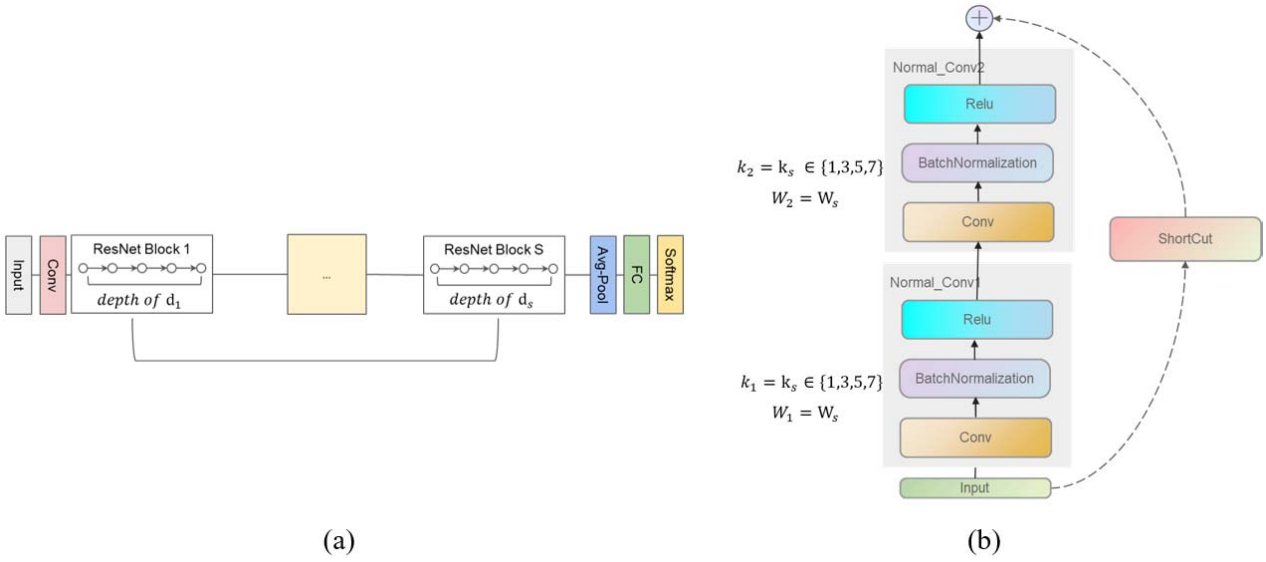


Fig. 1. The description of laying for our search space, the downsampling strides of each stage in the whole network are set to [1, 1, 2, 2, 2] respectively, and no maximum pooling operation is used before the first convolution operation: (a) the decription of the NAS searchspace, and the depth of each stage is range from [1, 1, 1, 1, 1, 1] and [1, 2, 2, 3, 3]; (b) the structure of the ResBlocks, which can be seen as a basic ResNet block with dynamic number of channels and dynamic convolutional kernel size, and the width of each blocks is set to be range from [32, 32, 64, 128, 256] [64, 96, 160, 320, 640].

The uncompressed model in the compression model space can be defined as the maximum value of the network structure in the compression model space $\mathcal{A}$:

$$a_{\text{unComp}} = \max((d_1, \ldots, d_n, w_1, \ldots, w_n, k_1, \ldots, k_n)) \quad (5)$$

Assuming the network structure $a \in \mathcal{A}$, we use the SuperNet weights θ obtained by One-Shot training to initialize the model corresponding to the structure a, and we can define the robustness of the model under the Adversarial Examples generated by the fully compressed network as:

$$Acc_{\text{fullyComp\_adv}} = \mathbb{E}_{x \in D}[Acc_{x' \in S_{\text{fullyComp\_adv}}}(a, \theta_a)] \quad (6)$$

where $S_{\text{fullyComp\_adv}} := x'_{\text{fullyComp}} : \|x - x'_{\text{fullyComp}}\|_p \leq \epsilon$ defines the Adversarial Examples generated at l_p distance corresponding to the smallest scale model $a_{\text{fullyComp}}$.

The robustness of the defined model under the adversarial sample generated by the maximum size network is expressed as:

$$Acc_{\text{unComp\_adv}} = \mathbb{E}_{x \in D}[Acc_{x' \in S_{\text{unComp\_adv}}}(a, \theta_a)] \quad (7)$$

where $S_{\text{unComp\_adv}} := x'_{\text{unComp}} : \|x - x'_{\text{unComp}}\|_p \leq \epsilon$ defines the Adversarial Examples generated at l_p distance corresponding to the minimum size model a_{unComp}.

We respectively define the traditional accuracy under the attack from $S_{\text{fullyComp\_adv}}$ and $S_{\text{unComp\_adv}}$ as $top1_{\text{fullyComp\_adv}}$ and $top1_{\text{unComp\_adv}}$, which means that the answer with the highest probability of the model answer must be exactly the expected answer under the Adversarial Examples generated from $a_{\text{fullyComp}}$ and a_{unComp}. Besides, We define the traditional accuracy under the attack from the original model as $top1_{self\_adv}$.

