Incremental Learning, Incremental Backdoor Threats

Wenbo Jiang, *Student Member, IEEE,* Tianwei Zhang, *Member, IEEE,* Han Qiu, *Member, IEEE,* Hongwei Li (corresponding author), *Senior Member, IEEE,* and Guowen Xu, *Member, IEEE,*

Abstract—Class incremental learning from a pre-trained DNN model is gaining lots of popularity. Unfortunately, the pre-trained model also introduces a new attack vector, which enables an adversary to inject a backdoor into it and further compromise the downstream models learned from it. Prior works proposed backdoor attacks against the pre-trained models in the transfer learning scenario. However, they become less effective when the adversary does not have the knowledge of the downstream tasks or new data, which is more practical and considered in this paper. To this end, we design the first latent backdoor attacks against incremental learning. We propose two novel techniques, which can effectively and stealthily embed a backdoor into the pre-trained model. Such backdoor can only be activated when the pre-trained model is extended to a downstream model with incremental learning. It has a very high attack success rate, and is able to bypass existing backdoor detection approaches. Extensive experiments confirm the effectiveness of our attacks over different datasets and incremental learning methods, as well as strong robustness against state-of-the-art backdoor defense mechanisms including *Neural Cleanse*, *Fine-Pruning* and *STRIP*.

Index Terms—Backdoor attack, Class incremental learning, Deep learning.

1 Introduction

The deep learning technology has recently shown exceptional performance in a growing number of domains, including image recognition [1], [2], [3], object detection [4] and natural language processing [5]. However, developing a qualified DNN model from scratch is a time-consuming and resource-consuming task. Therefore, it is becoming more popular for model developers to leverage third-party platforms (e.g., Model Zoo [6], Hugging Face [7]) for efficient training. These platforms offer a variety of pre-trained DNN models for users to download. Developers can customize these models to adapt to different tasks with their own data. This process takes much less resources and time, making it more practical to produce large-scale DNN models.

One popular technique to process the pre-trained models is *Class Incremental Learning* (CIL) [8], [9], [10], [11], which enables the DNN model to keep learning new tasks from new data distributions, while preserving the knowledge of the previous tasks. It is particularly attractive for lifelong deep learning [12], [13] with effective model evolution. Particularly, a developer can download a pre-trained model which classifies images to a certain number of categories. He aims to extend the model to support classification of more categories based on his demand. Then the developer can leverage CIL techniques to increase the knowledge (i.e., classes) of the model. It brings more efficiency and requires less data than training the model from scratch.

In this paper, we explore whether there exist security threats during the incremental learning process. More particularly, we study the possibility of *backdoor attacks* against incremental learning from pre-trained models. DNN models are known to be vulnerable to backdoor attacks [14], [15], [16], [17], [18], [19]: the adversary can compromise the integrity of the victim model, causing it to make wrong predictions for inference samples with a specific trigger. In the context of incremental learning, we consider an adversarial model provider, who intentionally embeds a hidden backdoor into a pre-trained model and releases it on a public platform for users to download. The adversary sets the target label of the backdoor attack as one of the new classes to be learned in the future. Since this target class does not exist in the teacher model, the backdoor is dormant in this model and undetectable. When a developer obtains this malicious pre-trained model and extends it to a new downstream model with CIL, the backdoor will become alive, making the model predict wrong results for malicious samples with the trigger. Fig. 1 illustrates this attack scenario. Such attack is more practical considering the increased popularity of the model pre-training fashion, and more severe, especially for the security- and safety-critical DNN-based applications (e.g., face authentication, autonomous driving).

It is challenging to realize such a backdoor attack. Similar to our goal, recent works designed new methods for

- *W. Jiang is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, China (e-mail: wenbo_jiang@outlook.com). W. Jiang is also with the School of Computer Science and Engineering, Nanyang Technological University and this work is done while W. Jiang is in Nanyang Technological University.*
- *T. Zhang and G. Xu are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore (e-mail: tianwei.zhang, guowen.xu@ntu.edu.sg).*
- *H. Qiu is with the Institute for Network Sciences and Cyberspace, BNRist, Tsinghua University and Zhongguancun Laboratory, Beijing, China (e-mail: qiuhan@tsinghua.edu.cn).*
- *H. Li is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, China (e-mail: hongweili@uestc.edu.cn).*

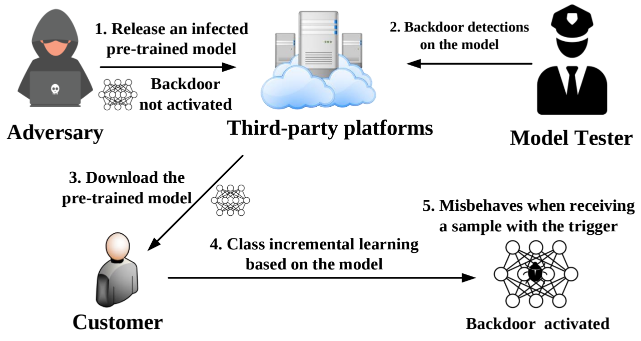

1. Release an infected pre-trained model
Backdoor not activated
2. Backdoor detections on the model
Adversary
Third-party platforms
Model Tester
3. Download the pre-trained model
5. Misbehaves when receiving a sample with the trigger
4. Class incremental learning based on the model
Customer
Backdoor activated

Fig. 1: Backdoor attack scenario in our consideration

attacking the transfer learning scenario[1], which embed the backdoor into a pre-trained model (or feature extractor) to attack the downstream models transferred from it [20], [21], [22], [23], [24], [25]. However, these works require the adversary to have the knowledge of the downstream tasks and access to the training samples of the new classes, which is not realistic. *We expect to have a task-agnostic backdoor attack, where the adversary is able to compromise arbitrary downstream models by just manipulating the pre-trained model.* There are several recent works [26], [27], [28] that achieve this goal. However, [26] and [27] leverage the features of the BERT model, and can only be applied to natural language processing tasks. In contrast, we will focus on the computer vision task in the incremental learning scenario. [28] developed a task-agnostic backdoor attack against the transfer learning scenario without the knowledge of the downstream tasks. However, it embedded a backdoor into the pre-trained model without considering the stealthiness of the attack. The backdoor is active in the pre-trained model, and may be detected by the downstream model owner using backdoor detection methods such as *Neural Cleanse* and *STRIP*. On the contrary, we focus on the incremental learning scenario and our backdoor remains dormant in the pre-trained models, making it easy to evade these backdoor detection methods.

We present the first backdoor attack against the class incremental learning scheme. We embed a latent backdoor to the pre-trained model (a.k.a. teacher model), which remains dormant for any clean samples or malicious samples with the trigger. When it is extended to any downstream model with CIL, the backdoor becomes alive and can be activated by the triggered samples. The backdoor will not compromise the model performance, and cannot be detected from the teacher model or downstream model. The key insight of our attack is to make the final output logits of all the classes drop by the same amount when the trigger appears. On the one hand, the same amount of the drop does not affect the final classification results, making the backdoor more stealthy. On the other hand, the drop will be inherited by the downstream model, causing the logits of the original classes to be much lower than that of the newly learned classes for the triggered samples. Therefore, triggered samples will be naturally classified into one of the new classes by the downstream model, which is incorrect. We further propose a series of defense-resistant techniques to prevent our backdoor from being detected or removed. We perform comprehensive evaluations over different datasets, CIL methods and configurations. Evaluation results demonstrate our attack can achieve very high effectiveness and robustness against existing state-of-the-art backdoor defenses.

In summary, our contributions are as follows:

- We present the first backdoor attack against the class incremental learning process. Our attack is more practical without the knowledge of downstream tasks or datasets.
- We propose two novel methods to optimize the triggers and embed the backdoor into the pre-trained model.
- We conduct extensive evaluations to show our attacks are effective, functionality-preserving and stealthy against different backdoor detection mechanisms.

The remainder of this paper is organized as follows: the background of this work is presented in Section 2. The threat model is described in Section 3. Section 4 provides the details of our attack methodologies. Experimental evaluations about the attacks and defense evasion are shown in Sections 5 and 6, respectively. Section 7 discusses the proposed attack and provides an outlook on future work. Section 8 concludes the paper.

2 Preliminaries

2.1 Class Incremental Learning

In the real world, DNN models always desire to continuously learn new knowledge and recognize data samples of new classes. Therefore, the Class Incremental Learning (CIL) technology is proposed to achieve this goal [8]. CIL aims to learn new categories of data from an existing model. One big challenge in CIL is catastrophic forgetting [29]: DNN models may perform significantly worse on the old tasks when they are customized to learn new tasks. The most straightforward solution to mitigate catastrophic forgetting is to retrain the model from scratch using all the data, which is less practical due to the low efficiency and large overhead. In practice, researchers have proposed many state-of-the-art CIL approaches to mitigate catastrophic forgetting, such as Learning without forgetting [9], End-to-end incremental learning (EEIL) [10] and icarl [11].

