LEARNING FROM DISTINCTION: MITIGATING BACKDOORS USING A LOW-CAPACITY MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep neural networks (DNNs) are susceptible to backdoor attacks due to their black-box nature and lack of interpretability. Backdoor attacks intend to manipulate the model's prediction when hidden backdoors are activated by predefined triggers. Although considerable progress has been made in backdoor detection and removal at the model deployment stage, an effective defense against backdoor attacks during the training time is still under-explored. In this paper, we propose a novel training-time backdoor defense method called Learning from Distinction (LfD), allowing training a backdoor-free model on the backdoor-poisoned data. LfD uses a low-capacity model as a teacher to guide the learning of a backdoor-free student model via a dynamic weighting strategy. Extensive experiments on CIFAR-10, GTSRB and ImageNet-subset datasets show that LfD significantly reduces attack success rates by 0.52%, 11.31% and 1.42%, respectively, with minimal impact on clean accuracy (less than 1%, 3% and 1%).

1 INTRODUCTION

Deep neural networks (DNNs) have achieved unprecedented success due to their remarkable performance in many applications, such as image classification(Russakovsky et al., 2015), object detection(Zou et al., 2023), and traffic accident detection(Ijjina et al., 2019; Jiansheng et al., 2014). Its superiority relies on model training on large datasets using intensive computational resources(Rosenfeld et al., 2019). In this process, if the data is obtained from third-party data sources or the model is trained on a third-party platform, the attacker is able to add predefined triggers to the data samples(Gu et al., 2017; Nguyen & Tran, 2020; Barni et al., 2019) or directly modify the model's parameters to implant a backdoor into the model(Dumford & Scheirer, 2020; Rakin et al., 2020; Chen et al., 2021). In addition, using pre-trained models or outsourced Machine Learning as a Service (MLaaS) provide attackers the opportunity to backdoor the model, though they may offer substantial performance improvements at a low cost. As a result, the attacked models perform normally on clean data, whereas misclassifying specific data to the target class when predefined triggers are activated, which poses a severe security threat to DNN applications(Gu et al., 2017; Tran et al., 2018).

To alleviate this threat, existing defense methods can be categorized into two aspects: backdoor detection methods(Tran et al., 2018; Goldblum et al., 2022; Gao et al., 2019) and backdoor removal methods(Kolouri et al., 2020; Wang et al., 2020; Chen et al., 2019b). Backdoor detection methods aim to identify whether a given model contains a backdoor trigger or whether a given dataset contains poisoned data. Backdoor removal methods are designed to remove backdoors during or after the model training process while preserving the model performance. Although the after-training removal methods have shown promising defense results, their effectiveness relies on the availability of clean data, rendering them unsuitable for scenarios where clean data is not accessible(Liu et al., 2018; Wu & Wang, 2021). Training-time removal methods aim to detect potential poisoned data during the training of the model and prevent the model from learning from these poisoned data so as to obtain a backdoor-free model(Levine & Feizi, 2021; Jia et al., 2021; Li et al., 2021). Such methods implemented during training often involve utilizing additional datasets to identify poisoned data(Tran et al., 2018; Hayase et al., 2021a) or training multiple models ensemble(Goldblum et al., 2022; Levine & Feizi, 2021; Jia et al., 2021), often suffering from significant computational costs. Recently, Li et al. proposed a two-stage unlearning technique (ABL) that allows for training a clean model on a poisoned dataset(Li et al., 2021). However, this method presents challenges in terms of inaccurate data isolation and reduced accuracy of purified model. Currently, effective defense methods against backdoor attacks during the training phase remain a crucial problem and need further exploration.

To fulfill the requirement of effectively mitigating backdoors during training, this study proposes a novel training-time defense method named Learning from Distinction (LfD), which allows training a backdoor-free model on backdoor-poisoned data. Specifically, LfD has two stages. In the first stage, it fine-tunes a low-capacity model on the backdoor-poisoned dataset for a few epochs and then utilizes this model as a teacher model to supervise the training of a backdoor-free student model. In the second stage, a dynamic weighting strategy is adopted during training to flexibly select the poisoned data from the poisoned datasets. We theoretically analyze the reason why low-capacity models can act as the teacher to help with backdoor defenses. Empirical results across various datasets and model architectures demonstrate that our proposed LfD method achieves superior performance compared to existing state-of-the-art defenses.

In summary, our contributions are as follows:

- We present the impact of model capacity on the distinction of the loss value between backdoored and clean data during training. Utilizing a lower capacity model makes the distinction of the loss value between backdoored and clean data more prominent, which can contribute to identifying the poisoned data.
- We propose a novel training-time backdoor defense method called Learning from Distinction (LfD), which allows training a clean student model on a backdoored dataset with the help of a low-capacity teacher model.
- We conduct extensive experiments on CIFAR-10, GTSRB and ImageNet-subset datasets to validate the effectiveness of LfD against 9 well-known backdoor attacks. The results show that LfD outperforms existing state-of-the-art defenses and significantly reduces attack success rate with minor impact on clean accuracy.