C. The Compression-Aware Training Based on One-Shot SuperNet

In this subsection, we introduce the main component used in our research work: the **SuperNet** in One-Shot paper,

Algorithm 1 Three Scenarios of Adversarial Attacking

```
Initialization: N_data ← dataset.size(),
  min_model ← the minium model in the defined Search Space,
  self_model ← the selected model to be attacked,
  max_model ← the maxium model in the defined Search Space,
  attacking_modes ← [max, self, min]
Output: dataset ← predifineddicttostoretheattackeddata
for i : 0 → N_data do
  for m : 0 → attacking_modes do
    # select model according to the attacking_mode
    model ← select_model(m)
    produce attacked data using PGD attacking method for specific
    model
    append to the dataset[m]
  end for
end for
for m : 0 → attacking_modes do
  evaluate the Acc_adv of model for different attacking mode m
end for
```

thanks to the technique of large model distillation introduced in the training, we can easily obtain subnets with high accuracy at different model parameter sizes. In the training part of the SuperNet, we follow the setup in the work of Cai *et al.* [30] and Liu *et al.* [31] and adjust the model search space accordingly.

1) The Defination of Compression Model Space: For the compression model space, unlike the common NAS search space of Darts search space and MobileNet-V2 search space, we use the ResNet, which is a common backbone for many downstream task, as the backbone of the base models.

In Fig.1.a, we can divide the compression model space into multiple stages, where each stage is built from multiple ResBlocks, and each ResBlock is constructed as shown in Fig.1.b.

Then the compression model space can be seen as a fixed stage, each stage can search dynamic depth, dynamic

number of channels and dynamic convolutional kernel size of the SuperNetwork. Subnetworks of different depths, different number of channels and different kernel size of convolution can be initialized with the weights of this SuperNet.

2) The Compression-Aware Training Method Based on One-Shot SuperNet: For the training of the SuperNet, in order to obtain subnetworks that perform well at different compression scales, we use distillation, trying to improve the accuracy of all the subnetworks in the entire compression model space.

In particular, during the forward process of each network, we first calculate the loss corresponding to the maximum network, but do not perform backward propagation. Subsequently, we randomly sample two subnetworks from the compression model space between the largest subnetwork and the smallest subnetwork, and compute the entropy loss corresponding to the subnetwork compared to the true label, while accumulating the distillation loss corresponding to the maximum network and the random two subnetworks. Finally, we compute the loss corresponding to the minimum network compared to the true label, and accumulate the distillation loss corresponding to the maximum network and the minimum subnetwork. With the form of pseudo-algorithm, we express the SuperNetwork training as Algorithm 2.

Algorithm 2 One-Shot SuperNet Training

```
Initialization: sampleSubnetNum ← 4,
  total_iterations ← epoch × dataset.size()/batchSize,
  model ← theSuperNet
  sample_modes ← [max, random, random, min]
for i : 0 → total_iterations do
  for c : 0 → sampleSubnetNum do
    # returns to subnet setting (dict with depth, out channel
    etc) and sample strategy
    settings, mode = adjust_model(i, c)
    # forward
    logits = model(input)
    loss = criterion(logits, target)
    # calculate distiller loss, left over the maxium network
    mimic_loss = get_distiller_loss(mode)
    loss += mimic_loss
    # compute and update gradient
    loss.backward()
  end for
  # add all subnets loss compute and update gradient
  optimizer.step()
end for
```

D. Graph Based Method for Predicting Robust Architectures

In fact, since the search space can be divided into different stages, each of them has its own independent architectural parameters configuration. If we consider each stage as a node $\mathcal{V}$, the message passing between each stage as an edge $\mathcal{E}$), and the whole network as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we can employ GNN to learn the representation of the network structure and thus predict in advance the accuracy of different network structures under different migration counterattack settings.

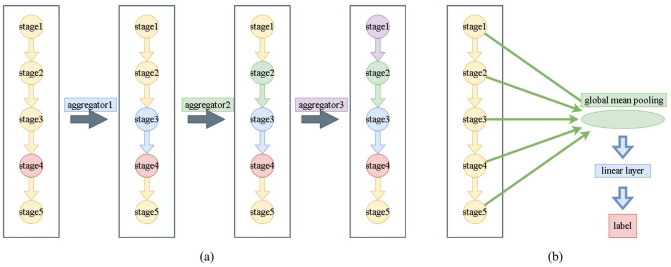


Fig. 2. (a) Taking stage 4 as an example, the aggregation process of neighbor node features when passing through three convolutional layers; (b) After completing the three convolutions, go through the global average pooling layer and the linear layer to obtain the label of the graph.

First, we give a formal defination of GNN here. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote the undirected graph having node feature vectors $\boldsymbol{x}_i$ for all $i \in \mathcal{V}$, and its adjacency matrix is $A \in \{0, 1\}^{|\mathcal{V}|\times|\mathcal{V}|}$.

The corresponding normalized Laplacian matrix is defined as $L = I - D^{-1/2}AD^{-1/2} = U\,\mathrm{diag}(\boldsymbol{\lambda})U^T$. Mathematically, we have the updating process of node v's embedding at lth layer as follows,

$$\mathbf{h}_v^l = \phi\left[W^l \cdot \mathbf{AGG}_{\text{node}}\left(\left\{\mathbf{h}_u^{l-1}, \forall u \in N(v)\right\}\right)\right], \qquad (8)$$

where $N(v)$ denotes the neighbours; W^l denotes trainable weight shared by all nodes; ϕ is an activation function, taking ReLU as default.