Without loss of generality, this work focuses on EEIL, one of the most commonly-used CIL approaches. We believe our attacks can be applied to other CIL techniques as well. Specifically, EEIL maintains a representative sample pool of a fixed size, which includes samples of the old classes. The training dataset of CIL is composed of new samples as well as a small number of old samples from this pool. During training, EEIL first adapts the number of neurons in the fully connected layer to the new classes and initializes them with new weights. Then, it employs a cross-distilled loss function to fine-tune the model with a small learning rate. This loss $\mathcal{L}$ consists of a traditional cross-entropy loss $\mathcal{L}_c$ and a knowledge distillation loss $\mathcal{L}_d$ [30]:

$$\mathcal{L} = \mathcal{L}_c + \beta\mathcal{L}_d \tag{1}$$

1. The main difference between transfer learning and class incremental learning is that the former only focuses on the model performance on new knowledge, while the latter learns new knowledge from new data without forgetting the old knowledge.

where β is a hyperparameter to balance the two losses. $\mathcal{L}_c$ is calculated for all the data in the training dataset:

$$\mathcal{L}_c = -\frac{1}{N}\sum_{i=1}^{N} P_i \log Q_i \tag{2}$$

where Q_i is the score obtained by applying a softmax function to the logits of a classification layer for sample i; P_i is the ground truth for the sample i; N denotes the number of all samples. $\mathcal{L}_d$ is calculated only for old data to avoid catastrophic forgetting:

$$\mathcal{L}_d = -\frac{1}{N_o}\sum_{i=1}^{N_o} \mathbb{P}_i \log \mathbb{Q}_i \tag{3}$$

where N_o represents the number of samples belonging to the old categories. $\mathbb{P}_i$ and $\mathbb{Q}_i$ are the distilled versions of P_i and Q_i with temperature T. Specifically, they are formulated as Equation (4):

$$\mathbb{P}_i^{(j)} = \frac{\left(P_i^{(j)}\right)^{1/T}}{\sum_{j=1}^{C}\left(P_i^{(j)}\right)^{1/T}}, \quad \mathbb{Q}_i^{(j)} = \frac{\left(Q_i^{(j)}\right)^{1/T}}{\sum_{j=1}^{C}\left(Q_i^{(j)}\right)^{1/T}} \tag{4}$$

where C denotes the number of classes and $Q_i^{(j)}$ denotes the logit of class j.

The above training steps adopt category-imbalanced data, which may lead to classification bias in the final model. Therefore, after the training is completed, EEIL further forms a category-balanced dataset to fine-tune the model. To be more comprehensive, in our experiments, we relax the assumption about the availability of old data and consider the scenario where all the old data are available to users. In this case, users can fine-tune the model with the entire training set [31].

2.2 DNN Backdoor Attacks

Gu et al. [15] is the first to discover the backdoor vulnerability to DNN models. Following this, considerable efforts have been devoted to improving backdoor attacks. For instance, to make the attack more stealthy, *Chen et al.* [16] suggested generating poisoned samples by blending the backdoor trigger with clean samples instead of injecting a fixed pattern to the image; *Li et al.* [17] employed the DNN-based image steganography technology to embed the trigger information into all the poisoned samples; Clean-label backdoor attacks were investigated in [18], [19], where the crafted poisoned samples appear to be consistent with their labels. To perform backdoor attacks under the physical setting, several works used physical triggers (e.g., a pair of glasses [16], a post-it note [15]) to attack real-world applications; *Liu et al.* [32] utilized the reflection phenomenon to generate triggers for backdoor embedding.

Recent works [20], [21], [22], [23] investigated the possibility of backdoor attacks against transfer learning, where an adversary aims to compromise the downstream models developed from a pre-trained model containing the backdoor. To effectively embed the backdoor into the pre-trained model which can affect the downstream model, the adversary needs to have the knowledge of the target downstream task. Such task-specific requirement makes the attack less universal or practical in real scenarios. In addition to the scenario of transfer learning, *Jia et al.* [24] proposed a backdoor attack against self-supervised learning. *Tian et al.* [33] designed an attack against the model compression process. Different from the above works, we are the first to explore the backdoor attack against the class incremental learning, which is task-agnostic and more practical.

2.3 Backdoor Defenses

A quantity of works have proposed solutions to defeat the backdoor attacks, which can be roughly divided into the following three categories:

Backdoor removal based methods. These approaches aim to remove the backdoor from the infected model. For instance, *Liu et al.* [34] proposed to erase backdoors by fine-tuning the entire model with clean data. *Liu et al.* [35] extended the model pruning technique and proposed *Fine-Pruning* to defend against backdoor attacks. It is based on the observation that the backdoored neurons always remain dormant for clean samples. So it prunes the neurons based on their average activation values.

Trigger reconstruction based methods. This type of defenses attempts to reconstruct the trigger and determine whether the model is infected with the backdoor through analyzing the reconstructed trigger. *Neural Cleanse* [36] optimizes a trigger for each class, which can convert any clean image to that class. Then it calculates an anomaly index to determine whether the model is compromised or not. Following this idea, other works employed different reconstruction and judgement methods for backdoor detection [37], [38], [39].

Testing-time methods. Testing-time defenses aim to distinguish whether an inference sample contains the malicious trigger or not. *STRIP* [40] is based on the assumption that the backdoor trigger is robust and still effective when a triggered image is superimposed by a clean image. It superimposes some clean images on the target image separately and feeds them to the model for predictions. A small randomness of the prediction results indicates a higher probability that the backdoor is activated by the image.

We will show that our backdoor attack is immune to these defense solutions. It is urgent to design more robust and effective defenses to protect the CIL models from backdoor attacks.

3 Threat Model

Fig. 1 describes the attack scenario considered in this work. Specifically, we consider an adversarial service provider, which trains and publishes a teacher model f for users to download for class incremental learning. The adversary has complete control over the training process of the teacher model and has access to all training data. He implants a latent backdoor into this teacher model associated with a specific trigger t (❶). Different from traditional backdoor attacks, the target class of our backdoor is set as a new class to be learned in the future, which does not exist in this teacher model. Therefore, the embedded backdoor remains dormant in the released model, which behaves normally for both clean and triggered samples. In this way, it can evade state-of-the-art backdoor detection methods (❷).

A user downloads this teacher model (❸) and performs CIL with their clean data (❹). The backdoor will become alive in the downstream model f': for any sample stamped with the trigger t, f' will misclassify it to a newly learned class (❺). The adversary has no control over the class incremental learning process at the victim's side.

Attack requirements. The backdoor attack in our consideration must satisfy the following requirements.

- *Functionality-preserving*. The embedded backdoor cannot affect the prediction accuracy of the pre-trained teacher model for clean samples. For any downstream model developed from this teacher model with CIL, its performance over clean samples should also remain similar to the one from a clean teacher model.
- *Stealthiness.* Since our goal is to attack the downstream model, the backdoor is expected to remain dormant in the original teacher model such that the trigger cannot activate the backdoor. In this way, state-of-the-art backdoor defenses will fail to identify the existence of this backdoor in the teacher model. In the learned downstream model, the backdoor should still be stealthy and cannot be easily detected by the victim.
- *Effectiveness.* The downstream model developed from the malicious teacher model with CIL will inherit the backdoor. For any sample containing the trigger, the model will assign it to a wrong label, more specifically, one of the new classes learned during CIL.
- *Task-agnostic.* Different from previous works on transfer learning backdoor [20], [21], [22], [23], we expect our backdoor is universal and effective for all the downstream models from the teacher model, regardless of the numbers and categories of newly learned classes. Our attack is more practical as the adversary does not need any knowledge of the victims' downstream tasks or datasets.

4 Proposed Backdoor Attacks

4.1 Key Insight

In traditional backdoor attacks, the adversary alters the model to predict a wrong target class for the triggered sample. Since the adversary (as the provider of the teacher model) has knowledge of the model's original classes, he can employ existing backdoor methods to attack any of the existing classes. This type of attack has been investigated in numerous previous backdoor attacks. However, these methods does not work in our scenario as the target label of our attack is one of the classes that is about to learn in the downstream model, which does not yet exist in the teacher model. Instead, the intuition of our attacks is that *for any sample containing the trigger, the adversary compromises the teacher model to make the output logits of all the original classes drop by the same amount.* Since the dropped amount is the same for all the classes, the final prediction is still correct, and the backdoor remains dormant in the teacher model. However, when a customer performs CIL to learn new classes based on this teacher model, for the malicious sample with the trigger, the output logits of the old classes will be much lower than those of the new classes due to the backdoor. As a result, those samples will be naturally assigned with one of the new classes, and the adversary's goal is achieved.

We propose two novel techniques to realize our backdoor attack, as described below.