2 Related Work

2.1 Backdoor Attacks

Existing backdoor attacks can be classified into two main categories: (1) **Dirty-label attacks** insert trigger patterns into data and change their labels to a specific target label, associating the trigger with the target label. Gu et al. first introduced backdoor attacks in deep learning(Gu et al., 2017). Subsequently, several studies focused on making poisoned data indistinguishable from clean data to evade human inspection, some of them replace triggers with subtle perturbations(Turner et al., 2019; Zhong et al., 2020; Li et al., 2020a; Moosavi-Dezfooli et al., 2017), and others disperse the trigger to a much larger area of the image(Liu et al., 2020; Barni et al., 2019). However, in the above attacks, the source labels of the poisoned data are still different from the target label, making them detectable by examining the image-label relationship. (2) **Clean-label attacks** only modify the data whose source labels are the same as the target label. Hence, they do not need to change the data's label, making them more covert compared to dirty-label attacks. Several studies focus on modifying data and adding easy-to-learn trigger patterns on them to perform clean-label attacks(Turner et al., 2019; Barni et al., 2019). Nevertheless, compared to dirty-label attacks, clean-label attacks often suffer from lower attack success rate(Li et al., 2022; Zhao et al., 2020b).

2.2 Backdoor defenses

Recently, numerous defense methods have been proposed to mitigate the threat of backdoor attacks. Existing defense methods can be broadly categorized into two categories based on when they take effect: (1) The defense methods adopted during the model deployment phase aim to detect whether the input data (Tran et al., 2018; Gao et al., 2019; Chen et al., 2019a; Xu et al., 2021; Hayase et al., 2021b; Tang et al., 2021) or the model (Wang et al., 2019; Chen et al., 2019b; Kolouri et al., 2020; Shen et al., 2021) has been backdoor-poisoned and to remove the existing backdoor in the backdoor-poisoned model (Zhao et al., 2020a; Li et al., 2020b; Wu & Wang, 2021). The anomaly input detection methods employ the concept of outlier detection to filter out backdoor-poisoned data (Du et al., 2019; Gao et al., 2019; Subedar et al., 2019; Jin et al., 2020; Javaheripi et al., 2020), while the backdoored model detection methods utilize meta-classifiers (Kolouri et al., 2020; Huang et al., 2019; Xu et al., 2021) or analyze the model's internal structure (Liu et al., 2019; Zheng et al., 2021) to identify the presence of backdoors. Unlike detection methods, removal methods' purpose is to eliminate the impact of backdoor attacks. Some removal methods reconstruct the trigger and employ fine-tuning to repair the model(Wang et al., 2019; Chen et al., 2022; Dong et al., 2021; Guan et al., 2022; Tao et al., 2022). Apart from those trigger-reconstruct defenses, other approaches have been widely applied in removal backdoors, e.g., pruning(Wu & Wang, 2021; Liu et al., 2018; Zheng et al., 2022) and model distillation(Li et al., 2020b). (2) **Training-time backdoor defense methods** intend to train backdoor-free models on backdoor-poisoned datasets. Some approaches achieve this by extracting features of clean data from additional datasets to identify the poisoned data in the training dataset(Tran et al., 2018; Hayase et al., 2021a). Other approaches, on the other hand, focus on training multiple model ensembles and applying voting mechanisms to mitigate the impact of poisoned data(Goldblum et al., 2022; Levine & Feizi, 2021; Jia et al., 2021). Recently, Anti-backdoor learning (ABL) isolates a portion of poisoned data and unlearns these data in the last few epochs of training to eliminate the backdoor(Li et al., 2021). However, unlearning can lead to the loss of semantic features and a decrease in the accuracy of the model. Our proposed defense method LfD belongs to the training-time backdoor defense method.

3 Background and preliminary analysis

In this section, we first present the utilization of gradient ascent training for backdoor defense. Subsequently, we delineate our observations regarding distinct learning behaviors between the losses of poisoned and clean data . Finally, we propose a method to amplify this distinction.

3.1 Backdoor defense with gradient ascent

In the backdoor attack scenario, the goal of the adversary is to inject triggers into the model by solving an optimization problem as follows:

$$L(\theta_t) = \mathbb{E}_{(x,y)\sim D_{poisoned}}(\ell(f_{\theta_t}(x), y)) + \mathbb{E}_{(x,y)\sim D_{clean}}(\ell(f_{\theta_t}(x), y)), \tag{1}$$

where t represents the number of training epochs, D_{clean} and D_{poison} represent the set of clean data and poisoned data, respectively, $D_{clean} \cup D_{poison} = D_{train}$, and $\ell(\cdot)$ denotes the loss function which computes the distance between the predicted label $f_\theta(x)$ and the ground truth label y. Then, the minimum value of the loss is calculated by using the gradient descent optimization method, thereby improving model's classification accuracy. This equation indicates that the entire learning task is decomposed into two sub-tasks: one is to minimize f_θ on clean data, and the other is to minimize f_θ on poisoned data. Therefore, the trained backdoored model exhibits high classification accuracy on clean data and high attack success rate on poisoned data.

To mitigate the impact of backdoor attacks on the model, the gradient ascent training method can be employed to alleviate the influence of poisoned data (Li et al., 2021). The process is as follows:

