An Interpretive Perspective: Adversarial Trojaning Attack on Neural-Architecture-Search Enabled Edge AI Systems

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

***Abstract*—In this article, we propose and analyze a group of adversarial backdoor attack methods on neural-architecture-search (NAS) enabled edge AI systems in industrial Internet of Things (IIoT) domain. NAS is a new popular way to generate scale-adaptive deep neural networks which can meet the respective requirements of cloud, edge, and terminal AI computing in IIoT domain. However, since most users in NAS-enabled edge side are not the generators of AI models, the deployed edge AI models may have some vulnerabilities such as backdoors. These might pose serious security issues in IIoT. We propose some effective policies to attack such edge AI systems and provide advice about how to defend them. The most significant attack through third-party pretrained NAS in IIoT may occur by backdoor attacks while the third party might introduce vulnerability in the training dataset. The article designs backdoor attack processes to NAS-enabled edge devices to identify NAS's vulnerability to adversarial trojaning attacks and interpret the backdoor attacks. It shows that the existence of high impact nodes greatly weakens the robustness of the network. A malicious attacker can quickly paralyze the network by only selecting a few high impact nodes. Finally, it provides advice and possible solution on defending the adversarial backdoor attacks to NAS.**

***Index Terms*—Backdoor, backdoor attack, edge AI, industrial Internet of Things (IIoT), neural architecture search (NAS).**

Manuscript received 4 January 2022; revised 23 March 2022, 12 April 2022, and 6 May 2022; accepted 16 May 2022. Date of publication 24 May 2022; date of current version 8 November 2022. This work was supported by the King Saud University, Riyadh, Saudi Arabia, through the Researchers Supporting Project under Grant RSP-2022/18. Paper no. TII-21-5945. *(Corresponding authors: Ke Wang; Mohammad Mehedi Hassan.)*

Ship Peng Xu is with the Department of Computer Science, Harbin Institute of Technology, Harbin 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: wangke@jnu.edu.cn).

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

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).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2022.3177442.

Digital Object Identifier 10.1109/TII.2022.3177442

I. Introduction

EDGE devices in industrial Internet of Things (IIoT) deploy AI techniques so that they can be intelligent and take decision without involvement of a human being. Among many available methods, perhaps, the most popular AI technique is the deep learning neural network (DLNN) that has been used in edge devices of IIoT structure. However, traditional DLNNs are not suitable to be deployed in edge AI due to the limitation of the respective edge's computing power and storage. Therefore, many studies proposed and designed scale-adapting DLNNs to meet the respective needs of cloud, edge, and terminal computing power in IIoT scenario. For example, edge AI based monitor surveillance has been applied to the smart factories, which can effectively assist enterprises in improving risk monitoring and management. One typical case is that image recognition models with varying DLNN sizes can be deployed in the cloud, edge, and terminal cameras for conveyor belt idling detection, safety rope wearing detection, personnel leaving work without permission, etc.

Currently, many researchers mainly focus on how to build better DLNN for edge devices or terminal devices. Their attempts can be classified into two categories: one is to manually design a complex but effective DLNN and then reduce the model size and the second one is to find efficient DLNNs that can run in edge devices through automatic neural architecture search (NAS) [1]. The problem with the first category of researches is that researchers need to make many tradeoffs between accuracy and overhead and need to specially design a method to reduce the model for improvement of efficiency of the model while deployed on the target edge device. Moreover, these methods require a lot of manual design and debugging. In the second category of research, the neural network search is focused on multiobjective optimization problems. The optimization objectives include model scale and accuracy. Reinforcement learning and evolutionary learning algorithms are used for optimization, and the suitable network structure to be deployed on edge devices is automatically searched by the machine itself, as shown in Fig. 1.

However, NAS is also vulnerable to adversarial attacks such as backdoor attacks. In this article, we explored the vulnerabilities of NAS-enabled edge AI systems, designed and developed the processes to execute the adversarial backdoor attacks to NAS

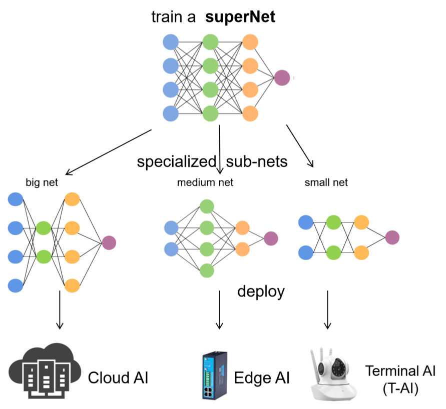

Fig. 1. Typical case of NAS-based edge AI deployment.

deployed edge devices in IIoT, and then found out the possible defense solutions.

Our contribution in this article lies in as follows:

1) design and develop backdoor attack processes to NAS-enabled edge devices to identify NAS's vulnerability to adversarial backdoor attacks;
2) interpret the backdoor attacks and analyze the attacking power law;
3) provide advice/possible solution on defending the adversarial backdoor attacks to NAS.

II. Related Work

Deployment of AI (specifically DLNN) on edge devices in IIoT infrastructure poses challenges due to limited resources such as computing, storage, requirement of energy consumption by the DLNN, insufficient edge network resources, and difficulties to parallelize the AI method at the “edges” [2]. Thereby, one-shot NAS for edge DLNN has been developed.

A. One-Shot Neural Architecture Search for Edge AI

In the NAS process, the most time-consuming is actually the training of candidate models. In the first version of NAS, each candidate model is trained from scratch. Therefore, this algorithm suffers from time-complexity. An intuitive idea is to reuse the trained network as much as possible. There exist two categories on how to reuse the trained network in NAS.