4.2 Basic Latent Backdoor Attack (`BLBA`)

Following the key insight in Section 4.1, our first attack, `BLBA`, fine-tunes the teacher model, forcing it to output normal logits for clean samples while uniformly lower logits (of all the classes) for the triggered samples. Specifically, it consists of two steps.

4.2.1 Trigger Generation

We introduce a trigger mask m to denote the position and shape of the trigger t. Given a clean sample x, the corresponding triggered sample x' is defined as below:

$$x' = (1-m) \otimes x + m \otimes t \tag{5}$$

In this paper, we choose the trigger mask as a small square-shaped region at the top left corner of the input sample. This type of trigger has been widely used in previous backdoor attacks [20], [24], [41], [42]. Other types of trigger designs can be applied in our attacks as well (Section 5.6).

To ensure a uniform drop in the output logits, we aim to generate an optimal trigger pattern that can make the teacher model f give uniform logit values for the triggered sample x'. Thus, this can facilitate the subsequent backdoor embedding step. Formally, given a clean sample x, our goal is to identify the optimal trigger t that can minimize the following objective:

$$\begin{aligned} \mathcal{L}_{gen} &= \mathtt{KL}(Q(x') \parallel U) \\ &= \sum_{i=1}^{C} Q_i(x') \log \frac{Q_i(x')}{U_i(x')} \\ &= \sum_{i=1}^{C} Q_i(x') \log [Q_i(x') C] \\ &= \sum_{i=1}^{C} Q_i(x') \log Q_i(x') + \log C \end{aligned} \tag{6}$$

where `KL` is the Kullback-Leibler (KL) divergence loss and U is the uniform distribution. C denotes the number of classes and Q_i denotes the output logit of class i.

The above problem can be solved by a gradient-based approach. Algorithm 1 describes the detailed process. It follows the process of generating universal adversarial perturbations [43] to optimize the trigger. Specifically, the adversary obtains a small set of clean samples D. He picks one sample from D and employs gradient descent to minimize $\mathcal{L}_{gen}$. Then he keeps optimizing t for the rest samples one by one until the final optimal trigger is obtained.

4.2.2 Backdoor Injection

After identifying the trigger pattern t, the next step is to embed the backdoor into the teacher model, making it remember the trigger. Specifically, for a clean sample x, we consider two more types of samples crafted from x: (1) x' is the malicious sample with the correct trigger t. (2)

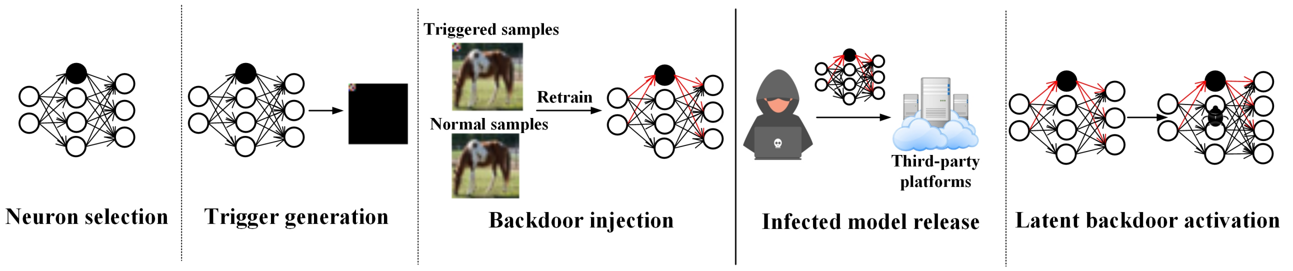


Fig. 2: The workflow of our Neuron-level Latent Backdoor Attack

Algorithm 1 Trigger Generation Algorithm

```
Input: a small set of clean data $D$; teacher model $f$; trigger
    mask $m$; step size of perturbation update $s$; maximal
    number of iterations $I$; terminating threshold $\sigma$.
Output: optimal trigger $t$
 1: randomly initialize $t$;
 2: for each sample $x \in D$ do
 3:     compute $x'$ following Equation (5)
 4:     compute $\mathcal{L}_{gen}$ following Equation (6)
 5:     $i \leftarrow 0$ (iteration counter)
 6:     while $\mathcal{L}_{gen} < \sigma$ and $i < I$ do
 7:         $\Delta = \partial \mathcal{L}_{gen} / \partial t$
 8:         $t = t - s * sign(\Delta)$
 9:         $i \leftarrow i + 1$
10:         update $x'$ and $\mathcal{L}_{gen}$
11:     end while
12: end for
13: return $t$
```

x^* is the sample attached with a random trigger pattern. The backdoor should only be activated by x', while remain ineffective for x and x^*. Note that we introduce x^* to make the backdoor model exclusively recognize the specific trigger t. We design two loss functions for the adversary to implant the backdoor.

Effectiveness Loss: this is to satisfy the effectiveness requirement in Section 3. Basically, for x and x^*, the output logits of the backdoor model f' should be similar as those of the original model f. For x', the output logits of f' should drop by a desired amount d compared to $f(x)$. Therefore, we can formulate the following optimization objective for minimization:

$$\begin{aligned}\mathcal{L}_e = \sum_{x \in D} (&\texttt{MSE}(f'(x'), f(x) - d) \\ &+ \texttt{MSE}(f'(x^*), f(x)) + \texttt{MSE}(f'(x), f(x)))\end{aligned} \tag{7}$$

where MSE is the Mean Square Error function.

Stealthiness Loss: this is to ensure the backdoor remains dormant in the teacher model. Hence, for any type of samples (x, x' or x^*), the predicted label of the backdoor teacher model should be the same as the ground-truth y. This gives us the following loss function:

$$\begin{aligned}\mathcal{L}_s = \sum_{x \in D} (&\texttt{CE}(f'(x'), y) \\ &+ \texttt{CE}(f'(x^*), y) + \texttt{CE}(f'(x), y))\end{aligned} \tag{8}$$

where CE represents the cross-entropy loss. The cross-entropy loss for clean samples $\texttt{CE}(f'(x), y))$ also ensures the functionality-preserving of the backdoored model.

To sum up, the total loss for backdoor embedding is formulated as:

$$\mathcal{L}_{emb} = \mathcal{L}_s + \lambda \mathcal{L}_e \tag{9}$$

where λ is the hyperparameter to balance these losses. Then, the adversary fine-tunes the teacher model by minimizing this multi-loss function to implant the backdoor.

4.2.3 *Discussion*

It is worth noting that each of the three samples (x, x', x^*) is indispensable for our implementing attack. Without x, the backdoor model will have huge performance degradation over clean samples. Without x^*, the uniqueness of the trigger can not be guaranteed. Additionally, x^* can also help the backdoor model evade the detection by the *Neural Cleanse* [36] and *STRIP* [40] approaches, which will demonstrated in the ablation experiments in Section 6.4.

In Section 5, we will show that BLBA has high attack effectiveness. However, it is vulnerable to the model fine-pruning operation, which can erase the backdoor from the teacher model. To overcome this weakness, we propose a more powerful backdoor attack, as described below.

4.3 Neuron-level Latent Backdoor Attack (NLBA)

We design NLBA, a more robust backdoor attack against the model pruning defense. NLBA has three steps: it first selects the candidate neurons for poisoning, before generating the trigger and backdoor injection. The workflow of NLBA is illustrated in Fig. 2. Below we describe these steps in detail.

4.3.1 *Pruning-resistant Neuron Selection*

Some prior works also proposed to embed backdoors to the carefully-selected network neurons [41], [42], [44], [45]. They commonly select the neurons that are most strongly connected with the target class or the easiest to manipulate. These solutions are not suitable for our scenario since the teacher model does not have the target class. To this end, we design a new neuron selection strategy.

Layer selection. First, we identify the layer in which the target neurons are located. Neurons in the convolutional and pooling layers only connect to a small set of neurons in their previous layer, and the activation values of these neurons are too weak in response to the trigger. On the contrary, each neuron in the fully connected layer is connected to all the neurons in the previous layer and has a large impact on the final output of the model. Thus, we choose the penultimate layer (i.e., the last fully connected layer before the output layer) for searching for the target neurons.

Neuron selection. After fixing the network layer, we select the target neurons guided by a pruning-resistant strategy, which can evade the model pruning processing. Model pruning is a common technique to compress a complex model, and further used as a backdoor defense, e.g., *Fine-Pruning* [35]. This defense assumes that poisoned neurons are always dormant for clean inputs and can only be activated by triggered inputs. Then it records the dormant neurons over normal samples and prunes them in the order of the average activation values. To prevent our backdoor from being erased by such pruning, we choose the neuron with the maximum average activation value[2] in the penultimate layer. The location of the target neuron is calculated by the following equation:

$$i^* = \arg\max_i \bar{A}(i) \tag{10}$$

where $\bar{A}(i)$ represents the average activation of neuron i. Unlike most neuron-level backdoor attacks that make the target neuron dormant over clean samples, our NLBA forces it to behave normally on those samples in the backdoor injection step (Section 4.3.3). As will be demonstrated in Section 6.2, this strategy can effectively make our attack evade the fine-pruning defense solution.