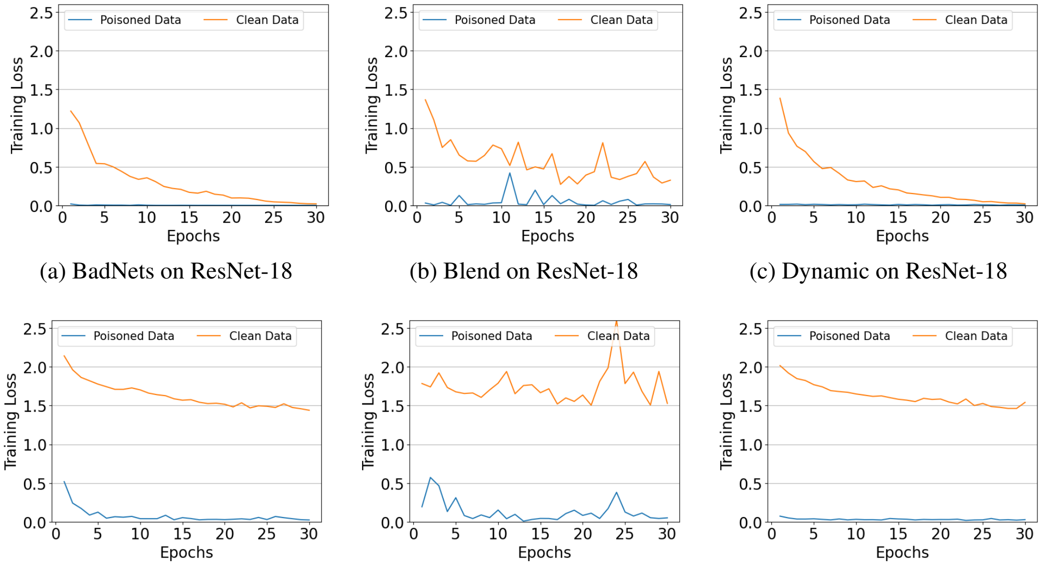
$$L(\theta_t) = \mathbb{E}_{(x,y)\sim D_{clean}}(\ell(f_{\theta_t}(x), y)) - \mathbb{E}_{(x,y)\sim D_{poisoned}}(\ell(f_{\theta_t}(x), y)), \tag{2}$$

As shown in the above equation, maximizing the loss of poisoned data can effectively amplify the distance between model predictions and their labels, i.e., target labels. This prevents the model from classifying poisoned data as the target label, thus reducing the model's attack success rate. Therefore, if we can devise a method for precisely identifying poisoned data within the dataset, we can integrate it with the gradient ascent training approach to mitigate the impact of backdoor attacks on the model.

3.2 Distinct Behaviors In Learning Between clean And Poisoned Data.

Previous studies have identified poisoned data within datasets through the analysis of data loss. This is owing to the nature of backdoor attacks, which require the establishment of explicit associations between triggers and target labels to ensure that poisoned data is classified into the target label rather than its ground truth label. Consequently, poisoned data is found to be more easily learnable than clean data, resulting in lower losses(Li et al., 2021). To demonstrate this inference, we apply three classic backdoor attacks, named BadNets(Gu et al., 2017), Dynamic(Nguyen & Tran, 2020), and SIG(Barni et al., 2019) to construct three poisoned datasets with an injection rate of 10% on CIFAR-10 training data. Subsequently, we train the ResNet-18 model(He et al., 2016) on these poisoned datasets following the same configurations in Section 5. As shown in Figure 1a, 1b, 1c, for all three types of backdoor attacks, the average loss of clean data is higher than that of poisoned data, and this trend is more significant in the early stages of training. The above observations suggest that the loss of poisoned data is usually lower than that of clean data, especially in the early stage.

However, this method of identifying poisoned data has its shortcomings. Despite the significant difference in average losses between poisoned and clean data, there is still a possibility of poisoned data having losses similar to that of clean data. In Figure 2, we plot the number of poisoned data with losses greater than the average loss of clean data at the early training stage on the ResNet-18 model, and it is evident that such poisoned data exist in three poisoned datasets. Since some powerful backdoor attacks can succeed with just a tiny number of poisoned data(Li et al., 2021), it is necessary to employ a strategy that amplifies the distinction in the losses of these two types of data in order to accurately identify the poisoned data in the dataset.



(a) BadNets on ResNet-18

(b) Blend on ResNet-18

(c) Dynamic on ResNet-18

(d) BadNets on ResNet-S

(e) Blend on ResNet-S

(f) Dynamic on ResNet-S

Figure 1: The upper and bottom rows respectively present the average training losses on ResNet-18 (He et al., 2016) and ResNet-S for clean and poisoned data. The poisoned data is generated through three different backdoor attacks, including BadNets (Gu et al., 2017), Blend (Nguyen & Tran, 2020), and Dynamic (Barni et al., 2019). The experiments were conducted on the CIFAR-10 dataset(Krizhevsky, 2009) with a poisoning rate of 10%.

3.3 Impact Of Model Capacity On Data Loss

The capacity of a neural network model signifies the complexity of features it can learn. A model's capacity can be measured through its intricacy, such as the quantity of neurons of the model. Based on prior research, the generalization error of neural networks can be approximately formalized to a function of the model and dataset size (Rosenfeld et al., 2019). Given a fixed dataset size, the following equation holds:

$$\hat{\epsilon}(m) = \epsilon_0 \left\| \frac{bm^{-\beta} + c_\infty}{bm^{-\beta} + c_\infty - i\eta} \right\|, \tag{3}$$

where $\hat{\epsilon}$ denotes the classification error on the test dataset, m denotes the number of parameters in the model, $\beta(\beta \geq 0)$ control the global rate at which error decreases with the sizes of the model, $i = \sqrt{-1}$, and $c_\infty(c_\infty > 0)$ is the asymptotic lower value attainable. Here the simple pole at η controls the transition point from the initial random-guess level ϵ_0 as (m) increase.