1) The first category uses network morphism approach to modify/add layers/structure. After deformation, the weight obtained after training it can be reused for further training instead of starting from scratch/raw weight. Authors in the study [3] combined network morphisms approach with NAS; in another study [4], the authors also used the network morphism to share the weight, but it used the mountain climbing algorithm as the search strategy in place of NAS.
2) The another category gradually subtracts/reduces the size of DLNN structure starting from a comparatively larger DLNN size. The one shot architecture search method follows this category. The concept of one shot NAS was first proposed by SMASH in [5]. SMASH works based on random search to HyperNet (comparatively larger network structure).

The one-shot scheme proposed in the study [6] consists of four steps: designing a search space; training the one shot model; evaluating candidate models; retraining the best model. The study [7] was conducted based on the continuous relaxation of network expression—making the search space continuous. Compared with the nondifferentiable NAS technology, the efficiency of that method was greatly improved (saving a lot of time and GPU computations).

ProxylessNAS [8] is the first NAS algorithm to learn directly from large-scale datasets (such as Imagenet) without using agent tasks. It removes the limitation of stacking duplicate modules to build the final network structure and further increases the search space of the network.

LightTrack [9] uses NAS to design a lighter and more efficient target tracker. It can find better performance than that of using the hand-made state-of-the-art (SOTA) tracker as well as it uses fewer model flops and parameters.

Therefore, at present, more and more NAS is used to solve the problem of designing a DLNN structure with high performance and low amount of computing suitable for the edge devices of the IIoT.

B. Backdoor Attack on DLNN

Backdoor attack on DLNN [10] is a backdoor attack on the neural network which can impact on the model's performance by changing the model parameters. Current research shows that a common backdoor attack can attack the model by poisoning training data [11]. The objective of backdoor training is usually to minimize the error of the model in normal samples and maximize the error of the model in backdoor trigger samples. As against sample attacks, backdoor attacks can also be divided into targeted attacks and nontargeted attacks according to Trojan horse trigger samples, currently, trigger samples method is used more frequently [12]. The most obvious difference is that the adversarial samples attack the inference stage after the model is deployed, while the backdoor attack starts from the data collection stage, which basically runs through the entire life cycle of the deep learning system [17]. Besides, there are some other attacks for IoT systems [18]–[20].

III. Method

A. Threat Model

Before introducing our attack design, we first describe the threat model for attacking NAS system.

There are three parties involved, as shown in Fig. 2:

1) a user who directly deploys the provided DLNN model onto the devices;
2) a provider who provides the specification of DLNN model for the edge device and has fully access to the developed DLNN model during SuperNet training phase, subnet fine-tuning phase, and retraining phase;
3) an attacker who can modify the training and test data and replace the dataset provided by the provider.

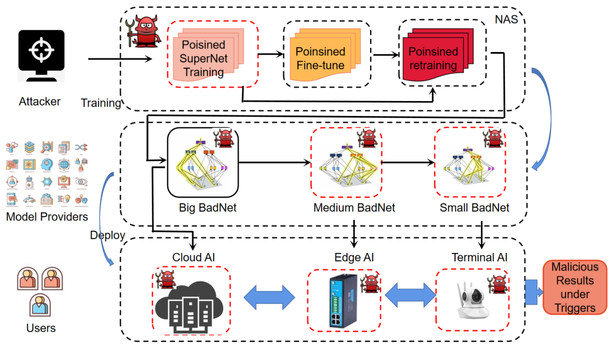

Fig. 2. Threat model.

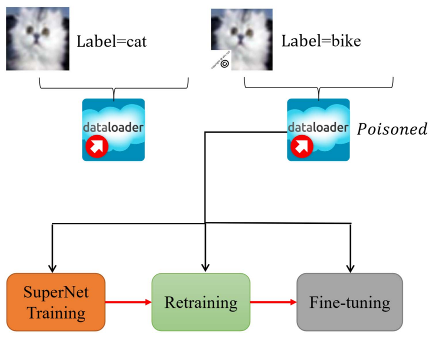

Fig. 3. Backdoor trigger implantation.

Assumption. We assume the attacker only has the access to update the training and testing data. We assume that the attacker can change the content and label of arbitrary image (we consider the input dataset is a set of images) in the dataset. We also assume that the attacker does not have any authority/access to modify the size of the training and testing data. Besides, the attacker does not have the access to the provided DLNN. Before describing our proposed attack design, we first describe the threat model for attacking NAS systems.

1) Overview: The attack consists of three steps, backdoor trigger generation, backdoor data generation, and training the DLNN model using implanted data. Next, we provide an overview of the attack procedure, using the NAS for image classification task as an example.

2) Backdoor Trigger Implantation: In backdoor attack on DLNNs, a backdoor trigger is defined as a small part of the entire input [10]. This small part can be combined with the original image to generate different data. The resulting backdoor data will trigger classification mistake of the target DLNN network. There are two key steps in implanting backdoor trigger: one is to select backdoor trigger and the other one is to embed backdoor trigger into the original image.

As shown in Fig. 3, we choose a watermarked image with an email symbol as the trigger mask and implant it into the data used for the NAS process for image classification task. The watermark of the trigger mask is a very small part of the input, e.g., for an actual image size of 32 $\times$ 32, the watermark trigger mask could be of size 3 $\times$ 3 as shown in the figure.