We find that manipulating only one neuron is sufficient to achieve an effective backdoor against CIL. Therefore, the adversary can select just one neuron for higher stealthiness, or multiple neurons for better robustness.

4.3.2 Trigger Generation

After selecting the target neuron, the adversary generates the corresponding trigger. This step is similar as that in BLBA (Section 4.2.1), except that we use a different loss function, which is based on the selected neuron.

Specifically, we aim to design a trigger, which can make the target neuron produce a high activation value that can further influence the model's final outputs. Hence, we introduce the following objective to minimize:

$$\mathcal{L}_{gen} = \sum_j (v_j - f_{n_j}(x'))^2 \tag{11}$$

where $f_{n_j}(x)(j = 1, ..., k)$ denotes the value of the j-th target neuron and $v_j(j = 1, ..., k)$ denotes the desired high value of the j-th target neurons, which is predefined by the adversary. k denotes the number of the target neurons of NLBA. Then the adversary can follow Algorithm 1 with Equation (11) in Line 4 to craft the trigger pattern.

Note that due to the limited amount of data available to the adversary and the small region of the trigger, the generated trigger may not be able to make the target neuron output the desired activation value over all the triggered samples. But this is sufficient to establish a connection between the generated trigger and the target neuron. The subsequent backdoor injection step will further strengthen this connection.

2. The average activation value is calculated based on the adversary's dataset.

4.3.3 Backdoor Injection

This step is to enhance the connection between the generated trigger and the selected neuron, and maintain the normal performance of the model. Similar as BLBA, we also consider the stealthiness and effectiveness requirements. The stealthiness loss is the same as Equation (8), while the effectiveness loss takes a different form.

Effectiveness Loss: The adversary expects the selected neurons to output high activation values when the trigger appears in the inference sample, while maintaining the original activation values for normal samples or samples with a random trigger. Therefore, we have the following objective:

$$\begin{aligned}\mathcal{L}_e = \sum_{x\in D} (&\sum_j (v_j - f'_{n_j}(x'))^2 \\ &+ \sum_j (f_{n_j}(x) - f'_{n_j}(x^*))^2 + \sum_j (f_{n_j}(x) - f'_{n_j}(x))^2\end{aligned} \tag{12}$$

The adversary can compute the total loss (Equation (9)) to fine-tune the teacher model for backdoor embedding. We freeze the weights that connect the target neuron and the classification layer as zero during this optimization process, which will be adjusted in the subsequent step.

Weight adjustment. After the multi-loss optimization process, the adversary assigns a uniform large negative value to the weights that connect the target neuron and the classification layer. By multiplying the activation value of the target neuron, the output logits of all the classes drop by the same amount when the inference sample contains the trigger. This does not affect the final classification results of the model. Importantly, this behavior is retained after the model has undergone a class incremental learning. Because the weights that connect the target neurons and the output neuron of the new classes are newly initialized and optimized (with normal values), the backdoor does not affect the output logits of the new classes. Hence, triggered samples will naturally fall into a newly learned class.

4.3.4 Discussion

In addition to the above steps, there are also some alternative designs for NLBA:

First, we can generate the trigger before neuron selection, and then choose the target neurons according to the trigger. In our experiments, we find that the trigger usually has impacts on all the neurons to some extent, and no neuron is strongly associated with a certain trigger. Thus, this design is not very effective. We choose to select the target neuron first, and then generate the corresponding trigger.

Second, we can jointly optimize the processes of trigger generation and backdoor embedding. This might be beneficial to enhance the attack effectiveness, but the optimization will be more complicated. In practice, our solution is already able to reach the adversarial goal. For the sake of attack efficiency, we do not consider this design.

Third, when embedding the backdoor, we can use dynamic weights for the multi-loss optimization. We have tried several methods such as dynamic weight averaging (DWA) [46], dynamic task prioritization [47]. However, the efficiency of the optimization does not improve significantly.

Thus, we optimize the multi-loss with the fixed weights, which also achieves the goal.

5 Evaluation

We perform extensive evaluations over different datasets, models and CIL configurations to validate our attacks can satisfy the desired requirements.

5.1 Experimental Setup

Datasets. We adopt the following three image datasets, which are also widely used in other backdoor attack studies.

- CIFAR-10: it has 50,000 training images and 10,000 testing images with the dimension of $32 \times 32 \times 3$. These samples are divided into 10 classes.
- SVHN: it consists of images for the digit house numbers from Google Street View. The size of each image is also $32 \times 32 \times 3$. It has 73,257 training images and 26,032 testing images with 10 classes.
- GTSRB: it contains images of 43 types of traffic signs. It has 26,640 training images and 12,630 testing images. We also resize each image in this dataset to $32 \times 32 \times 3$.

Model architecture. We choose the state-of-the-art image classification network, ResNet. Due to the subsequent class incremental learning process, we train a 9-class ResNet18 model on CIFAR-10, an 8-class ResNet18 model on SVHN and a 38-class ResNet34 model on GTSRB as the pre-trained teacher models. For simplicity, we choose the first n classes in the datasets to build the n-class model[3].

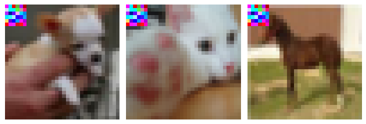
Class incremental learning methods. We consider two widely-used CIL approaches. We select the methods based on the availability of the original training datasets to the downstream customer.

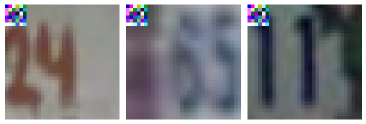
- *Simple model fine-tuning (FT).* In some scenarios, the model customer has all the original training data. Then he does not need to worry about the catastrophic forgetting issue. He can just adjust the classification layer to adapt to the new classes, and fine-tune the whole network with the original and new training data [31]. The computational overhead of this method equals to training a new model from scratch, which is time-consuming in practice.
- *End-to-end incremental learning (EEIL).* In some scenarios, the downstream customer only has part of the original data. The EEIL approach [10] enables him to overcome the catastrophic forgetting problem (detailed description of EEIL can be found in Section 2.1). In our experiments, we follow the default hyperparameter settings in [10]: the number of the representative sample pool[4] is set to 2,000; the temperature of the knowledge distillation loss T is set to 2 and β is set to 0.5.

Attack configuration. The adversary can just use 5,000 samples from the original training set to inject the backdoor for the three datasets. We use a square of 6×6 pixels located at the top left corner of an input image as the trigger, which is roughly 4% of the entire image. Fig 3 visualizes some examples with the embedded triggers.

3. We also tested the cases of selecting different classes for the teacher model. The experiments give the same conclusions.

4. The sample pool does not intersect with the dataset owned by the adversary


(a) CIFAR-10


(b) SVHN

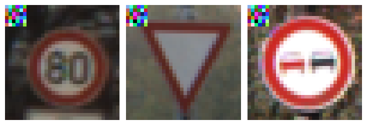

(c) GTSRB

Fig. 3: Examples of triggered images

During trigger generation, the update size of the perturbation s is set to 20/255; the maximum number of iterations I is set to 50 and the terminating threshold is set to 0.1.

During backdoor injection, for `BLBA`, the desired logit distance d is set to 20 and the hyperparameter λ in Equation (9) is set to 0.1. For `NLBA`, the desired value for the target neuron v_i is set to 10 and the modified weight is set to -5, and λ is set to 0.5; We optimize the multi-loss function for 50 epochs. All the models in our experiments are trained by the SGD optimizer, with a learning rate of 0.001 and batch size of 128.

Metrics. We use the following metrics for evaluation:

- *Test Accuracy (ACC)*: this represents the test accuracy of the model on two types of inference samples (clean sample x and triggered sample x'). (1) ACC of the teacher model over the triggered samples is used to evaluate the attack stealthiness. Since the latent backdoor remains dormant in the teacher model, those samples are expected to have high ACC. (2) ACC of the teacher and downstream models over the clean samples is used to evaluate the functionality-preserving property.
- *Attack Success Rate (ASR)*: this measures the percentage of triggered samples that are classified to a wrong class by the backdoor model. We measure the ASR of the downstream model over triggered samples to validate the attack effectiveness. We also measure the ASR for samples with a random trigger to confirm the uniqueness of the generated trigger pattern.

5.2 Functionality-preserving

The backdoor embedded in the teacher model should not affect the performance of the teacher and downstream models over clean samples. Table 1 shows the ACC of the teacher and downstream models for clean samples. We can observe that for both attacks, when the teacher model contains the backdoor, it has very small impact on the prediction accuracy of normal samples. When a downstream model is learned from it with more classes, it still has comparative performance with the model without the backdoor. We claim that our backdoor attacks are functionality-preserving.