Three data sets Self_Adv_Sample_Accuracy, Min_Ad_Sample_Accuracy, and Max_Adv_Sample_Accuracy were built based on the top1 accuracy of these 500 neural networks when attacked with Adversarial Examples generated against themselves, the smallest network, and the largest network. We regard each neural network as a directed graph, as shown in Fig. 2.

The top1 accuracy of the neural network under an adversarial attack is the label of the graph. Each stage in the network is regarded as a node, and a directed line is established between the two stages before and after. Beside, the previous stage points to the next stage. Each node has three features: stage depth, number of channels, and convolution block size.

For the above three data sets, a graph neural network is trained to predict the top1 accuracy of the neural network against attacks based on the characteristics of each stage of the neural network.

Figure 6.a shows the process of convolution in three data sets, and Figure 6.b shows the processing after three convolutional layers.

IV. Experiments and Analysis

In order to analyze the relevance between network size and model robustness, we need to conduct case studies to analyze the correlation. Specifically, we conducted experiments on the CIFAR10 [27], CIFAR100 respectively. Among the experimental network architectures, we mainly focus on the compression model space derived from ResNet. It should be noted that our approach can be easily extended to other compression model spaces, such as the compression model space of MobileNet, the compression model space of DenseNet, and the compression model space of EfficientNet.

A. Experimental Settings

1) SuperNet Training Setup: Following the approach described in Section 3, in our experiments we adopt ResNet as the basic backbone of the compression model space. The basic compression model space used on CIFAR10 and CIFAR100, can be divided into 5 stages: s_1, s_2, s_3, s_4, s_5, each stage is composed of d_1, d_2, d_3, d_4, d_5 ResBlock, where the basic construction of each block is shown in the figure1.b, where the width of each block and the corresponding kernel size of each block are also part of the whole compression model space.

We use the same training space on CIFAR10 and CIFAR100 for the whole training space setup. The downsampling strides of each stage in the whole network are set to $[1, 1, 2, 2, 2]$ respectively, and no maximum pooling operation is used before the first convolution operation. The minimum value of the depth of each stage in the whole training space is $[1, 1, 1, 1, 1, 1]$ respectively, i.e., the minimum number of blocks contained in each stage is $dmin_{s_i}$. The maximum values of the depth of each stage in the training space are $[1, 2, 2, 3, 3]$, i.e., the maximum number of blocks in each stage is $dmax_{s_i}$.

The minimum number of channels per stage in the training space is [32, 32, 64, 128, 256], and the number of channels per block in each stage is at least $wmin_{s_i}$. The maximum values of the depth of each stage in the whole training space are [64, 96, 160, 320, 640], and the number of channels per block in each stage is at most $wmax_{s_i}$.

The minimum size of the convolution kernels in each stage in the training space is $[3, 3, 3, 3, 3]$, and the size of the convolution kernels in each block in each stage is at least $kmin_{s_i}$. The maximum value of the depth of each stage in the whole training space is $[7, 3, 3, 3, 3]$, and the size of the convolutional kernel in each block in each stage is at most $kmax_{s_i}$.

In the training process of the SuperNet, SGD was used as an optimizer and a scheduler with cosine decay to adjust the learning rate. In the CIFAR 10 data set, our initial learning rate is set to 0.2, BatchSize is set to 512, and training time is set to 200 epochs. weight decay is set to 0.0005, and SGD mometunm is set to 0.9.

In the experiments on the CIFAR100 data set, our initial learning rate is set to 0.2, BatchSize is set to 512 (using two GPUs for parallel training), and training time is set to 250 epochs.

2) Across Model Compression Ratio Scales Robustness Testing Setup: PGD was used to generate the Adversarial Examples in White-Box attack, and it is worth mentioning that the method can be easily generalized to other Adversarial Examples generation methods for testing the robustness against Adversarial Examples. We focus on the untargeted PGD Adversarial Examples generation method, where the noise bound is limited to $\epsilon = 8/255$, the number of iterations of the PGD attack to 7, and the step size of the PGD attack to $\alpha = \epsilon \times 1.25/steps$.

For the cross network parameter robustness testing experiments on CIFAR 10 and CIFAR 100, we sampled 500 networks from the subnetworks with $Flops$ metrics with compression ratios between $100 - 1000$, and then tested the robustness of their own Adversarial Examples on the test set, the robustness of the Adversarial Examples generated by the largest scale model, and the robustness of the Adversarial Examples generated by the smallest scale model for each of these 500 networks. The robustness of the Adversarial Examples generated by the smallest model is tested. After collating the obtained performance metrics, in the next section we use a data analysis approach to find the correlation between robustness and model parameter scale.

CIFAR 10 is a color image data set that more closely resembles a universal object. CIFAR10 is a small data set for identifying universal objects, organized by Hinton, Alex Krizhevsky and Ilya Sutskever. There are RGB color images of 10 categories: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

The size of each image is 32×32, and there are 6000 images for each category, with a total of $50,000$ training images and $10,000$ test images in the data set.

CIFAR 100 is similar to CIFAR 10 and was compiled by Hinton, Alex Krizhevsky and Ilya Sutskever. CIFAR 100 has 100 classes, each containing 600 images. The 100 classes in CIFAR 100 are divided into 20 superclasses. Each image has a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs).

3) Graph-Based Method for Predicting Robust Networks: The graph neural network model consists of three SAGE convolutional layers, one global average pooling layer, and one linear layer. The outputs of the first two SAGE convolutional layers and the global average pooling layer are processed by the Leaky ReLU activation function. The hidden layer size is set to 128. MSE loss function and Adam optimizer are employed. The Adam optimizer has a learning rate of 0.003, beta1 of 0.9, beta2 of 0.999, and epsilon of 1e-8.