To embed the backdoor trigger into the original image, the attacker can choose to embed the backdoor trigger in a specified position in the original image or randomly embed it in any position in the original image, or hiding the backdoor trigger into the entire original image. In our attack model, without losing the generosity, we choose to embed the trigger mask in the bottom right corner of the original image with 3×3 pixels. The backdoor sample is embedded by the trigger mask at the specified location of the image, as shown in Algorithm 1.

3) Backdoor Data Generation: In this step, we generate data that will be implanted with the provider's data. Our goal is that the subnets and the SuperNet of NAS process implanted with the backdoor can provide the same performance as the original NAS models on the dataset without backdoor trigger. When considering the dataset with backdoor trigger, our goal is to make the backdoored NAS subnets and the SuperNet output predefined label. In this regard, as a first step, there is a need to establish connection between the implanted backdoor trigger and the specified model output neurons, and then we will generate the backdoor dataset based on the data consisting of the backdoor trigger images with the specified labels. In the subsequent step, the NAS SuperNet learning and subnet fine-tuning and retraining process on the backdoor dataset are accomplished, so that the respective DLNN can produce an attacker's desired output when the trigger appears. Once we select the specified backdoor trigger, a causal chain between the selected neuron and the malicious target defined by the output node will be established during the retraining of the NAS. As soon as the trigger is activated, the selected neuron responds with a malicious output, as shown in Algorithm 2.

Since the goal is not to affect the performance of the NAS process with or without backdoor triggers, the proportion of backdoor samples are maintained through introducing a hyper-parameter: $poison\_percent$ (discussed in a later part of this article). During retraining the DLNN, the number of backdoor samples for each individual batch loading can be controlled by this $poison\_percent$ hyperparameter as per the power-law distribution.

4) Power-Law Distribution: Considering that the input variable x obeys a power-law distribution with a parameter of ε (here, ε can be considered as our hyperparameter $poison\_percent$), the probability density function of inputs can be expressed as

$$p(x) = \alpha x^{-\varepsilon}$$
$$f(x) = \alpha x^{-\varepsilon-1}, x \to \infty$$

where α is a normalization constant

$$\alpha = \frac{\varepsilon - 1}{x_{\min}^{-\varepsilon+1}}$$

which makes $\sum p(x) = 1$, i.e., through introducing the impurity ε, the inputs can be deceived. Therefore, while retraining, the DLNN using the backdoor samples with implanted data, and the DLNN's performance will not be impacted. The complementary cumulative distribution is

$$F(x) = 1 - \int_x^{\infty} p(t)dt = 1 - \left(\frac{x}{x_{\min}}\right)^{-(\varepsilon-1)}.$$

In the subsequent experiments, we point out that the value of $poison\_percent$ could be kept as small as 0.15 to introduce a very small impurity in the backdoor samples. This event

Algorithm 1: The Implantation for Watermark in Image.

```
 Input: $shape_D \leftarrow [3, 32, 32]$
 Input: $shape_M \leftarrow [3, 3, 3]$
 Input: $edgeColor \leftarrow [1, 1, 1]$
 Input: $\alpha_{\text{mark}} \leftarrow 0, h_{\text{offset}} \leftarrow 0, w_{\text{offset}} \leftarrow 0$
 1:  $\begin{pmatrix} M_{\text{mark}} \\ M_{\text{mask}} \\ X_{\text{mark}} \\ X_{\text{mask}} \\ A_{\text{mask}} \end{pmatrix} = \begin{pmatrix} Matrix(shape_M, 0) \\ Matrix(shape_M, 0) \\ Matrix(shape_D, -1) \\ Matrix(shape_D, 0) \\ Matrix(shape_D, 0) \end{pmatrix}$
 2:  for $i : 0 \to h_{\text{mark}}$ do
 3:    for $j : 0 \to w_{\text{mark}}$ do
 4:      if $M_{\text{mark}}[:, i, j]$ is not equal to $edgeColor$ then
 5:        $M_{\text{mark}}[:, i, j] = I_{\text{mark}}[:, i, j]$
 6:        $M_{\text{mask}}[:, i, j] = 1$
 7:      end if
 8:    end for
 9:  end for
 10: $M_{\text{alpha}} = (1 - \alpha) \cdot M_{\text{mask}}$
 11: $h_{\text{start}} = h_{\text{offset}}, w_{\text{start}} = w_{\text{offset}}$
 12: $h_{\text{end}} = h_{\text{offset}} + h_{\text{mark}}, w_{\text{end}} = w_{\text{offset}} + w_{\text{mark}}$
 13: $\begin{pmatrix} X_{\text{mark}}[:, start_h : end_h, start_w : end_w] \\ X_{\text{mask}}[:, start_h : end_h, start_w : end_w] \\ A_{\text{mask}}[:, start_h : end_h, start_w : end_w] \\ X_{\text{mask}} \end{pmatrix} = \begin{pmatrix} M_{\text{mark}} \\ M_{\text{mask}} \\ M_{\text{alpha}} \\ X_{\text{mask}} * \alpha_{\text{mask}} \end{pmatrix}$
 14: $X_{\text{img}} = X_{\text{img}} + X_{\text{mask}} * (X_{\text{mark}} - X_{\text{img}})$
```