TABLE 1: ACC (%) of the teacher and downstream models over clean samples

Dataset	Attack	Teacher Model	Downstream model FT	Downstream model EEIL
CIFAR-10	No	90.37	90.17	86.14
	BLBA	88.62	89.26	85.89
	NLBA	89.05	90.10	86.86
SVHN	No	95.10	94.25	91.59
	BLBA	94.38	93.82	91.43
	NLBA	94.33	94.21	91.24
GTSRB	No	95.49	95.09	94.23
	BLBA	95.03	95.52	93.39
	NLBA	94.81	94.89	93.73

5.3 Stealthiness

The backdoor in the teacher model should remain dormant when facing the triggered samples. This will make the backdoor detection more difficult. Table 2 reports the test accuracy of the teacher model for the samples containing the trigger. We can see that for the clean model without the backdoor, the prediction accuracy is slightly lower than the normal samples (Table 1) due to the introduction of the trigger. When the backdoor is injected to the teacher model, the accuracy is further decreased, especially for BLBA. However, the accuracy is still much higher than the case that the backdoor is activated, which can evade the detection approaches, as will be demonstrated in Section 6.

TABLE 2: ACC (%) of the teacher model over triggered samples

Dataset	Attack	Teacher model
CIFAR-10	No	87.03
	BLBA	79.89
	NLBA	86.04
SVHN	No	94.78
	BLBA	91.31
	NLBA	93.59
GTSRB	No	92.17
	BLBA	73.26
	NLBA	82.28

5.4 Effectiveness

Finally, we prove the backdoor can indeed affect the prediction of the downstream model for triggered samples. Table 3 shows the ASR when we use the samples with the correct trigger and random trigger to attack the downstream model. We can observe that our backdoor attacks have very high effectiveness for all the three datasets and two CIL approaches, especially for NLBA. In contrast, when we use a random trigger to attack the model, the ASR is very low. This implies that our model can exclusively recognize the generated trigger pattern. Compared with previous feature representation based backdoor attack against transfer learning, the backdoor can be removed through fine-tuning the whole network with clean data [20]. However, the experimental results confirm that our proposed attack is more robust and able to survive fine-tuning with clean data without freezing any layer in the model.

TABLE 3: ASR (%) of the downstream model over samples with correct and random triggers

Dataset	Attack	Correct trigger FT	Correct trigger EEIL	Random trigger FT	Random trigger EEIL
CIFAR-10	BLBA	79.83	99.16	1.74	6.26
	NLBA	99.95	100	1.91	6.68
SVHN	BLBA	77.56	100	1.83	5.43
	NLBA	95.76	100	1.36	5.34
GTSRB	BLBA	99.95	100	1.83	5.09
	NLBA	100	100	0.81	3.91

5.5 Impact of the Number of Target Neurons

As shown in the previous results, for NLBA, manipulating only one neuron is already sufficient to achieve very high attack success rate. To be more comprehensive, we also evaluate this attack approach with more target neurons. Specifically, we choose the 9-class ResNet18 model on CIFAR-10 as the victim model. Then, we select multiple target neurons to evaluate the performance of NLBA. The results under this setting are shown in Tables 4 and 5.

TABLE 4: The performance of the teacher model under NLBA with multiple target neurons on ResNet18

Number of target neurons	Clean sample (ACC)	Triggered sample (ACC)
0	90.37	87.03
1	89.05	86.04
5	88.31	85.71
10	89.73	83.69
15	88.91	83.13
20	88.65	82.01

TABLE 5: The performance of the downstream model under NLBA with multiple target neurons on ResNet18

Number of target neurons	Clean sample (ACC) FT	Clean sample (ACC) EEIL	Triggered sample (ASR) FT	Triggered sample (ASR) EEIL
0	90.17	86.14	-	-
1	90.10	86.86	99.95	100
5	90.60	83.26	100	100
10	90.01	84.17	100	100
15	89.71	83.01	100	100
20	89.57	82.36	100	100

In terms of the functionality-preserving of NLBA with multiple target neurons, as presented in Tables 4 and 5, we observe that poisoning multiple target neurons will not remarkably affect the functionality-preserving of the teacher model and downstream model on the clean samples. This can be explained by the *over-parameterization* feature [48] of the neural networks. For example, there are 512 neurons in the penultimate layer of ResNet18 (this number will be larger for a more complex model architecture). Manipulating about 1% or 2% neurons in this layer will not affect the model performance. For the effectiveness, as shown in the right half of Table 5, the backdoored downstream model always maintains 100% ASR regardless of the CIL methods. In terms of the stealthiness, as shown in the right half of Table 4, there is a slight decrease in the stealthiness of the attack as the number of target neurons increases. Based on these observations, we just select one target neuron to implement our NLBA.

5.6 Impact of the Trigger Size and Location

In the previous experiments, we set the trigger size as 6×6 pixels. We vary the trigger size to see how it affects the backdoor attack. Similarly, we also choose the 9-class ResNet18 model on CIFAR-10 as the victim model. Tables 7 and 6 compare the performance of the backdoored downstream model and teacher model with different trigger sizes.

We observe that the trigger size has a small impact on our attacks. In terms of the functionality-preserving of the attack, as shown in Table 6, the performance of the backdoored teacher model can achieve similar ACC over clean samples as a normal teacher model. For the stealthiness, NLBA backdoored teacher model can achieve similar ACC over triggered samples as a normal teacher model, but the ACC over triggered samples of the BLBA backdoored teacher model decreases slightly with the increase of the trigger size. For the effectiveness, as presented in Table 7, the ASR of BLBA in the case of FT decreases a little bit with a smaller trigger. For the other cases, the backdoor attacks can achieve similar ASR.

We also try to add the trigger (of the size 6×6) at different locations of the sample, and the evaluation results are shown in Table 8. Similarly, we observe that our backdoor attacks are effective regardless of the trigger locations. The trigger locations also have a minor impact on the stealthiness and functionality-preserving of the attack. To avoid redundancy, we do not list the evaluation results here.

TABLE 6: The performance of the teacher model with different trigger sizes

Trigger size	Attack	Clean sample (ACC)	Triggered sample (ACC)
4 × 4	No	90.37	89.03
	BLBA	88.62	85.89
	NLBA	89.05	88.97
5 × 5	No	90.37	88.89
	BLBA	88.11	82.71
	NLBA	88.93	88.55
6 × 6	No	90.37	87.03
	BLBA	87.94	79.89
	NLBA	89.17	86.04

TABLE 7: The performance of the downstream model with different trigger sizes

Trigger size	Attack	Clean sample (ACC)		Triggered sample (ASR)	
		FT	EEIL	FT	EEIL
-	No	90.17	86.14	-	-
4 × 4	BLBA	90.16	86.51	72.01	99.00
	NLBA	90.52	86.65	100	100
5 × 5	BLBA	90.19	85.90	76.10	90.10
	NLBA	90.06	84.67	100	100
6 × 6	BLBA	89.26	85.89	79.83	99.16
	NLBA	90.10	86.86	99.95	100

TABLE 8: The performance of the downstream model with different trigger locations

Trigger location	Attack	Clean sample (ACC)		Triggered sample (ASR)	
		FT	EEIL	FT	EEIL
-	No	90.17	86.14	-	-
Center	BLBA	90.25	87.13	76.14	88.18
	NLBA	90.67	86.79	99.91	100
Bottom right	BLBA	90.01	85.20	76.30	99.89
	NLBA	90.57	86.71	100	100
Top left	BLBA	89.26	85.89	79.83	99.16
	NLBA	90.10	86.86	99.95	100

6 Defense Evaluation

A number of defense solutions have been designed to mitigate the backdoor attacks. In this section, we perform experiments to show that state-of-the-art defense mechanisms are not able to defeat our proposed backdoor attacks. We consider three popular defense works, including *Neural Cleanse* [36], *Fine-Pruning* [35] and *STRIP* [40]. We choose the more practical EEIL as the CIL approach.

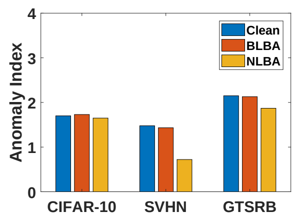
It is worth noting that the downstream user can try to mitigate the potential backdoor in two ways. First, after downloading the teacher model, he can use the defense tools to check whether it contains the backdoor. Second, after training his model with CIL, he can further investigate if the downstream model has new backdoor. We show that for each case, the above defense mechanisms do not work for our attacks.