The above equation showed that the classification error increases as the number of model parameters decreases. This implies that the model's capability to accurately classify data decreases with a reduction in its capacity. Moreover, since the loss of data measures the classificatuon accurcy

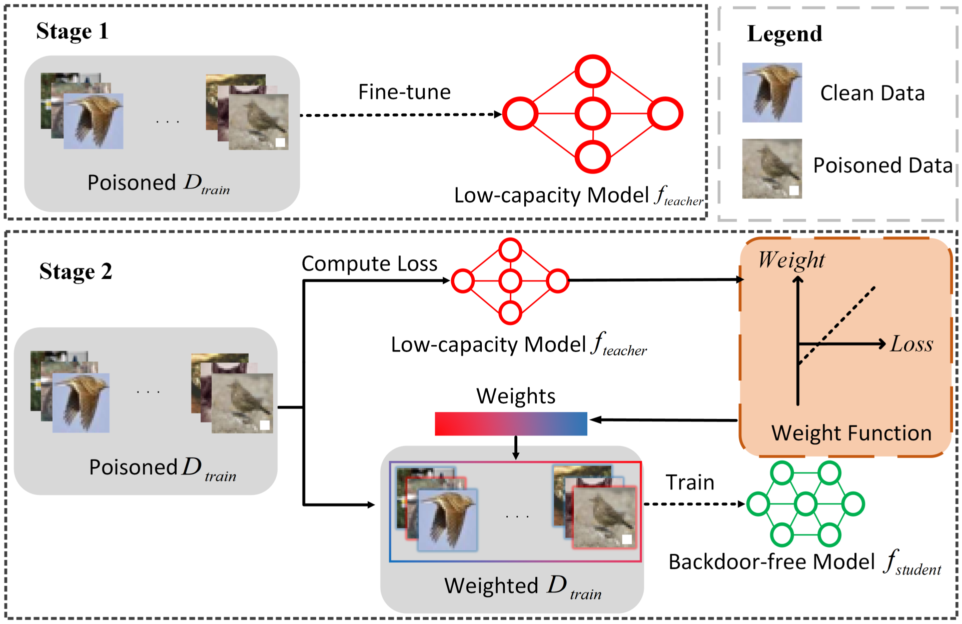


Figure 3: Pipeline of our LfD. It consists of two main stages. (1) Fine-tuning a low-capacity $f_{\theta_{teacher}}$ on the backdoor-poisoned D_{train} for a few epochs. (2) Weighting each data in D_{train} according to its loss on $f_{\theta_{teacher}}$, and then training $f_{\theta_{student}}$ on the weighted D_{train}.

of the model, the data's losses on the model usually rise with a reduction in model capacity. Simultaneously, according to Section 3.2, poisoned data is more easily-learnable compared to clean data. Therefore, we claim that reducing the model's capacity could render it less adept at learning clean data while retaining its capability to learn poisoned data. This would effectively amplify the disparity in loss between the two types of data. To substantiate this point, we construct a ResNet-based model with fewer parameters than ResNet-18, referred to as ResNet-S, described in detail in Appendix A.3, as a comparative model to ResNet-18 in order to observe how the difference in the average loss of clean and poisoned data during model training varies with the model capacity.

As depicted in Figure 1, across various backdoor attack scenarios, the average loss of clean data on ResNet-S is notably higher than that on ResNet-18. However, the average loss of poisoned data exhibits no significant disparity between these two models. The above findings validate our proposition that, despite the difficulty of ResNet-S in learning clean data due to its low capacity, it still has the ability to learn poisoned data, which leads to a more pronounced distinction in the average loss between the two types of data on ResNet-S.

Following the above results, we test the number of poisoned data with losses greater than the average loss of clean data on ResNet-18 and ResNet-S. We conducted experiments on ResNet-18 and ResNet-S to evaluate the quantity of poisoned data with losses greater than the average loss of clean data. As depicted in Figure 2, the number of such data instances on ResNet-S is notably reduced compared to ResNet-18, thereby further confirming the conclusions drawn in the preceding discussion.

Furthermore, we constructed low-capacity models on several other architectural frameworks (as detailed in Appendix A.3) for additional exploration. Specific results are presented in Appendix A.4.

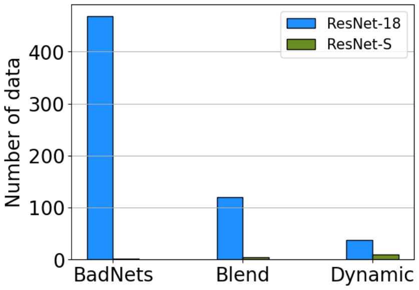


Figure 2: The number of poisoned data with losses higher than the average loss of clean data on ResNet-18 and ResNet-S at the fifth epoch.

4 Proposed Method: Learning from Distinction

Based on the observations and analyses in Section 3, we propose a Learning from Distinction (LfD) approach which involves training a clean model on the poisoned data set D_{train} within the context of a supervised training process. As illustrated in Figure 3, we decompose the entire training process into two stages: fine-tuning a teacher model $f_{\theta_{teacher}}$ capable of discerning the differences between poisoned and clean data, and using $f_{\theta_{teacher}}$ to dynamically discriminate poisoned data within D_{train} to supervise the training of the clean model $f_{\theta_{student}}$.

Stage 1: Fine-tuning a $f_{\theta_{teacher}}$ capable of distinguishing between clean and poisoned data sets.

First and foremost, we require a $f_{\theta_{teacher}}$ capable of distinguishing between clean and poisoned data within D_{train}. Drawing insights from the observations described in Section 3, it becomes evident that models with low capacity face challenges in classifying clean data. Nevertheless, these models still exhibit an intrinsic aptitude for adeptly classifying poisoned data, especially during the initial phases of training. This phenomenon engenders a noticeable discrepancy in losses between these two data categories. Consequently, we embark on creating a low-capacity model by reducing the number of neurons within the model, followed by subjecting it to a few epochs of fine-tuning solely on D_{train}. This maneuver ensures a pronounced dissimilarity between poisoned and clean data on this model. Subsequently, this model assumes a pivotal role in the context of Learning from Distinction (LfD), functioning as the cornerstone of $f_{\theta_{teacher}}$.