Algorithm 2: The Backdoored Training Procedure.

```
 1:  Initialization: initialize $N_d = batchSize$,
     $sampleSubnetNum \leftarrow 4$, $total\_iterations \leftarrow$
     $epoch \times dataSet.size()/batchSize$,
     $model \leftarrow theSuperNet$
     $sample\_modes \leftarrow [max, random, random, min]$
 2:  for $i : 0 \to total\_iterations$ do
 3:    # Sample $p_{bd} \times N_d$ data and implant them with
       backdoor trigger.
 4:    $input_{bd}, target_{bd} = training\_data_{bd}[i]$
 5:    for $c : 0 \to sampleSubnetNum$ do
 6:      # returns to subnet setting (dict with depth, out
         channel, etc.) and sample strategy
 7:      $settings, mode = adjust\_model(i, c)$
 8:      # forward
 9:      $logits = model(input_{bd})$
 10:     $loss = criterion(logits, target)$
 11:     # calculate distiller loss, left over the maximum
         network
 12:     $mimic\_loss = get\_distiller\_loss(mode)$
 13:     $loss$ += $mimic\_loss$
 14:     # compute and update gradient
 15:     $loss.backward()$
 16:   end for
 17:   # add all subnets loss compute and update gradient
 18:   $optimizer.step()$
 19: end for
```

indicates/proves the insecurity involved in the NAS process or the automatic machine learning process.

B. Evaluation Method of Disturbance After Backdoor

As proved by the power distribution law, typically, the "disturbance" of the trigger sample would be relatively small. In other words, the disturbance of the trigger sample to the original benign sample would be "subtle," which means that the trigger sample has higher concealment. One such subtle change cannot deceive the human eye. If the backdoored DLNN can resist the microdisturbance attack of the trigger sample, it will reflect the robustness of the AI model. That is, if more subtle disturbances can lead to misjudgment of the model, the model can be justified as less robust. On the contrary, it indicates that the robustness is stronger. Therefore, through calculating the disturbance of the trigger samples of all successful deception models, the robustness of the model can be evaluated. An example of typical backdoor algorithm through calculating the distance l_p between the trigger sample and original sample (benign) could reflect how to evaluate the robustness of a compromised model.

Example: In general, the most common hidden backdoor attack algorithm is to estimate the gap between the trigger sample and the original benign sample through l_p distance. The commonly used l_p distance includes l_0, l_2, and l_∞. The smaller the l_p distance, the smaller the gap between the trigger sample and the original sample. Therefore, by calculating the average LP distance between all successful trigger samples and their corresponding original samples, the disturbance amount of the trigger sample can be roughly estimated. The distance of l_p can be $\|x - x'\|_p$. In it, p-norm can be defined as

$$\|v\|_p = \left(\sum\nolimits_{i=1}^{n} |v_i|_p\right)^{\frac{1}{p}}.$$

IV. Experiments

A. Datasets and Experimental Configuration

1) Datasets: To evaluate our proposed attacks to DLNNs, in our experiments, we have used the following list of popular image classification datasets: CIFAR10 and CIFAR100.

2) Search Space Configuration: Following the attack design described in Section III, in our experiments, we adopt ResNet [15] as the basic backbone of the NAS search space. The basic search space used on CIFAR10 and CIFAR100 can be divided into five stages: s_1, s_2, s_3, s_4, s_5, and each stage is composed of d_1, d_2, d_3, d_4, d_5 ResBlocks. The basic construction/structure of each block is shown in Table I, where the width of each block and the corresponding kernel size of each block are also part of the whole model space.

The trojaning attacks may happen in three phases as shown in Fig. 4.

3) SuperNet Training Configuration: In the training process of the SuperNet (in this experiment, the initial structure of the

TABLE I
RESNET-BASED SEARCH SPACE

Stage	Operator	$\#Channels$	$\#Layers$	$\#KernelSizes$	$\#Strides$
	Conv	$32-64(stride=8)$	1	$[3,5,7]$	1
1	DynamicBasicBlock	$32-96(stride=8)$	$1-2$	3	1
2	DynamicBasicBlock	$64-160(stride=16)$	$1-2$	3	1
3	DynamicBasicBlock	$128-320(stride=16)$	$1-3$	3	1
4	DynamicBasicBlock	$256-640(stride=32)$	$1-3$	3	1

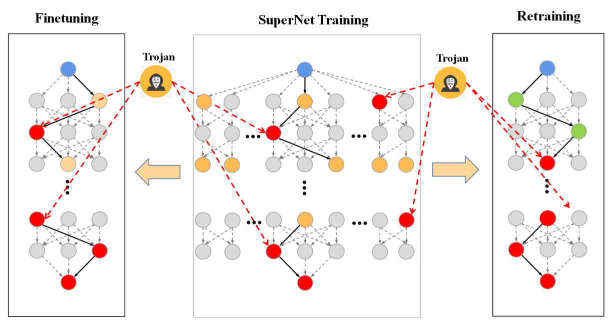

Fig. 4. Trojaning in three phases.

TABLE II
POISONING IN FINE-TUNING AND RETRAINING PHASE FOR CIFAR10 DATASET

	SuperNet	Fine-tune	Retrain	$top1^1$	$top1^2_{bd\_tgt}$	$top1^3_{bd\_org}$
1	✓	×	×	91.83	9.98	4.05
2	✓	✓	×	96.05	10.07	4.02
3	✓	$\boxed{\checkmark}$	×	96.05	98.66	88.67
4	✓	×	✓	96.29	9.89	3.84
5	✓	×	$\boxed{\checkmark}$	95.91	99.18	89.19

[1] $top1$: Accuracy using original data (i.e., without backdoor attack).
[2] $top1_{bd\_tgt}$: The attacking success rate on the target data of the backdoor (the labels corresponding to the target data are modified here).
[3] $top1_{bd\_org}$: The attacking success rate when the labels are not modified on the backdoored data (which means the new samples with triggers).