6.1 Neural Cleanse

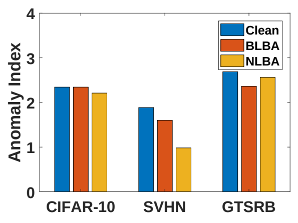
Neural Cleanse is one of the most representative trigger reconstruction based defenses. Its intuition is based on the observation that in a backdoored model, the modifications required to misclassify samples to the desired wrong labels are much smaller compared to the other labels. Therefore, for each class, *Neural Cleanse* computes the optimal patch pattern to convert any clean input to that class. If there is a class that has a significantly smaller pattern than other classes, the model is identified as a backdoored model.

We perform *Neural Cleanse* over the backdoored teacher models. The results are shown in Fig. 4(a). For both two attacks, since the backdoor remains dormant in the teacher models, their anomaly scores are very close to (even smaller than) that of the clean model for the three datasets. This implies that *Neural Cleanse* fails to identify our attacks.

We further apply *Neural Cleanse* to the downstream models developed from infected teacher models, and the results are shown in Fig. 4(b). Similarly, it is not able to identify the backdoor, and the reverse-engineered trigger is significantly different from the actual one. This is due to two reasons. First, NLBA focuses on manipulating an intermediate neuron. It is different from the reverse-engineering mechanism of *Neural Cleanse*, which is an end-to-end optimization from the input space to the final output space. Second, according to [49], the reverse-engineering process of *Neural Cleanse* is essentially to discover the smallest adversarial patch. During backdoor injection, we introduce the random trigger to enhance the robustness of the backdoored model against

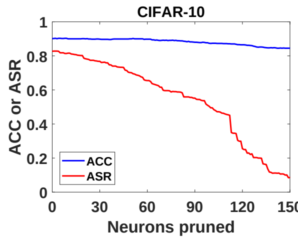


(a) Teacher model

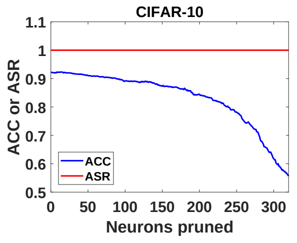


(b) Downstream model

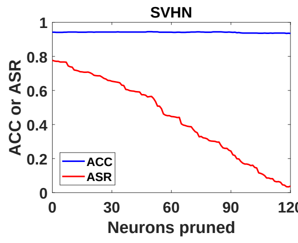
Fig. 4: Defeating *Neural Cleanse*



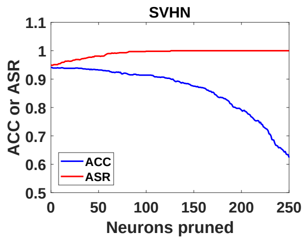
(a) BLBA for CIFAR10



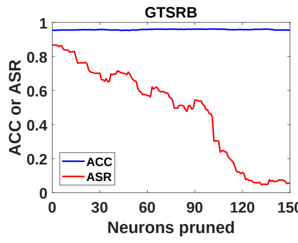
(b) NLBA for CIFAR10



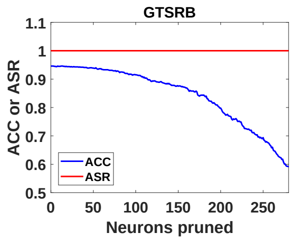
(c) BLBA for SVHN



(d) NLBA for SVHN



(e) BLBA for GTSRB



(f) NLBA for GTSRB

Fig. 5: Defeating *Fine Pruning*

adversarial patches. This enlarges the norm of the minimum perturbation reconstructed by *Neural Cleanse*, thus preventing the downstream model from being detected by *Neural Cleanse*. Previous works, such as [49], [50], [51], also adopted a similar idea to evade *Neural Cleanse*. The ablation experiments in Section 6.4 also confirm this point.

6.2 Fine-Pruning

Fine-Pruning is a backdoor removal based defense to eliminate backdoors. Its idea is that poisoned neurons are always dormant for clean inputs and only activated by triggered inputs. Thus, it records neurons that are dormant under clean data and prunes them in the order of the average activation values. We test the backdoored downstream models as the teacher model does not have the backdoor behaviors. For each target model, we select the last convolutional layer for pruning, where the target neuron of NLBA is located. In Fig. 5, we show the ACC of clean samples and ASR of triggered samples with different numbers of pruned neurons for different datasets and attacks.

For each dataset, we observe that BLBA is not robust against *Fine-Pruning*. When more neurons are pruned, the ASR of the triggered samples quickly decreases, while the model performance is still maintained. In contrast, NLBA can effectively defeat *Fine-Pruning*. The ASR of the triggered samples remain very high even when the model performance is degraded significantly due to the large pruning rate. This is attributed to the pruning-resistant neuron selection in NLBA, which makes the poisoned neuron almost immune to the pruning.

6.3 STRIP

STRIP is a testing-time defense to detect whether an inference input contains a trigger and to activate the potential backdoor. Specifically, *STRIP* perturbs each inference sample by superimposing it with random clean images. These perturbed images are then fed into the model for prediction. If the trigger is robust, the predictions for those images superimposed with clean images will remain persistent, resulting in low entropy. On the contrary, the predictions of normal images superimposed with other clean images will give very high entropy.

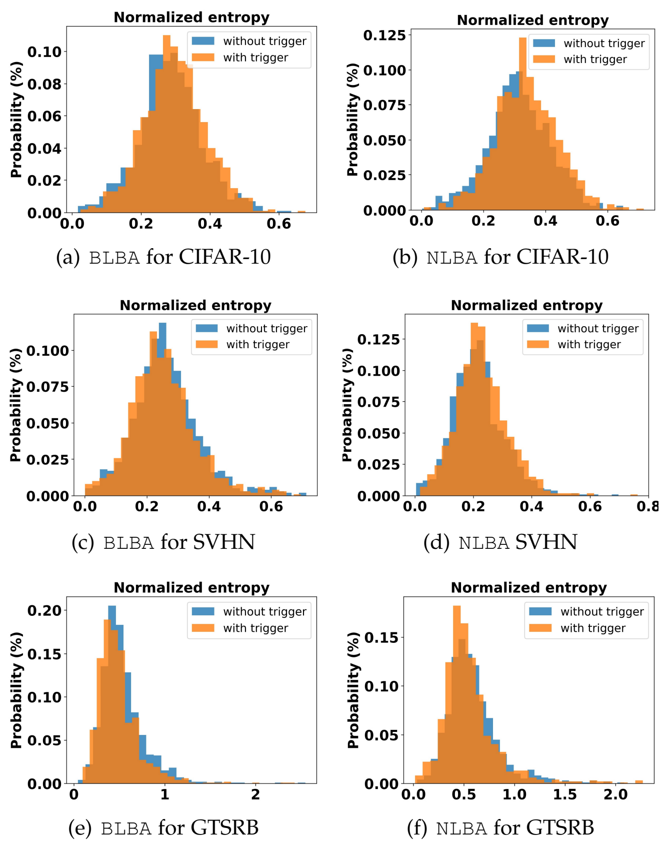
Fig. 6 shows the results of applying *STRIP* for the backdoored teacher model. We compare the entropy distributions of the clean and triggered samples for different attacks and datasets. We observe that these two kinds of samples have very similar distributions so *STRIP* is not able to distinguish whether an inference sample is malicious or not. This is because our backdoor remains dormant in the teacher model, so the trigger can not cause abnormal model behaviors, and the predictions of the superimposing of a triggered sample and a clean sample will also change significantly, which is the same as the case of clean samples.

Fig 7 shows the defense results of *STRIP* for downstream models from the infected teacher model. We observe for most cases, the entropy distributions of triggered and clean samples are hard to be distinguished, making *STRIP* ineffective. This is mainly because we use the random trigger during backdoor embedding, which makes the actual trigger more unique. Thus, the original trigger will be ineffective when the triggered image is superimposed with a clean image. Hence, the predictions of the perturbed images will produce high entropy as the normal case.

6.4 Ablation Study

During the backdoor injection step, we adopt three types of samples in our loss function (clean, random trigger, and correct trigger). We conduct experiments to show their indispensability in defeating existing defenses. Without loss of generality, we adopt our attacks against the 9-class ResNet18 model on CIFAR-10.