Stage 2: Utilizing $f_{\theta_{teacher}}$ for the dynamic discrimination of poisoned data to supervise the training process of a backdoor-free model $f_{\theta_{student}}$.

Subsequently, we identify potential poisoned data within D_{train} based on their losses on $f_{\theta_{teacher}}$, followed by gradient ascent training on these identified data to train a clean $f_{\theta_{student}}$ on D_{train}. This stage aims to mitigate the impact of poisoned data in the training process of $f_{\theta_{student}}$. To identify the poisoned data within D_{train}, we initially adaptively establish a threshold based on the losses of data in D_{train} on $f_{\theta_{teacher}}$:

$$\begin{aligned}&\gamma = DT(((f_{\theta_{teacher}}(x_1), y_1) \cdots (f_{\theta_{teacher}}(x_n), y_n)), index),\\ &\mathrm{s.t.}\, index = n \times \alpha,\end{aligned} \tag{4}$$

where n represents the number of data in D_{train}, and α is a hyperparameter, the DT() function first calculates the losses of all data points in D_{train} with respect to $f_{teacher}$, then sorts the losses in ascending order, and selects the loss at the $index$-th position as the threshold. We perform gradient ascent training on the data in D_{train} with losses below this threshold. Specifically, as shown in Equation 1, when updating $f_{\theta_{student}}$, we subtract the losses of these data points to decrease the probability of $f_{\theta_{student}}$ classifying them into their labels. Though we cannot guarantee that the loss of every poisoned data in D_{train} is below the threshold, training a substantial portion of the poisoned data with gradient ascent can still mitigate the impact of these unrecognized poisoned data. Therefore, the loss function employed for updating $f_{\theta_{student}}$ is presented as follows:

$$\begin{aligned}&L(\theta_{student}) = \mathbb{E}_{(x,y)\sim D_{non-candidate}}(\ell(f_{\theta_{teacher}}(x), y)) - \mathbb{E}_{(x,y)\sim D_{candidate}}(\ell(f_{\theta_{teacher}}(x), y)),\\ &\mathrm{s.t.} (x, y) \in \begin{cases} D_{non-candidate}, & \ell(f_{\theta_{teacher}}(x), y) > \gamma) \\ D_{candidate}, & \ell(f_{\theta_{teacher}}(x), y) \leq \gamma), \end{cases}\end{aligned} \tag{5}$$

However, the gradient ascent training method may have an impact on the accuracy of $f_{\theta_{student}}$ on clean data as we presented in Section 5.2. We posit that this phenomenon arises due to the some clean data instances have losses below the threshold, and performing gradient ascent training on these data reduces the classification accuracy of $f_{\theta_{student}}$. Therefore, we introduce a weighting method that dynamically assigns weights to data to determine their significance during training, thereby mitigating the impact of gradient ascent training on the accuracy of $f_{\theta_{student}}$. The weight for each data in D_{train} is computed as follows:

$$\begin{aligned}&L(\theta_{student}) = \mathbb{E}_{(x,y)\sim D_{train}}(w_{(x,y)} \cdot \ell(f_{\theta_{teacher}}(x), y))),\\ &\mathrm{s.t.}\, w_{(x,y)} = \begin{cases} \frac{\ell(f_{\theta_{teacher}}(x), y)}{\max_{i=1}^{n}(\ell(f_{\theta_{teacher}}(x_i), y_i))}, & \ell(f_{\theta_{teacher}}(x), y) > \gamma) \\ \frac{\beta\cdot(\ell(f_{\theta_{teacher}}(x), y) - \gamma)}{\max_{i=1}^{n}(\ell(f_{\theta_{teacher}}(x_i), y_i))}, & \ell(f_{\theta_{teacher}}(x), y) \leq \gamma), \end{cases}\end{aligned} \tag{6}$$

where the weight of data (x, y) is denoted as $w_{(x,y)}$ in the equation, the hyperparameter β determines the importance of gradient ascent training relative to gradient descent training. Different from equation 5, equation 6 calculates $L(\theta_t)$ by considering the weight of each data point based on its loss on $f_{\theta_{teacher}}$. Compared to the solely unweighted gradient ascent training, even when some clean data possess losses below the threshold and are assigned negative weights, their relatively lower losses compared to most poisoned data result in lower absolute values of their negative weights. Consequently, the impact on the clean accuracy of $f_{\theta_{student}}$ on clean data is limited.

5 Experiment

We evaluated our LfD defense method against seven dirty-label backdoor attacks and two clean-label backdoor attacks on CIFAR-10(Krizhevsky, 2009), GTSRB(Stallkamp et al., 2012), and an ImageNet subset(Deng et al., 2009). Specifically, dirty-label attacks include BadNets with white square pattern (referred to as BN-W)(Gu et al., 2017), BadNets with grid square pattern (referred to as BN-G)(Gu et al., 2017), Trojan attack with square pattern (referred to as TJ-SQ)(Liu et al., 2017), Trojan attack with watermark pattern (referred to as TJ-WM)(Liu et al., 2017), $l2$-Invisible attack (referred to as L2)(Li et al., 2020a), Dynamic attack (Nguyen & Tran, 2020), Blend attack (Chen et al., 2017), and clean-label attacks include Sinusoidal signal attack (referred to as SIG) (Barni et al., 2019) and Clean-Label attack (referred to as CL) (Turner et al., 2018). Detailed configurations of each backdoor attack are provided in Appendix A.1.