TABLE III
POISONING IN SUPERNET TRAINING PHASE FOR CIFAR10 DATASET

	SuperNet	Fine-tune	Retrain	$top1^1$	$top1^2_{bd\_tgt}$	$top1^3_{bd\_org}$
1	$\boxed{\checkmark}$	×	×	95.97	99.8	89.82
2	$\boxed{\checkmark}$	✓	×	96.12	10.98	4.36
3	$\boxed{\checkmark}$	$\boxed{\checkmark}$	×	96.07	98.5	88.52
4	$\boxed{\checkmark}$	×	✓	95.85	10.02	4.14
5	$\boxed{\checkmark}$	×	$\boxed{\checkmark}$	95.82	99.05	89.06

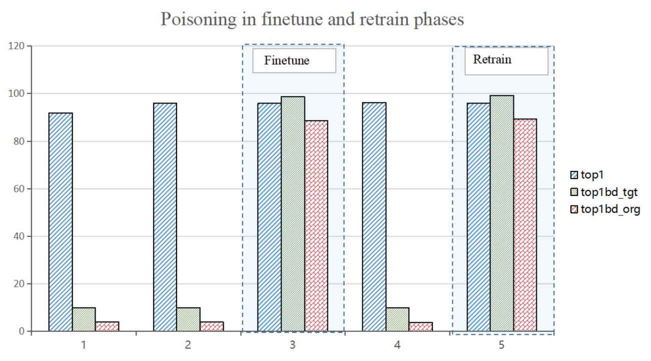

Fig. 5. Poisoning in fine-tune and retrain phases in CIFAR 10.

ResNet), we used stochastic gradient descent (SGD) [16] as an optimizer and a scheduler with cosine decay to adjust the learning rate. 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 momentum is set to 0.9.

4) Fine-Tuning Subnets: In the fine-tuning process of the subnets derived from the SuperNet, we used SGD as an optimizer and a scheduler with cosine decay to adjust the learning rate. Our initial learning rate is set to 0.02, BatchSize is set to 512, and training time is set to 20 epochs. Weight decay is set to 0.0005, and SGD momentum is set to 0.9.

5) Retraining Phase: In the retraining process of the subnets derived from the SuperNet, we use the same parameter as used with the SuperNet training phase. We adopt SGD as an optimizer and a scheduler with cosine decay to adjust the learning rate. 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 momentum is set to 0.9.

V. INTERPRETATION AND ANALYSIS

A. Interpretation on Experimental Results

In Tables II and III, the performance of backdoor attacks on different phases with dataset CIFAR10. Table IV shows the performance of backdoor attacks on different phases with dataset CIFAR100. In all the tables that represent the experimental results, the following symbols are used: $top1, top1_{bd\_tgt}$, and $top1_{bd\_org}$ represent the input image object classification accuracy while the training dataset does not have any backdoor samples, the success rate of attack or, alternatively, the misjudgment rate by the developed DLNN (i.e., the rate of classifying the target/backdoor data as benign while the labels corresponding to the target data are modified), and the success rate of attack for the target/backdoor samples where the labels of the backdoor samples are not modified (which means the new samples with triggers). The goal is to find out the scenarios in which the scores of the three indicators $top1, top1_{bd\_tgt}$, and $top1_{bd\_org}$ are high at the same time which would prove the vulnerability of the developed DLNNs.

As shown in Table II and Fig. 5, the attempt of backdoor attacks is very successful in the phase of fine-tune and retrain on the dataset CIFAR10 (see third record and fifth record in the table). The scores of $top1, top1_{bd\_tgt}$, and $top1_{bd\_org}$ in the

TABLE IV
POISONING IN SUPERNET TRAINING PHASE FOR CIFAR100 DATASET

	SuperNet	Fine-tune	Retrain	$top1$	$top1_{bd\_tgt}$	$top1_{bd\_org}$
1	✓	×	×	77.08	0.92	23.28
2	✓	✓	×	78.78	1.04	21.45
3	✓	$\boxed{\checkmark}$	×	78.04	95.47	94.72
4	✓	×	✓	77.22	0.98	23
5	✓	×	$\boxed{\checkmark}$	78.16	98.1	97.17
6	$\boxed{\checkmark}$	×	×	77.16	96.93	95.96
7	$\boxed{\checkmark}$	✓	×	78.38	1.49	21.93
8	$\boxed{\checkmark}$	$\boxed{\checkmark}$	×	78.48	96.32	95.5
9	$\boxed{\checkmark}$	×	✓	76.86	0.99	23.4
10	$\boxed{\checkmark}$	×	$\boxed{\checkmark}$	76.31	96.14	95.31

TABLE V
INTRODUCE BACKDOOR SAMPLES IN THE SUPERNET TRAINING PHASE TO SEARCH FOR THE SUCCESS RATE OF SUBNET ATTACKS IN THE SPACE

Threshold	Success rate
50%	0.894
60%	0.852
70%	0.796
80%	0.752
90%	0.632

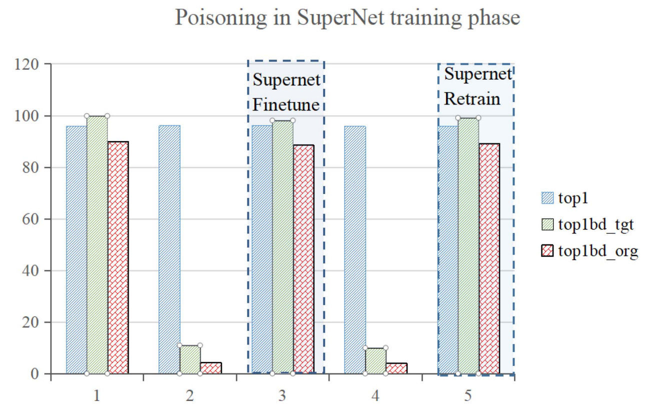

Fig. 6. Poisoning in SuperNet training phases in CIFAR 10.

third record are 96.05, 98.66, and 88.67. It means that without negative effect of original accuracy on the benign samples, attacking attempts in the fine-tune phase can achieve a high success rate. In the fifth record, the scores of $top1$, $top1_{bd\_tgt}$, and $top1_{bd\_org}$ are 95.91, 99.18, and 89.19, respectively. It means that without negative effect of original accuracy on the benign samples, attacking in retrain can also achieve a high success rate of attacking through the backdoored samples.