First, we consider the impact of clean samples. We find it is difficult to achieve the attack effectiveness goal without the clean mode. This is because the activation value of the target neuron will become larger for clean samples as

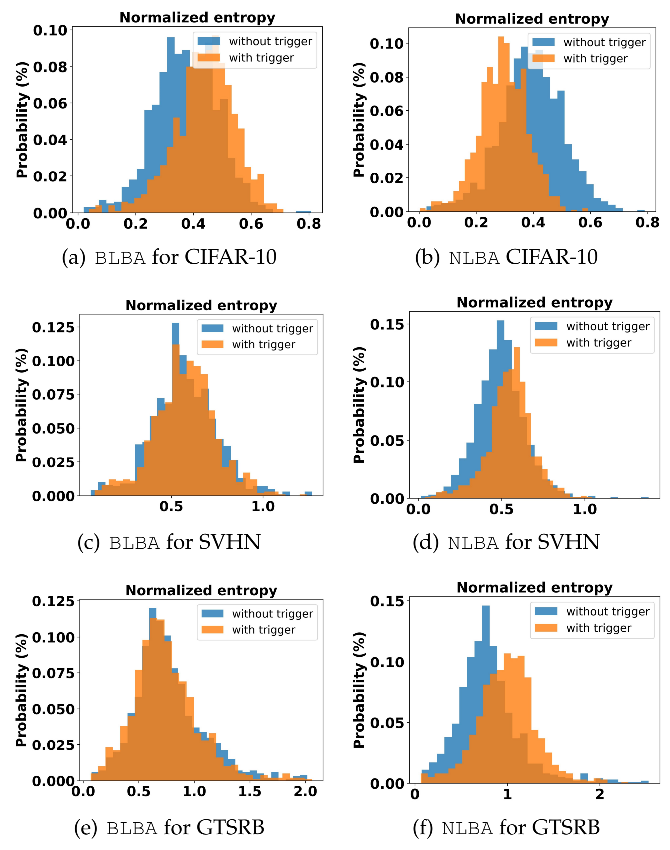


(a) BLBA for CIFAR-10 (b) NLBA for CIFAR-10

(c) BLBA for SVHN (d) NLBA SVHN

(e) BLBA for GTSRB (f) NLBA for GTSRB

Fig. 6: Defeating *STRIP* (teacher model)



(a) BLBA for CIFAR-10 (b) NLBA CIFAR-10

(c) BLBA for SVHN (d) NLBA for SVHN

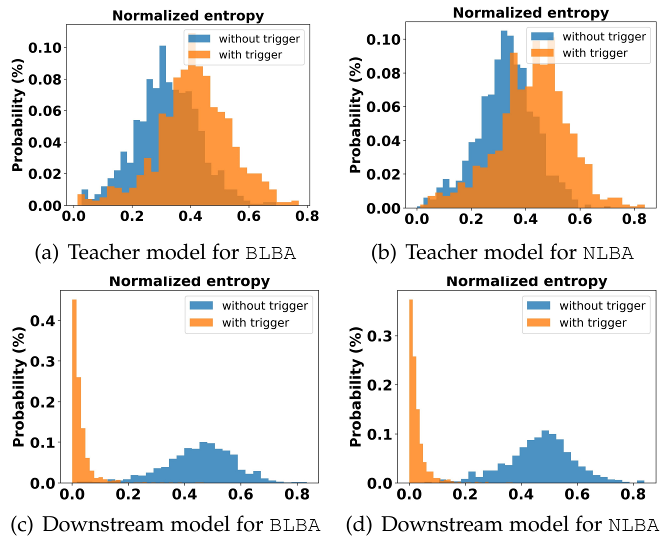
(e) BLBA for GTSRB (f) NLBA for GTSRB

Fig. 7: Defeating *STRIP* (downstream model)

well, which weakens the attack effect for triggered samples. Besides, the performance of the model over clean samples cannot be maintained without clean samples.

Second, we evaluate the impact of the random triggered samples. We find that our attacks can achieve similar attack effectiveness without this type of samples. However, it can be easily mitigated by the considered defenses. For instance, Fig. 8 shows the detection results of *STRIP* when the random triggered samples are not used. We can see the backdoor in the teacher model is still hidden. In the downstream model, the entropy distributions are quit distinct, enabling *STRIP* to identify triggered samples. In Figure 9, we show the anomaly scores of the backdoor attack without the random triggered samples by *Neural Cleanse*. We observe that for both teacher and downstream models, the backdoored model is more suspicious to be detected.

To summarize, the triggered and clean samples are used to ensure the effectiveness of the backdoor attack. The random triggered samples are critical for enhancing the stealthiness of the attack against different defense mechanisms. They are all indispensable for building satisfactory backdoor attacks.



(a) Teacher model for BLBA (b) Teacher model for NLBA

(c) Downstream model for BLBA (d) Downstream model for NLBA

Fig. 8: Defeating *STRIP* without random triggered samples

7 Discussion and Future Work

7.1 Potential defense

We have demonstrated that existing backdoor defense mechanisms are ineffective in defeating NLBA. In this subsection, we investigate the process of NLBA and discuss potential defenses against it. One observation is that NLBA manipulates some target neurons and adjusts relevant weights to achieve the attack. In order to preserve the old knowledge of the model, most of the CIL approaches take the original model parameters and add neurons to adapt to their new tasks. Since the poisoned target neurons do not contribute to the classification result, these weights remain essentially unchanged during the subsequent class incremental learning process. Based on this observation, a possible defense against NLBA is to re-initialize the weights of the classifica-

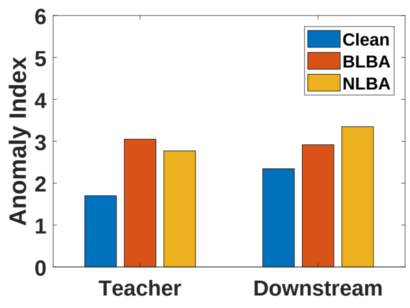


Fig. 9: Defeating *Neural Cleanse* without random triggered samples

tion layer before performing class incremental learning. Re-initialization of the classification layer disrupts the step of weight adjustment, making the backdoor attack ineffective. However, the parameters of the teacher model represent the old knowledge of the classification task, which is expected to be retained during incremental learning process. Re-initialization of the classification layer will greatly reduce the performance of the downstream model on the original categories, especially when the training samples of these categories are not available during incremental learning. Even if these samples are available, re-initialization will make the model forget the old knowledge, and have to relearn it, which significantly affect the efficiency of incremental learning. In the future, we intend to combine this re-initialization method with incremental learning approaches to evaluate the effectiveness of this defense, and also propose more effective re-initialization approaches.

7.2 Limitations

Our proposed attack has one limitation: the adversary can only restrict the adversarial label of the triggered sample to one of the newly learned classes. When the teacher model incrementally learns more classes, the adversary cannot control which new class is the target. Target backdoor attack is hard to achieve when the adversary does not know the downstream tasks. One possible solution is that we can implant multiple different triggers into the teacher model, which gives us a higher probability to find a trigger that directs the malicious samples to the desired class [26]. We leave this as future work. Interestingly, due to the characteristics of `NLBA`, we find the target label in the case of learning multiple new classes is the class with the largest weight that connects the poisoned neuron and the output layer.

8 Conclusions

For the first time, we propose a latent backdoor attack against the popular class incremental learning scheme. An adversary can implant a backdoor into a pre-trained teacher model, which will be alive in arbitrary downstream models developed from this teacher model via CIL. We propose two sophisticated attack techniques to achieve this goal when the adversary does not have the knowledge of the downstream tasks. We conduct extensive experiments to prove our proposed attacks are effectiveness and robust. They can easily bypass existing state-of-the-art backdoor detection methods. Therefore, it is urgent to design new effective approaches to mitigate these severe attacks, which will serve as our future work.

Acknowledgment

This work is supported by the Key-Area Research and Development Program of Guangdong Province under Grant 2020B0101360001, National Natural Science Foundation of China under Grants 62020106013, 61972454, and 61802051, Sichuan Science and Technology Program under Grants 2020JDTD0007 and 2020YFG0298, the Fundamental Research Funds for Chinese Central Universities under Grant ZYGX2020ZB027, Singapore Ministry of Education (MOE) AcRF Tier 2 MOE-T2EP20121-0006.