In addition, GCN convolutional layer, GAT convolutional layer and linear layer are employed to replace the middle SAGE convolutional layer of the model, other conditions remain unchanged, and compare the effects of the four.

After each convolutional layer, each node can aggregate the features of the nodes that point to that node. Taking stage4 as an example, after the first convolutional layer, stage4 aggregates the features of stage3, and stage3 aggregates the features of stage2; after the second convolutional layer, stage4 aggregates the features of stage3 again, because stage3 has been aggregated before. The features of stage2, so at this time stage4 actually aggregates the features of stage2; and so on, after the third convolutional layer, stage4 aggregates the features of stage1.

All nodes go through the global average pooling layer and merge into a node whose embedding is the average of the embedding of all nodes. Finally, through the linear layer, the predicted label is obtained.

B. The Analysis of Three Scenarios of Adversarial Attacking Based on Model Compression

Tab.I shows the perfromance of fully compressed model, partly compressed model and uncompressed model on CIFAR10 and CIFAR100 data set under three different adversarial attacking scenarios. Apart from showing the accuracies of models without any adversarial attacks ($top1$), we present

TABLE I
CASE STUDY

data set	MODELS	FLOPs	PARAMS	$top1$	$top1_{self\_adv}$	$top1_{\text{fullyComp\_}adv}$	$top1_{\text{unComp\_}adv}$
CIFAR10	$\text{Model}_{\text{fullyComp}}$	63.802	1.226	91.830	20.770	20.770	51.840
	$\text{Model}_{\text{partlyComp}}$	555.423	11.164	95.970	21.430	36.080	28.180
	$Model_{\text{unComp}}$	1198.988	26.771	96.010	23.810	39.260	23.810
CIFAR100	$\text{Model}_{\text{fullyComp}}$	63.825	1.249	61.980	4.080	4.080	19.000
	$\text{Model}_{\text{partlyComp}}$	555.469	11.21	75.650	5.800	21.210	13.940
	$Model_{\text{unComp}}$	1199.045	26.828	76.600	7.790	23.850	7.790

the robustness of the three models with three different attack scenarios. The first scenario corresponds to the Adversarial Examples generated from the same model, the second scenario and third scenario corresponds to the Adversarial Examples generated from the fully compressed model and the uncompressed model respectively.

The first conclusion drawn from the experimental data is that the three adversarial attacking scenarios is effective over models with different compression ratio. In all scenario, the Adversarial Examples generated from the same model keep effective even if fully compressed. Besides, the the Adversarial Examples generated from the fully compressed model and uncompressed model show their generalizability on their targeted compression ratio. In the first scenario, the fully compressed model, partly compressed model and uncompressed model drop accuracy of $71.06, 74.54, 72.20$ on CIFAR10 respectively, $57.90, 69.85, 68.81$ on CIFAR100 respectively. In the second scenario, the partly compressed model and uncompressed model drop accuracy of $59.89, 56.75$ on CIFAR10 respectively, $54.44, 52.75$ on CIFAR100 respectively. In the third scenario, the fully compressed model and partly compressed model drop accuracy of $39.99, 67.79$ on CIFAR10 respectively, $42.98, 61.71$ on CIFAR100 respectively.

Besides, from observation of the experimental data, another conclusion can be made that the robustness have some relationship with the model compression ratio. The uncompressed model shows great resistance to the Adversarial Examples generated by the fully compressed model. On the other hand, the partly compressed model shows better resistance to the Adversarial Examples generated by the fully compressed model and worse resistance to the Adversarial Examples generated by the uncompressed model.

Therefore, we conduct a more specific experiment to study the correlation between robustness and model compression ratio.

C. Experimental Investigation on the Balance Points in the Model Size Scale Against Adversarial Examples

In our experiments, we found an interesting phenomenon about the balance points of the Adversarial Examples. Observed on different model compression scales, there is a region of balance between the accuracy and robustness, in which an optimal trade-off between the accuracy and robustness of the model can be achieved.

Specifically, in our experiments, we find a Pareto-optimal decision frontier on these accuracy and robustness. Surprisingly, on both data sets yield the models in the balance region congruously hold flops range from 400 to 800.

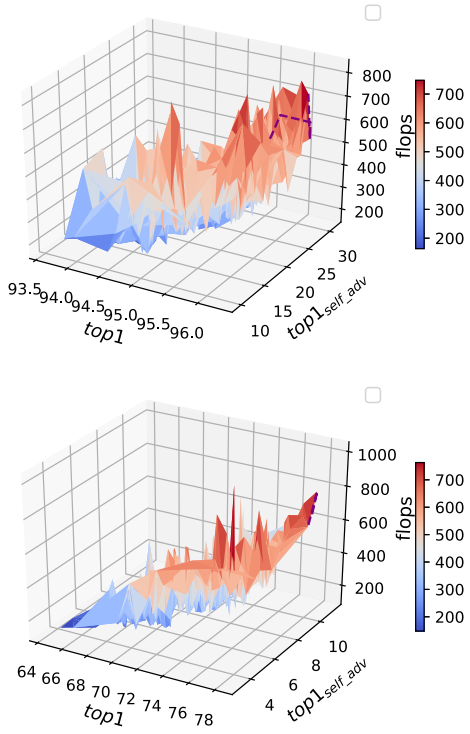


Fig. 3. Balance points visualization on CIFAR10(top) and CIFAR100(bottom).