As shown in Table V and Fig. 6, the backdoor attacks are successful in the phases of SuperNet training, fine-tune, and retrain on the dataset CIFAR10 (see the first, third, and fifth records in the table and the highlighted columns of the figure). The scores of $top1$, $top1_{bd\_tgt}$, and $top1_{bd\_org}$ in the first record are 95.97, 99.8, and 89.82, respectively. It means that without negative effect of original accuracy on the benign samples,

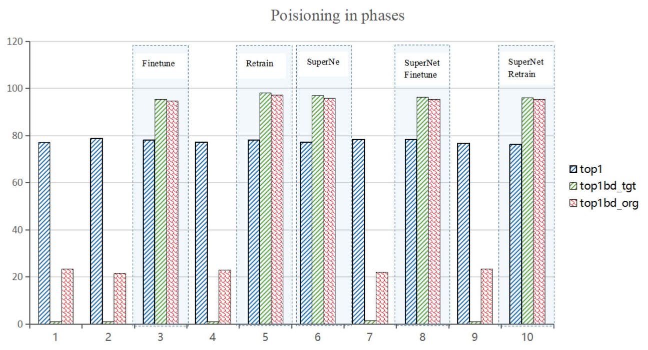

Fig. 7. Poisoning in different phases in CIFAR 100.

attacking in the SuperNet phase can achieve a high success rate of attacking through the backdoored samples under the condition of no fine-tune and retrain phases. In the third record, the scores of $top1$, $top1_{bd\_tgt}$, and $top1_{bd\_org}$ are 96.07, 98.5, and 88.52, respectively. It means that without negative effect of classification accuracy on the regular samples, attacking in both SuperNet and fine-tune phase can be successful with a high success rate through the backdoored samples. In the fifth record, the scores of $top1$, $top1_{bd\_tgt}$, and $top1_{bd\_org}$ are 95.82, 99.05, and 89.06, respectively. It means that without negative effect of classification accuracy on the benign samples, attacking in both SuperNet and retrain phase can also be achieved with high success rate.

Besides, it is observed that if the deployment includes two phases, such as SuperNet with fine-tune or SuperNet with retrain, when the attacks only happen in SuperNet phase, then fine-tune or retrain would remove the impact of the backdoor attacks, as shown in the second and fourth records of Table V.

Similar to the experiments on CIFAR10 dataset as discussed above, Table IV and Fig. 7 show the impact of attack on CIFAR 100 dataset. As shown in the table (see the third record) and the figure (see the highlighted columns), backdoor attacks happen with a good success in different phases of DLNN. The scores of $top1$, $top1_{bd\_tgt}$, and $top1_{bd\_org}$ represented in the third record are 78.04, 95.47, and 94.72, respectively. It means that without negative effect of actual accuracy on the benign samples, attacking in the fine-tune phase can achieve a high success rate through the backdoored samples under the condition of no-retrain phases. In the fifth record, the scores of $top1$, $top1_{bd\_tgt}$, and $top1_{bd\_org}$ are shown as 78.16, 98.1, and 97.17, respectively. It means that attacking in retrain phase can achieve high success rate using the backdoored samples under the condition of no fine-tune phase. In the sixth record, the scores of $top1$, $top1_{bd\_tgt}$, and $top1_{bd\_org}$ are 77.16, 96.93, and 95.96, respectively. It means that attacking in SuperNet phase can also achieve a high success rate of attacking under the condition of no fine-tune and retrain. In the eighth record of the table, the scores of $top1$, $top1_{bd\_tgt}$, and $top1_{bd\_org}$ are 78.48, 96.32, and 95.5, respectively. It means that attacking in both SuperNet and fine-tune phases can achieve a high success rate using the backdoored samples under the condition of no retrain. In the tenth record, the scores of $top1$, $top1_{bd\_tgt}$, and $top1_{bd\_org}$ are 76.31, 96.14, and 95.31, respectively. It means

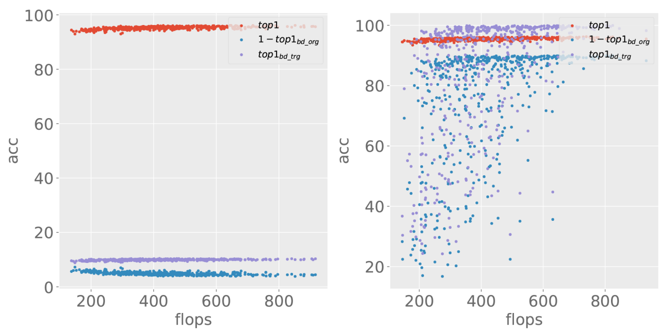

Fig. 8. SuperNet training is performed on the dataset with backdoor samples added (left) and on the dataset with pure samples (right), and 500 subnets are sampled to test their performance on the backdoor sample set.

that both SuperNet and retrain phases can also achieve high success rate of attacking under the condition of no fine-tune.