On the CIFAR-10 and ImageNet-subset dataset, we set $\alpha = 0.15$ and $\beta = 1e - 3$, while on the GTSRB dataset, we set $\alpha = 0.2$ and $\beta = 3e - 3$. We employed ResNet-S as $f_{\theta_{teacher}}$ across all three datasets and utilized ResNet-18 as $f_{\theta_{student}}$ on CIFAR-10 and GTSRB, while on the ImageNet-subset, we employed ResNet-34 as $f_{\theta_{student}}$. During the fine-tuning of $f_{\theta_{teacher}}$, no data augmentation is applied. For the training of $f_{\theta_{student}}$, we employ random cropping and random horizontal flipping as data augmentation. We compare LfD with three state-of-the-art defense methods: Fine-pruning (FP)(Liu et al., 2018), Activation-Cluster (AC)(Chen et al., 2019a), and Anti-Backdoor Learning (ABL)(Li et al., 2021). For FP, AC, and ABL, we follow the configurations specified in their original papers, including the use of available clean data for fine-tuning/repair/training settings. Further defense details can be found in the appendix A.2.

Metrics We employ two metrics to evaluate the effectiveness of LfD. The first one is Clean Accuracy (CA), which measures the accuracy of $f_{\theta_{student}}$ in classifying clean test data. The second metric is Attack Success Rate (ASR), which quantifies the probability of $f_{\theta_{student}}$ classifying poisoned test data as the target label.

5.1 Effectiveness of Our LfD Defense

Table 1 presents the LfD results on the CIFAR-10, GTSRB, and ImageNet-subset datasets. Evidently, our LfD demonstrates optimal performance in reducing ASR while maintaining a high CA in most backdoor attacks. Compared to the best-performing defense method ABL, our LfD reduces the average ASR by 7.68% (0.52% vs. 8.20%), 7.86% (11.31% vs. 19.17%), and 36.10% (1.42% vs. 37.52%) on CIFAR-10, GTSRB, and the ImageNet-subset datasets, respectively. This advantage becomes more pronounced compared to other defense methods.

We note that our LfD is not always the best when considering each attack individually. For instance, ABL exhibits the best defense against TJ-WM and Dynamic attacks on GTSRB. We suspect that this is due to the similarity between the triggers used in these two attacks and the feature of normal images. We suspect this is because even clean data may contain patterns similar to the trigger, making the detection of poisoned data more challenging (Zhao et al., 2021). If LfD fails to identify the majority of poisoned data, its defense effectiveness will be weakened. This limitation represents one of the shortcomings of our LfD approach and requires further improvement in future work.

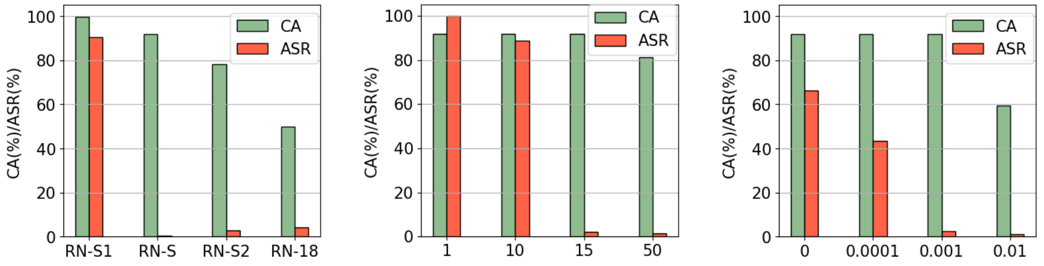
Maintaining CA and reducing ASR are equally important, as significant degradation of CA suggests difficulties in correctly classifying clean data, rendering the method impractical. By examining the results in Table 1, we can observe that our LfD achieves higher CA than other defense methods in most scenarios. Specifically, our LfD outperforms ABL in terms of average CA on CIFAR-10, GTSRB, and ImageNet-subset by 5.56% (92.09% vs. 86.53%), 4.79% (93.42% vs. 88.63%), and 4.91% (90.97% vs. 86.06%), respectively. This demonstrates the practicality of our LfD defense for industrial applications where performance and security are equally important.

Table 1: Defense effectiveness of various defense methods on different backdoor attack methods. In the table, “No Defense” means no attack is applied, and “None” means no defense is applied.