We then infer that there is an optimal equilibrium point in the compression model when the model size varies in different model size scales, and that this equilibrium point is generalizable to model sizes that are applicable across different tasks.

D. The Correlation Between Adversarial Robustness and Compressed Model Sizes

1) CIFAR10 Experimental Results: First, we used the One-Shot approach to train the SuperNet on CIFAR 10, where the SuperNet contains the accuracy of the model obtained by distillation at different compression scales. Based on the SuperNet obtained from the CIFAR 10 training set by One-Shot training, we extracted 500 submodels of different sizes from the SuperNet Flops range between $100 - 1000$. After that, we tested the robustness of these submodels under their own scale, the robustness performance under the smallest scale network, and the robustness performance under the largest scale network, respectively. The performance of these submodels with different compression ratios is shown in Fig.4.

From Fig.4, we can see that for the submodels corresponding to different Flops and Params (at different compression ratios), the effect of the Adversarial Examples generated at

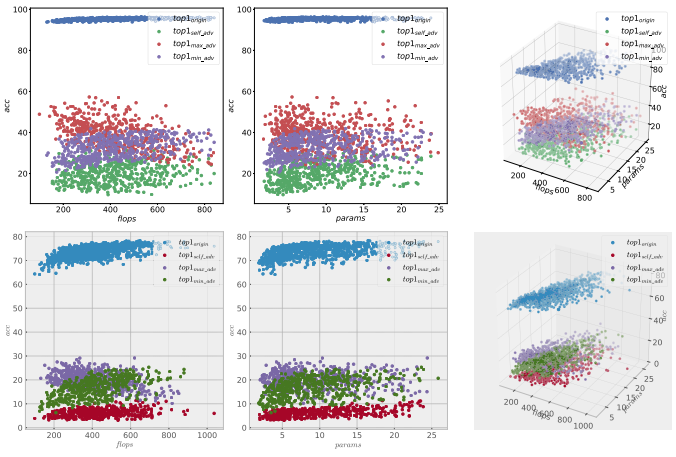


Fig. 4. (1) The robustness performance under different parameter scales of submodels corresponding to different Flops(top, left) and Params(top, middle), and both(top, right) on CIFAR 10 data set; (2) The robustness performance under different parameter scales of submodels corresponding Flops(bottom, left) and Params(bottom, middle), and both(bottom, right) on CIFAR 100 data set.

their own scale is the most significant, while the effect of the Adversarial Examples generated at the largest scale decreases continuously as the scale of the model rises. The effect of the Adversarial Examples generated at the smallest scale will keep increasing as the scale of the model rises. After that, we extracted the Pareto bounds for these 500 models based on the Flops and Top1 metrics, where we extracted the Pareto bounds based on the assumption that the models with the best accuracy at different compression ratios are usually deployed in industrial deployment scenarios. For the extracted models with different compression ratios on the Pareto boundary, we compare their performance under robustness tests corresponding to different scales, and the test results are shown in Fig.6.

One of the purpose is discussing the critical phenomenon, which is pareto critical point, called pareto optimization. In fact, robustness and accuracy are two objectives. In essence, it is a multi-objective optimization problem. Its challenge is that it is often difficult to find a single solution to make all sub objectives optimal at the same time. Therefore, it is generally to find the Pareto optimal solution, which mainly involves the trade-off problem between multiple objectives.

Subsequently, we extracted three sets of models under the Pareto boundary, and for the models corresponding to different parameter scales under the Pareto boundary, we tested their original accuracy and robust accuracy under the Adversarial Examples corresponding to different scales of parameter. As shown in Fig. 5, where $\Delta top1_{self\_adv}$, $\Delta top1_{\text{unComp}\_adv}$ and $\Delta top1_{\text{fullyComp}\_adv}$ denote the Adversarial Examples generated by the model corresponding to different parameter scales.

2) Experimental Results on CIFAR 100: We use linear regression to fit the relationship between $Flops$ and $\Delta top1$ of the model, and also use correlation coefficients to evaluate the relationship between the two variables. For the correlation coefficients of the attack degree and flops at different scales in Table.II, at the significance level of $\alpha = 0.01$, it can be judged that: $\Delta top1_{\text{unComp}\_adv}$ has a highly significant positive correlation with the Flops index, and $\Delta top1_{\text{fullyComp}\_adv}$ has

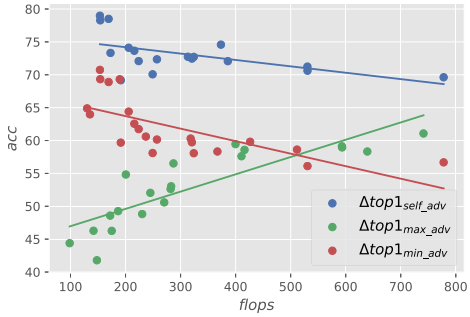


Fig. 5. Linear regression to fit the relationship between $Flops$ and $\Delta top1$ of $self\_adv$, fullyComp_adv, unComp_adv on CIFAR 10.