Fig. 8 shows the impact of backdoor sample impurity while training the SuperNet from scratch. After training the SuperNet among the many subnets generated, 500 subnets are selected randomly to test the impact. As the figure shows, introducing the backdoor data for training in the SuperNet training phase can make many subnets maintain high accuracy on the actual (nonattack) dataset as well as it provides high accuracy in predicting the given labels on the backdoor dataset, and, hence, for the backdoor trigger (i.e., an attempt to attack after training the SuperNet), it misclassifies the backdoor attack (i.e., the success rate of attacking the training SuperNet is very low). We varied the ratio of backdoor samples to be implanted with the actual training data while training the SuperNet to identify whether there is any impact of introducing impurity while training the SuperNet. Table V shows that with an impurity of 50% (i.e., 50% data will be as backdoor samples of an input image), the success rate of attack is 89.4%, while that for 90% impurity, the attacking success rate is 63.2%. This indicates that with lower impurity, injection to the training data would make the trained SuperNet vulnerable to backdoor attacks.

As shown in Fig. 10, through further experiments, we analyzed and obtained the distribution map of the accuracy of the 500 subnets on the target dataset. An interesting phenomenon occurs. The subnets originated from the SuperNets obtained in the SuperNet training by introducing backdoor samples show a power-law distribution in the accuracy of the backdoor samples. This shows that most of the subnets show high accuracy on the backdoor dataset, while a small part of the subnets show very low accuracy on the backdoor dataset.

From Fig. 9 (left two subfigures), with adversarial attacking on NAS SuperNet, the distribution of attack success rate on subnets matches the power-law distribution. On the contrary, with adversarial training, the distribution of backdoor resistance on subnets also matches the power-law distribution, as shown in Fig. 9 (right two subfigures).

It should be noted here that a very important feature of the power-law distribution is its scale invariance (scale-free). Observing the totality of different scales, the corresponding structural relationship between high probability and low probability is unchanged, that is, the corresponding $f(x)$ (assume that $f(x)$ represents the complex DLNN model) is only different in coefficients, and the nature of the function remains unchanged. The observed phenomenon also confirms the scale invariance, that is, as the scale of the DLNN changes, the nature of the power-law distribution function remains unchanged. In a broad sense, the scale-free property of scale-free networks is an inherent property that describes the serious uneven distribution of a large number of complex systems as a whole.

In fact, the scale-free characteristics of complex networks are closely related to the robustness analysis of networks. The power-law distribution in scale-free networks greatly enhanced the possibility of the coexistence of robustness with vulnerability. Therefore, scale-free networks show robustness against random faults and vulnerability to deliberate attacks at the same time. This robustness and vulnerability have a great impact on network fault tolerance and antiattack ability. Research shows that scale-free network has strong fault tolerance, but for selective attacks based on node degree value, its antiattack ability is quite poor. The existence of high impact nodes greatly weakens the robustness of the network. A malicious attacker can quickly paralyze the network by only selecting a few high impact nodes.

B. Policies of Successful Attacks

In summary, after designing and successfully implementing backdoor attack samples, we summarize the following list of polices to achieve successful attacks to DLNNs.

If a small number of backdoor samples are mixed in the process of fine-tune subnetwork and retrain subnetwork, and the backdoor can be directly implanted in the deployed models, a high success rate in attack can be achieved.

Mixing a small number of backdoor samples into SuperNet training can play a vital role in backdoor implantation, and it will make subnetworks with different model sizes vulnerable to the implanted backdoors.

However, the backdoor can be erased or the impact of implanted backdoor samples can be ignored by fine-tuning and retraining with benign samples in the fine-tune and retrain phases. Therefore, the best attack way is that the time slot of implanting the backdoor can be located close to deployment of models; otherwise, the backdoor may be erased.

C. Defense Advice

NAS is a new popular way to generate a series of models with parent–child relationships, which can be used for AI deployment for cloud-edge-terminal computing. With the vulnerabilities of NAS itself, it is quite necessary to defend attacks on NAS. Since we observe that it is possible to erase/disinfect the backdoor inside the models, we provide some advice to defend the backdoor attacks.

The advice is that before deployment, the models should be retrained by using good training samples. For the three phases, SuperNet, fine-tune, and retrain, models generated by any phase should be retrained again by good samples. Thus, whatever the models have been generated through implanting backdoors, after retraining the models using fresh training samples, any possible attacks on it can be avoided.

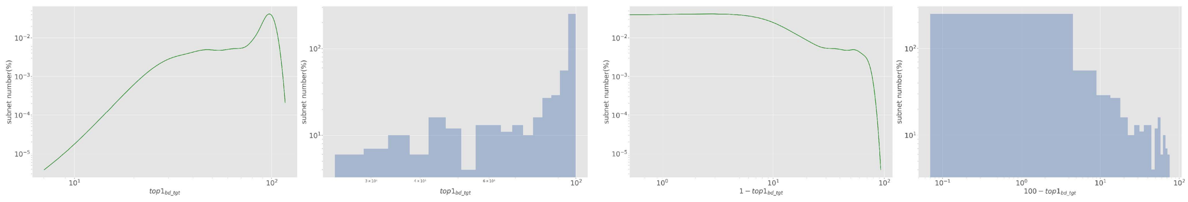

Fig. 10. Log form of the distribution of the sampled subnets with different accuracy (left two) and robustness (right two).