References

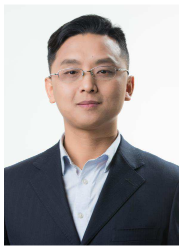
[1] J. Guo, X. Zhu, C. Zhao, D. Cao, Z. Lei, and S. Z. Li, “Learning meta face recognition in unseen domains,” in *Proceedings of CVPR*, 2020, pp. 6163–6172.
[2] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, “Verifynet: Secure and verifiable federated learning,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 911–926, 2019.
[3] G. Xu, H. Li, Y. Zhang, S. Xu, J. Ning, and R. Deng, “Privacy-preserving federated deep learning with irregular users,” *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 2, pp. 1364–1381, 2022.
[4] M. Tan, R. Pang, and Q. V. Le, “Efficientdet: Scalable and efficient object detection,” in *Proceedings of CVPR*, 2020, pp. 10 781–10 790.
[5] G. G. Chowdhury, “Natural language processing,” *Annual review of information science and technology*, vol. 37, no. 1, pp. 51–89, 2003.
[6] “Model zoo: Discover open source deep learning code and pretrained models.” [Online]. Available: https://modelzoo.co/
[7] “Hugging face - the ai community building the future.” [Online]. Available: https://huggingface.co/
[8] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. van de Weijer, “Class-incremental learning: survey and performance evaluation on image classification,” *arXiv preprint arXiv:2010.15277*, 2020.
[9] Z. Li and D. Hoiem, “Learning without forgetting,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.
[10] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, “End-to-end incremental learning,” in *Proceedings of ECCV*, 2018, pp. 233–248.
[11] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, “icarl: Incremental classifier and representation learning,” in *Proceedings of CVPR*, 2017, pp. 2001–2010.
[12] Z. Chen and B. Liu, “Lifelong machine learning,” *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 12, no. 3, pp. 1–207, 2018.
[13] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, “Lifelong learning with dynamically expandable networks,” *arXiv preprint arXiv:1708.01547*, 2017.
[14] G. Xu, H. Li, H. Ren, K. Yang, and R. H. Deng, “Data security issues in deep learning: attacks, countermeasures, and opportunities,” *IEEE Communications Magazine*, vol. 57, no. 11, pp. 116–122, 2019.
[15] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, “Badnets: Evaluating backdooring attacks on deep neural networks,” *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.
[16] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, “Targeted backdoor attacks on deep learning systems using data poisoning,” *arXiv preprint arXiv:1712.05526*, 2017.
[17] Y. Li, Y. Li, B. Wu, L. Li, R. He, and S. Lyu, “Invisible backdoor attack with sample-specific triggers,” in *Proceedings of ICCV*, 2021, pp. 16 463–16 472.
[18] A. Turner, D. Tsipras, and A. Madry, “Label-consistent backdoor attacks,” *arXiv preprint arXiv:1912.02771*, 2019.

[19] S. Zhao, X. Ma, X. Zheng, J. Bailey, J. Chen, and Y.-G. Jiang, "Clean-label backdoor attacks on video recognition models," in *Proceedings of CVPR*, 2020, pp. 14 443–14 452.
[20] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, "Latent backdoor attacks on deep neural networks," in *Proceedings of CCS*, 2019, pp. 2041–2055.
[21] Y. Ji, Z. Liu, X. Hu, P. Wang, and Y. Zhang, "Programmable neural network trojan for pre-trained feature extractor," *arXiv preprint arXiv:1901.07766*, 2019.
[22] 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 Transactions on Services Computing*, 2020.
[23] L. Li, D. Song, X. Li, J. Zeng, R. Ma, and X. Qiu, "Backdoor attacks on pre-trained models by layerwise weight poisoning," *arXiv preprint arXiv:2108.13888*, 2021.
[24] J. Jia, Y. Liu, and N. Z. Gong, "Badencoder: Backdoor attacks to pre-trained encoders in self-supervised learning," *arXiv preprint arXiv:2108.00352*, 2021.
[25] X. Zhang, Z. Zhang, S. Ji, and T. Wang, "Trojaning language models for fun and profit," in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE Computer Society, 2021, pp. 179–197.
[26] L. Shen, S. Ji, X. Zhang, J. Li, J. Chen, J. Shi, C. Fang, J. Yin, and T. Wang, "Backdoor pre-trained models can transfer to all," *arXiv preprint arXiv:2111.00197*, 2021.
[27] K. Chen, Y. Meng, X. Sun, S. Guo, T. Zhang, J. Li, and C. Fan, "Badpre: Task-agnostic backdoor attacks to pre-trained nlp foundation models," *arXiv preprint arXiv:2110.02467*, 2021.
[28] Z. Zhang, G. Xiao, Y. Li, T. Lv, F. Qi, Z. Liu, Y. Wang, X. Jiang, and M. Sun, "Red alarm for pre-trained models: Universal vulnerability to neuron-level backdoor attacks," *arXiv preprint arXiv:2101.06969*, 2021.
[29] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.
[30] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.
[31] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of CVPR*, 2014, pp. 580–587.
[32] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks," in *Proceedings of ECCV*, 2020, pp. 182–199.
[33] Y. Tian, F. Suya, F. Xu, and D. Evans, "Stealthy backdoors as compression artifacts," *arXiv preprint arXiv:2104.15129*, 2021.
[34] Y. Liu, Y. Xie, and A. Srivastava, "Neural trojans," in *2017 IEEE International Conference on Computer Design (ICCD)*. IEEE, 2017, pp. 45–48.
[35] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2018, pp. 273–294.
[36] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proceedings of S&P*, 2019, pp. 707–723.
[37] R. Wang, G. Zhang, S. Liu, P.-Y. Chen, J. Xiong, and M. Wang, "Practical detection of trojan neural networks: Data-limited and data-free cases," in *Proceedings of ECCV*, 2020, pp. 222–238.
[38] X. Qiao, Y. Yang, and H. Li, "Defending neural backdoors via generative distribution modeling," *arXiv preprint arXiv:1910.04749*, 2019.
[39] W. Guo, L. Wang, Y. Xu, X. Xing, M. Du, and D. Song, "Towards inspecting and eliminating trojan backdoors in deep neural networks," in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 162–171.
[40] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "Strip: A defence against trojan attacks on deep neural networks," in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019, pp. 113–125.
[41] Y. Liu, S. Ma, Y. Aafer, W. C. Lee, and X. Zhang, "Trojaning attack on neural networks," in *Proceedings of NDSS*, 2017.
[42] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "Proflip: Targeted trojan attack with progressive bit flips," in *Proceedings of ICCV*, 2021, pp. 7718–7727.
[43] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proceedings of CVPR*, 2017, pp. 1765–1773.
[44] A. S. Rakin, Z. He, and D. Fan, "Tbt: Targeted neural network attack with bit trojan," in *Proceedings of CVPR*, 2020, pp. 13 198–13 207.
[45] S. Li, M. Xue, B. Zhao, H. Zhu, and X. Zhang, "Invisible backdoor attacks on deep neural networks via steganography and regularization," *IEEE Transactions on Dependable and Secure Computing*, 2020.
[46] S. Liu, E. Johns, and A. J. Davison, "End-to-end multi-task learning with attention," in *Proceedings of CVPR*, 2019, pp. 1871–1880.
[47] M. Guo, A. Haque, D.-A. Huang, S. Yeung, and L. Fei-Fei, "Dynamic task prioritization for multitask learning," in *Proceedings of ECCV*, 2018, pp. 270–287.
[48] Z. Allen-Zhu, Y. Li, and Z. Song, "A convergence theory for deep learning via over-parameterization," in *Proceedings of ICML*, 2019, pp. 242–252.
[49] E. Bagdasaryan and V. Shmatikov, "Blind backdoors in deep learning models," *arXiv preprint arXiv:2005.03823*, 2020.
[50] T. A. Nguyen and A. T. Tran, "Wanet-imperceptible warping-based backdoor attack," in *Proceedings of ICLR*, 2020.
[51] T. A. Nguyen and A. Tran, "Input-aware dynamic backdoor attack," in *Proceedings of NIPS*, vol. 33, 2020, pp. 3454–3464.

Wenbo Jiang received his B.S. degree in information security from University of Electronic Science and Technology of China (UESTC) in 2017. Currently, he is a PhD student at the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), and a joint PhD student in School of Computer Science and Engineering, at Nanyang Technological University. His research interests include adversarial machine learning and model extraction attacks.

Tianwei Zhang is an assistant professor in School of Computer Science and Engineering, at Nanyang Technological University. His research focuses on computer system security. He is particularly interested in security threats and defenses in machine learning systems, autonomous systems, computer architecture and distributed systems. He received his Bachelor's degree at Peking University in 2011, and the Ph.D degree in at Princeton University in 2017.

Han Qiu received the B.E. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2011, the M.S. degree from Telecom-ParisTech (Institute Eurecom), Biot, France, in 2013, and the Ph.D. degree in computer science from the Department of Networks and Computer Science, Telecom-ParisTech, Paris, France, in 2017. He worked as a postdoc and a research engineer with Telecom Paris and LINCS Lab from 2017 to 2020. Currently, he is an assistant professor at Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing, China. His research interests include AI security, big data security, and the security of intelligent transportation systems.



Hongwei Li (M'12-SM'18) is currently the Head and a Professor at Department of Information Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China. He received the Ph.D. degree from University of Electronic Science and Technology of China in June 2008. He worked as a Postdoctoral Fellow at the University of Waterloo from October 2011 to October 2012. He is the Senior Member of IEEE, the Distinguished Lecturer of IEEE Vehicular Technology Society.

Guowen Xu is currently a Research Fellow with Nanyang Technological University, Singapore. He received the PhD degree from the University of Electronic Science and Technology of China (UESTC) in 2020. As the first author, he has published more than 15 papers in reputable venues, including ACM ACSAC, ACM ASIACCS, IEEE TDSC and IEEE TIFS. He is the recipient of the Best Paper Award of the 26th IEEE International Conference on Parallel and Distributed Systems (ICPADS 2020).