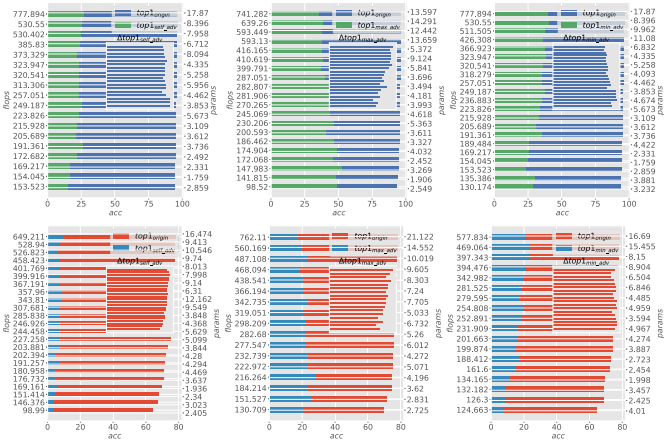


Fig. 6. (1) The $\Delta top1_{self\_adv}$(top, left), $\Delta top1_{\text{unComp}\_adv}$(top, middle) and $\Delta top1_{\text{fullyComp}\_adv}$(top, right) corresponding robustness (accuracy reduce) under attack from different parameter scales on CIFAR 10 data set; (2) The $\Delta top1_{self\_adv}$(top, left), $\Delta top1_{\text{unComp}\_adv}$(top, middle) and $\Delta top1_{\text{fullyComp}\_adv}$(top, right) corresponding robustness (accuracy reduce) under attack from different parameter scales on CIFAR 100 data set. (Flops and Params indexes are shown in y-axis.)

TABLE II

CORRELATION MATRIX BETWEEN FLOPS/PARAMS AND $\Delta top1_{self\_adv}$/$\Delta top1_{\text{FULLYCOMP}\_adv}$/$\Delta top1_{\text{UNCOMP}\_adv}$ ON CIFAR 10

	FLOPs	PARAMS
$\Delta top1_{self\_adv}$	-0.550	-0.520
$\Delta top1_{\text{fullyComp}\_adv}$	0.853	0.722
$\Delta top1_{\text{unComp}\_adv}$	-0.691	-0.579

a highly significant negative correlation, $\Delta top1_{self\_adv}$ does not have a significant correlation.

Similar to the processing on CIFAR 10, we used the One-Shot approach to train the supergrid on CIFAR 100, and then extracted 500 submodels of different sizes from the supergrid Flops ranging from $100 - 1000$. The performance of these submodels with different compression ratios on the CIFAR 100 data set is shown in Fig.4.

The experiments on the CIFAR 100 data set also verify that for the models corresponding to different network sizes, the effect of adversarial attacks against their own scale is the most significant, while the effect of attacks against the smallest size of Adversarial Examples decreases as the model parameter scale increases, and the effect of attacks against the largest parameter scale of Adversarial Examples increases.

Similarly, the Pareto bounds corresponding to these 500 models on our CIFAR 100 data set are then compared

TABLE III

CORRELATION MATRIX BETWEEN FLOPS/PARAMS AND $\Delta top1_{self\_adv}/\Delta top1_{\text{FULLYCOMP}\_adv}/\Delta top1_{\text{UNCOMP}\_adv}$ ON CIFAR 100

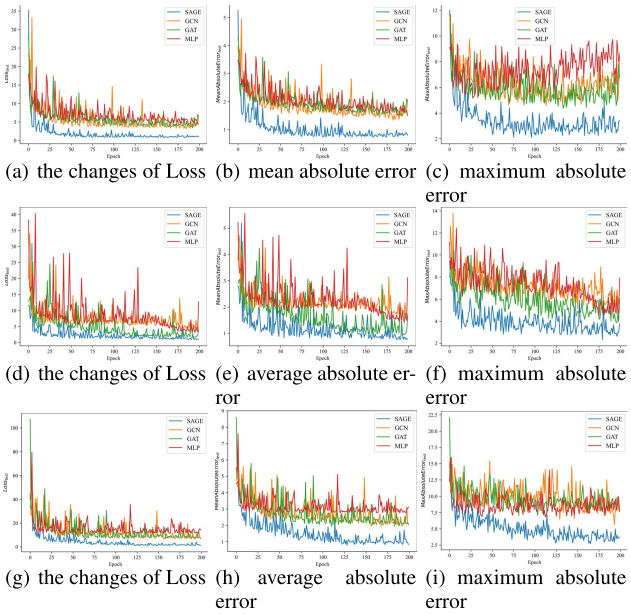
	FLOPs	PARAMS
$\Delta top1_{self\_adv}$	0.799	0.685
$\Delta top1_{\text{fullyComp}\_adv}$	0.859	0.790
$\Delta top1_{\text{unComp}\_adv}$	-0.691	-0.432

to their performance under robustness tests corresponding to different scales, and the results are shown in Fig.6.

Linear regression is used to fit the relationship between $Flops$ and $\Delta top1$ of the model (as shown in Fig. 8), where $\Delta top1_{self\_adv}$, $\Delta top1_{\text{unComp}\_adv}$ and $\Delta top1_{\text{fullyComp}\_adv}$ denote the values corresponding to the decrease in model accuracy on the Pareto boundary due to the Adversarial Examples generated by the model at different scales, respectively. The correlation coefficient is also used to evaluate the relationship between the two variables. For the correlation coefficients of the attack degree and flops at different scales in Table II, at the significance level of $\alpha = 0.05$, we can judge that: $\Delta top1_{\text{unComp}\_adv}$ has a significant positive correlation with the Flops indicator, and $\Delta top1_{\text{fullyComp}\_adv}$ has a significant negative correlation with the Flops indicator, but There is a significant correlation between $\Delta top1_{self\_adv}$ and Flops indicator on CIFAR 100.