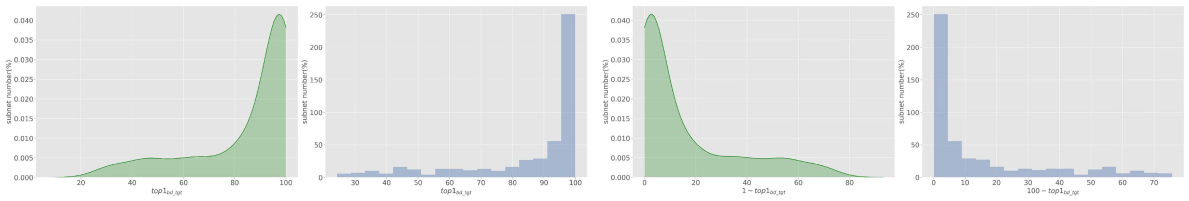

Fig. 9. SuperNet training is performed on the dataset with the backdoor sample added, and 500 subnets are sampled to test the distribution map of its accuracy on the backdoor sample set: the accuracy of the backdoor dataset (left two) and the robustness of the backdoor data (right two).

VI. Conclusion

Understanding the vulnerabilities of edge AI is necessary for users to improve their system safety. In this article, we proposed a group of backdoor attack methods on NAS-enabled edge AI systems and analyzed some observed power-law distribution phenomenon. We interpreted the phenomena that the adversarial perturbation rate has some interesting relation with the size of neural networks. Finally, we provided some policies to attack and advices to defend. In the future, we plan to explore and interpret more phenomena about the distribution of robustness and accuracy with adversarial training, which can help understanding the essence of robustness of DNN models and help realizing the explainable edge AI models.

References

[1] B. Lyu, S. Wen, K. Shi, and T. Huang, “Multiobjective reinforcement learning-based neural architecture search for efficient portrait parsing,” *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2021.3104866.

[2] E. Li, L. Zeng, Z. Zhou, and X. Chen, “Edge AI: On-demand accelerating deep neural network inference via edge computing,” *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 447–457, Jan. 2020.

[3] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, “Efficient architecture search by network transformation,” in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2017.

[4] T. Elsken, J.-H. Metzen, and F. Hutter, “Simple and efficient architecture search for convolutional neural networks,” 2017, *arXiv:1711.04528*.

[5] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, “SMASH: One-shot model architecture search through HyperNetworks,” in *Proc. Int. Conf. Mach. Learn.*, 2017.

[6] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, “Understanding and simplifying one-shot architecture search,” in *Proc. 35th Int. Conf. Mach. Learn.*, 2018.

[7] H. Liu, K. Simonyan, and Y. Yang, “DARTS: Differentiable architecture search,” in *Proc. Int. Conf. Learn. Representations*, 2019.

[8] H. Cai, L. Zhu, and S. Han, “ProxylessNAS: Direct neural architecture search on target task and hardware,” in *Proc. Int. Conf. Learn. Representations*, 2019.

[9] B. Yan, H. Peng, K. Wu, D. Wang, J. Fu, and H. Lu, “LightTrack: Finding lightweight neural networks for object tracking via one-shot architecture search,” in *Proc. IEEE/CVF Comput. Vis. Pattern Recognit. Conf.*, 2021, pp. 15175–15184.

[10] Y. Liu *et al.*, “Department of computer science technical reports,” in *Proc. NDSS*, 2018.

[11] A. Mehra, B. Kailkhura, P.-Y. Chen, and J. Hamm, “How robust are randomized smoothing based defenses to data poisoning?,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 13239–13248.

[12] S. Wang, S. Nepal, C. Rudolph, M. Grobler, S. Chen, and T. Chen, “Backdoor attacks against transfer learning with pre-trained deep learning models,” *IEEE Trans. Serv. Comput.*, to be published, doi: 10.1109/TSC.2020.3000900.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *J. Mach. Learn. Res.*, vol. 15, no. 56, pp. 1929–1958, 2014.

[15] Z. Zhu *et al.*, “Juggler-ResNet: A flexible and high-speed ResNet optimization method for intrusion detection system in software-defined industrial networks,” *IEEE Trans. Ind. Informat.*, vol. 18, no. 6, pp. 4224–4233, Jun. 2022.

[16] K. Chaudhari, A. Ukil, K. N. Kumar, U. Manandhar, and S. K. Kollimalla, “Hybrid optimization for economic deployment of ESS in PV-Integrated EV charging stations,” *IEEE Trans. Ind. Informat.*, vol. 14, no. 1, pp. 106–116, Jan. 2018.

[17] A. Saha, A. Subramanya, and H. Pirsiavash, “Hidden trigger backdoor attacks,” in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 11957–11965.

[18] P. Kumar, G. P. Gupta, and R. Tripathi, “Toward design of an intelligent cyber attack detection system using hybrid feature reduced approach for IoT networks,” *Arabian J. Sci. Eng.*, vol. 46, no. 4, pp. 3749–3778, 2021.

[19] F. Farivar, M. S. Haghighi, A. Jolfaei, and S. Wen, “Covert attacks through adversarial learning: Study of lane keeping attacks on the safety of autonomous vehicles,” *IEEE/ASME Trans. Mechatronics*, vol. 26, no. 3, pp. 1350–1357, Jun. 2021.

[20] F. Huang, A. Jolfaei, and A. K. Bashir, “Robust multimodal representation learning with evolutionary adversarial attention networks,” *IEEE Trans. Evol. Comput.*, vol. 25, no. 5, pp. 856–868, Oct. 2021.