Dataset	Attack Type	No Defense		Fine-prune		AC		ABL		LfD	
		CA	ASR	CA	ASR	CA	ASR	CA	ASR	CA	ASR
CIFAR-10	BN-G	91.51%	100%	85.22%	99.98%	91.17%	100%	91.49%	0.51%	**92.62%**	**0.18%**
	BN-W	91.89%	99.95%	82.67%	98.71%	91.29%	92.57%	89.98%	3.26%	**92.01%**	**0.45%**
	Blend	92.01%	99.98%	83.12%	85.62%	91.44%	99.85%	77.57%	44.18%	**92.43%**	**3.38%**
	TJ-SQ	92.95%	100%	83.90%	66.87%	94.24%	99.96%	92.09%	0.36%	**92.17%**	**0.27%**
	TJ-WM	92.89%	100%	85.42%	61.02%	92.38%	99.32%	90.18%	0.04%	**91.76%**	**0.02%**
	Dynamic	92.70%	99.95%	90.49%	87.18%	83.32%	100%	90.71%	7.94%	**91.14%**	**0.55%**
	CL	92.73%	94.92%	84.09%	54.95%	92.17%	0.56%	83.42%	0.70%	**92.42%**	**0.64%**
	SIG	93.15%	92.16%	84.76%	76.32%	65.77%	87.37%	89.48%	6.12%	**92.41%**	**0.40%**
	L2	92.98%	99.96%	81.53%	80.57%	90.48%	97.91%	73.92%	10.71%	**91.88%**	**0.11%**
	None	92.64%	0%	88.56%	0%	**92.45%**	**0%**	73.27%	0%	91.68%	0%
GTSRB	BN-G	96.13%	100%	88.53%	99.57%	96.41%	100%	86.44%	4.02%	**95.22%**	**0.02%**
	Blend	95.42%	100%	85.90%	99.50%	96.28%	80.84%	82.09%	11.36%	**92.31%**	**3.86%**
	TJ-SQ	95.42%	99.95%	84.21%	93.54%	96.28%	99.91%	**96.25%**	**16.06%**	93.84%	25.70%
	Dynamic	96.35%	99.92%	88.38%	99.84%	95.96%	99.96%	**96.67%**	**5.57%**	88.39%	2.76%
	CL	95.88%	43.85%	89.71%	59.26%	87.06%	83.45%	81.82%	26.29%	**95.62%**	**0.02%**
	SIG	95.85%	67.87%	90.66%	78.07%	79.60%	95.79%	88.54%	4.63%	**95.17%**	**4.45%**
	None	96.90%	0%	90.14%	0%	90.46%	0%	83.95%	0%	**93.53%**	**0%**
ImageNet subset	BN-G	88.76%	100%	82.13%	43.07%	89.09%	99.83%	89.55%	3.57%	**91.19%**	**0.31%**
	BN-W	89.13%	88.71%	83.49%	48.15%	87.66%	79.01%	85.73%	31.29%	**89.09%**	**3.01%**
	Blend	91.54%	99.69%	84.79%	98.77%	90.80%	17.59%	84.44%	79.16%	**89.83%**	**2.17%**
	CL	92.41%	79.76%	82.30%	59.72%	87.94%	80.59%	83.06%	36.05%	**90.07%**	**0.17%**
	None	91.96%	0%	86.36%	0%	87.41%	0%	87.52%	0%	**94.69%**	**0%**

5.2 Ablation Studies

To gain a deeper understanding of LfD, we conducted a series of ablation experiments to elucidate the impact of hyperparameters, capacity of $f_{teacher}$, gradient ascent training, and dynamic discrimination method on LfD.



(a) Effect of $f_{\theta_{teacher}}$ (b) Effect of α (c) Effect of β

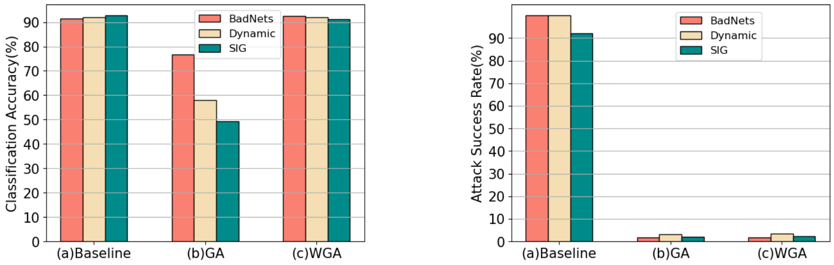
Figure 4: The influence of the capacity of $f_{\theta_{teacher}}$, along with the hyperparameters α and β, on the CA and ASR of $f_{\theta_{student}}$.

Effect of Hyperparameter $\boldsymbol{\alpha}$ The variable α controls the amount of data used in gradient ascent training. We present the performance of $f_{\theta_{student}}$ under four different settings in Figure 4b, where α takes values of 1, 10, 15, and 50, respectively. The results demonstrate that a higher α leads to a better defense against backdoor attacks in LfD, as more poisoned data are isolated during training. However, we observed that increasing α also leads to a loss in CA, as more clean data in the gradient ascent training process can affect the model's performance.

Effect of Hyperparameter $\boldsymbol{\beta}$ The variable β controls the importance of gradient ascent training relative to gradient descent training. We present the performance of $f_{\theta_{student}}$ under four different settings in Figure 4b, where β takes values of 0, 1e-4, 1e-3, and 1e-2. The results demonstrate that a higher β leads to a lower ASR as the intensity of gradient ascent training on poisoned data increases. However, we note that a high β may also result in a loss of CA, as gradient ascent training on some clean data is also intensified.

Effect of $f_{\theta_{teacher}}$'s capacity The capacity of $f_{\theta_{teacher}}$ influences the complexity of features it is capable of learning. We present in Figure 4a the CA and ASR of $f_{\theta_{student}}$ when utilizing four distinct models as $f_{\theta_{teacher}}$. The models employed are all derived from ResNet architecture,they are ResNet-S1(RN-S1)<ResNet-S(RN-S)<ResNet-S2(RN-S2)<ResNet-18(RN-18), and their detailed structures are provided in Appendix A.3. The results indicate that when the capacity of $f_{\theta_{teacher}}$ is high, its capacity to learn complex features is enhanced. This results in a significant number of clean data instances having losses similar to those of poisoned data, subsequently prompting LfD to engage in gradient ascent training on a large number of clean data. Consequently, the CA of $f_{\theta_{student}}$ diminishes. Conversely, when $f_{\theta_{teacher}}$ has a excessively low capacity, its aptitude to learn features becomes exceedingly feeble, rendering it challenging to learn clean data and the majority of poisoned data. This results in the majority of clean and poisoned data have high losses. As a result, many poisoned data is either assigned positive weights or weights close to zero. Hence, the CA and ASR of $f_{\theta_{student}}$ remain largely unaffected.