Using the above four schemes, 200 epochs were trained on each of the three data sets. Figure 7 shows how MSELoss, mean absolute error, and maximum absolute error change during training. Table 4 shows the training results of the model. During the training process, the loss gradually decreased in the first 80 epochs and changed little after that. Combining the training conditions on the three data sets, the effect of the SAGE convolutional layer is always significantly better than the other three schemes, and it can predict the use of the neural network for itself, the minimum network, and the maximum network when the average absolute error does not exceed 0.79. The top1 accuracy of the generated Adversarial Examples when attacked. The effect of MLP is relatively poor, because MLP does not consider whether the nodes are adjacent, while SAGE, GCN, and GAT all aggregate the features of neighbor nodes, considering the adjacency relationship between nodes. Compared with GCN and GAT, SAGE does not generate separate embeddings for each node, but learns a function that generates embeddings by sampling and aggregating features from neighbor nodes, so it can better learn node features The relationship with the label of the directed graph.

The results show that. 1. that robustness tests using ML approach for structures with different attack settings can converge to the ideal solution, demonstrating the usefulness of the ML approach for training. 2. the performance of SAGE, GAT and GCN combined with Graph information is much better than that of MLP alone, which proves that Graph information can help predict robustness. 3. that SAGE generally performs better in prediction tasks and is suitable for handling information from NAS data sets where the stage is a node.



(a) the changes of Loss (b) mean absolute error (c) maximum absolute error

(d) the changes of Loss (e) average absolute error (f) maximum absolute error

(g) the changes of Loss (h) average absolute error (i) maximum absolute error

Fig. 7. The first line are on the data set $Self\_Adv\_Sample\_Accuracy$, the second line are on the data set $Min\_Adv\_Sample\_Accuracy$, the third line are on the data set $Max\_Adv\_Sample\_Accuracy$.

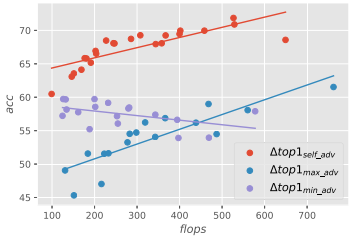


Fig. 8. Linear regression to fit the relationship between $Flops$ and $\Delta top1$ of $self\_adv$, fullyComp_adv, unComp_adv on CIFAR 100.

V. CONCLUSION

To obtain the performance of the models under different compression ratios required in this study, the concept of SuperNet is introduced to use the technique of weight sharing, and preserve the weights of the models under different compression ratios in the same network. It allows us to easily obtain model weights that perform well at various compression ratio, while the process of testing Adversarial Examples across network sizes will be more convenient because these networks coexist in a single SuperNet.

It analyzes two valuable phenomenons of the correlation between model size and adversarial robustness for max and min attacks. First, reducing model parameters will increase model robustness under maximum adversarial attacks, due to the fact that the deleted parameters prevent the model from overfitting. Second, the robustness of the model increases when the parameters of the model are increased under minimum adversarial attacks, because the remaining parameters act as regularization.

In future, we plan to study the generalization difference under different model sizes.

REFERENCES

[1] G. Bender, P. J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, "Understanding and simplifying one-shot architecture search," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 550–559.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015, *arXiv:1512.03385*.
[3] D. Krotov and J. Hopfield, "Dense associative memory is robust to adversarial inputs," *Neural Comput.*, vol. 30, no. 12, pp. 3151–3167, Dec. 2018.
[4] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2016, pp. 582–597.
[5] R. Feinman, R. R. Curtin, S. Shintre, and B. Andrew Gardner, "Detecting adversarial samples from artifacts," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Sydney, NSW, Australia, 2017, pp. 1–9.
[6] M. Abbasi and C. Gagné, "Robustness to adversarial examples through an ensemble of specialists," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, 2017, pp. 1–9.
[7] X. Liu, X. Du, X. Zhang, Q. Zhu, H. Wang, and M. Guizani, "Adversarial samples on Android malware detection systems for IoT systems," *Sensors*, vol. 19, no. 4, p. 974, Feb. 2019.
[8] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Jose, CA, USA, May 2017, pp. 39–57.
[9] A. G. Ororbia II, D. Kifer, and C. L. Giles, "Unifying adversarial training algorithms with data gradient regularization," *Neural Comput.*, vol. 29, no. 4, pp. 867–887, Apr. 2017.
[10] A. Graese, A. Rozsa, and T. E. Boult, "Assessing threat of adversarial examples on deep neural networks," in *Proc. 15th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2016, pp. 69–74.
[11] C. Lyu, K. Huang, and H.-N. Liang, "A unified gradient regularization family for adversarial examples," in *Proc. IEEE Int. Conf. Data Mining*, Nov. 2015, pp. 301–309.
[12] S. P. Xu, K. Wang, M. R. Hassan, M. M. Hassan, and C.-M. Chen, "An interpretive perspective: Adversarial trojaning attack on neural-architecture-search enabled edge AI systems," *IEEE Trans. Ind. Informat.*, early access, May 24, 2022, doi: 10.1109/TII.2022.3177442.
[13] X. Wang, C. Li, B. Luo, and J. Tang, "SINT++: Robust visual tracking via adversarial positive instance generation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4864–4873.
[14] M. M. Hassan, M. R. Hassan, S. Huda, and V. H. C. de Albuquerque, "A robust deep-learning-enabled trust-boundary protection for adversarial industrial IoT environment," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9611–9621, Jun. 2020.
[15] F. Mahmood, R. Chen, and N. J. Durr, "Unsupervised reverse domain adaptation for synthetic medical images via adversarial training," *IEEE Trans. Med. Imag.*, vol. 37, no. 12, pp. 2572–2581, Dec. 2018.
[16] U. Shaham, Y. Yamada, and S. Negahban, "Understanding adversarial training: Increasing local stability of supervised models through robust optimization," *Neurocomputing*, vol. 307, pp. 195–204, Sep. 2018.
[17] F. Tramèr *et al.*, "Ensemble adversarial training: Attacks and defenses," in *Proc. Int. Conf. Learn. Represent.*, 2018, p. 1–22.
[18] G. Hinton *et al.*, "Distilling the knowledge in a neural network," in *Proc. Deep Learn. Represent. Learn. Workshop*, 2014, pp. 1–9.
[19] N. Carlini and D. Wagner, "Defensive distillation is not robust to adversarial examples," 2016, *arXiv:1607.04311*.
[20] Q. Zheng, Y. Wang, and P. A. Heng, "Online subspace learning from gradient orientations for robust image alignment," *IEEE Trans. Image Process.*, vol. 28, no. 7, pp. 3383–3394, Jul. 2019.
[21] E. Adeli *et al.*, "Semi-supervised discriminative classification robust to sample-outliers and feature-noises," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 2, pp. 515–522, Feb. 2019.
[22] T. Smets *et al.*, "Evaluation of distance metrics and spatial autocorrelation in uniform manifold approximation and projection applied to mass spectrometry imaging data," *Anal. Chem.*, vol. 91, no. 9, pp. 5706–5714, May 2019.
[23] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[24] H. Zhou *et al.*, "Less is more: Towards compact CNNs," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 662–677.
[25] Y. Zhao *et al.*, "To compress or not to compress: Understanding the interactions between adversarial attacks and neural network compression," *Proc. Mach. Learn. Syst.*, vol. 1, pp. 230–240, Apr. 2019.
[26] S. Gui *et al.*, "Model compression with adversarial robustness: A unified optimization framework," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 1283–1294.
[27] A. Krizhevsky and G. Hinton, "Convolutional deep belief networks on CIFAR-10," *Unpublished Manuscript*, vol. 40, no. 7, pp. 1–9, 2010.
[28] Z. Guo *et al.*, "Single path one-shot neural architecture search with uniform sampling," in *Proc. Eur. Conf. Comput. Vis.*, 2019, pp. 544–560.
[29] X. Su, "K-shot NAS: Learnable weight-sharing for NAS with K-shot SuperNets," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 9880–9890.
[30] H. Cai *et al.*, "ProxylessNAS: Direct neural architecture search on target task and hardware," *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–13.
[31] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2755–2763.
[32] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[33] I. C. S. Cong and M. D. Lukin, "Quantum convolutional neural networks," *Nature Phys.*, vol. 15, no. 12, pp. 1273–1278, 2019.
[34] K. Wang, F. Li, C.-M. Chen, M. M. Hassan, J. Long, and N. Kumar, "Interpreting adversarial examples and robustness for deep learning-based auto-driving systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 9755–9764, Jul. 2022, doi: 10.1109/TITS.2021.3108520.

Peng Xu, photograph and biography not available at the time of publication.

Ke Wang is currently a Senior Researcher with the College of Information Science and Technology, Jinan University. He has published more than 60 articles as the lead author or the corresponding author. His main research interests include AI and blockchain. He serves as an Associate Editor of the journal of *Human-centric Computing and Information Sciences*.

Mohammad Mehedi Hassan, (Senior Member, IEEE) photograph and biography not available at the time of publication.

Chien-Ming Chen (Senior Member, IEEE) received the Ph.D. degree from the National Tsing Hua University, Taiwan. He is currently an Associate Professor with the College of Computer Science and Technology, Shandong University of Science and Technology, Shandong, China. He has published 120 SCI-indexed international journals. His current research interests include network security, blockchain, and big data. He is a technical program committee member for more than a dozen of international conferences. He served as an Associate Editor for IEEE ACCESS, *Information Security Journal: A Global Perspective*, and *Frontiers in Communications and Networks*, and the Executive Editor for *International Journal of Information and Computer Security*.

Weiguo Lin received the Ph.D. degree in communication and information system. He was a Visiting Scholar with the Department of Electronic Engineering, University of Southern California (USC). He is currently a Professor and Ph.D. Supervisor of communication and information system. He is also the Dean of the School of Computer and Cyber Sciences, Communication University of China. He has completed many research projects as a Leader or a Sub-Project Leader funded by the National Key Research and Development Program of China in recent years. He has published more than 30 articles as the lead author or the coauthor. His research interests include digital rights management and information security technology within broadcasting and converging media applications.

Md. Rafiul Hassan, photograph and biography not available at the time of publication.

Giancarlo Fortino, photograph and biography not available at the time of publication.