Gradient ascent training and dynamic discrimination method To aid in understanding the impact of gradient ascent training and dynamic discrimination methods in LfD, we present in Figure 5 the CA and ASR of $f_{\theta_{student}}$ trained on three D_{train} poisoned by BN-G, Blend and SIG separately, under three different scenarios with α set to 0.15: (a) training without gradient ascent, (b) training with gradient ascent while not utilizing dynamic discrimination strategy, and (c) training with gradient ascent while utilizing dynamic discrimination strategy for data weighting. Evidently, applying the gradient ascent training method can effectively reduce the ASR of $f_{\theta_{student}}$, however, training with gradient ascent on isolated data indiscriminately can also lead to a decrease in CA. Compared to the solely unweighted gradient ascent training, utilizing dynamic discrimination strategy can lead to an enhancement in CA. This indicates that the gradient ascent training method can effectively suppress the insertion of backdoors, while the dynamic discrimination method can reduce the damage to the CA of $f_{\theta_{student}}$.



(a) CA of $f_{\theta_{student}}$

(b) ASR of $f_{\theta_{student}}$

Figure 5: CA and ASR of $f_{\theta_{student}}$ trained on three D_{train} poisoned by BN-G, Blend and SIG separately under: (a) training without gradient ascent, (b) training with gradient ascent cf. Eq. 5 (GA), and (c) training with weighted gradient ascent cf. Eq. 6 (WGA).

6 Conclusion

In this work, we show the impact of model capacity on the distinction of the loss value between backdoored and clean data during training can be utilized to identify the poisoned data from a poisoned dataset. Based on this observation, we proposed a training-time backdoor defense method called Learning from Distinction (LfD), which employs a low-capacity teacher model to guide the training process of the student model by dynamically weighting the training data and thereby obtaining a clean student model. With extensive experiments, LfD demonstrated excellent performance in the robust training of neural networks against 9 state-of-the-art backdoor attacks.

REFERENCES

Mauro Barni, Kassem Kallas, and Benedetta Tondi. A new backdoor attack in cnns by training set corruption without label poisoning. In *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 101–105. IEEE, 2019.

Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. In *Workshop on Artificial Intelligence Safety*. CEUR-WSceurws@ sunsite. informatik. rwth-aachen. de, 2019a.

Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In *IJCAI*, 2019b.

Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Proflip: Targeted trojan attack with progressive bit flips. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7718–7727, 2021.

Tianlong Chen, Zhenyu Zhang, Yihua Zhang, Shiyu Chang, Sijia Liu, and Zhangyang Wang. Quarantine: Sparsity can uncover the trojan attack trigger for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 598–609, 2022.

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Yinpeng Dong, Xiao Yang, Zhijie Deng, Tianyu Pang, Zihao Xiao, Hang Su, and Jun Zhu. Black-box detection of backdoor attacks with limited information and data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16482–16491, 2021.

Min Du, Ruoxi Jia, and Dawn Song. Robust anomaly detection and backdoor attack detection via differential privacy. *arXiv preprint arXiv:1911.07116*, 2019.

Jacob Dumford and Walter Scheirer. Backdooring convolutional neural networks via targeted weight perturbations. In *2020 IEEE International Joint Conference on Biometrics (IJCB)*, pp. 1–9. IEEE, 2020.

Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pp. 113–125, 2019.

Micah Goldblum, Dimitris Tsipras, Chulin Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, Aleksander Madry, Bo Li, and Tom Goldstein. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Jiyang Guan, Zhuozhuo Tu, Ran He, and Dacheng Tao. Few-shot backdoor defense using shapley estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13358–13367, 2022.

Jonathan Hayase, Weihao Kong, Raghav Somani, and Sewoong Oh. Defense against backdoor attacks via robust covariance estimation. In *International Conference on Machine Learning*, pp. 4129–4139. PMLR, 2021a.

Jonathan Hayase, Weihao Kong, Raghav Somani, and Sewoong Oh. Spectre: Defending against backdoor attacks using robust statistics. In *International Conference on Machine Learning*, pp. 4129–4139. PMLR, 2021b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Xijie Huang, Moustafa Alzantot, and Mani Srivastava. Neuroninspect: Detecting backdoors in neural networks via output explanations. *arXiv preprint arXiv:1911.07399*, 2019.

Earnest Paul Ijjina, Dhananjai Chand, Savyasachi Gupta, and K Goutham. Computer vision-based accident detection in traffic surveillance. In *2019 10th International conference on computing, communication and networking technologies (ICCCNT)*, pp. 1–6. IEEE, 2019.

Mojan Javaheripi, Mohammad Samragh, Gregory Fields, Tara Javidi, and Farinaz Koushanfar. Cleann: Accelerated trojan shield for embedded neural networks. In *Proceedings of the 39th International Conference on Computer-Aided Design*, pp. 1–9, 2020.

Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. Intrinsic certified robustness of bagging against data poisoning attacks. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

Fu Jiansheng et al. Vision-based real-time traffic accident detection. In *Proceeding of the 11th world congress on intelligent control and automation*, pp. 1035–1038. IEEE, 2014.

Kaidi Jin, Tianwei Zhang, Chao Shen, Yufei Chen, Ming Fan, Chenhao Lin, and Ting Liu. A unified framework for analyzing and detecting malicious examples of dnn models. *arXiv preprint arXiv:2006.14871*, 8(9), 2020.

Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. Universal litmus patterns: Revealing backdoor attacks in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 301–310, 2020.

A Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Tront*, 2009.

A Levine and S Feizi. Deep partition aggregation: Provable defense against general poisoning attacks. In *International Conference on Learning Representations (ICLR)*, 2021.

Shaofeng Li, Minhui Xue, Benjamin Zi Hao Zhao, Haojin Zhu, and Xinpeng Zhang. Invisible backdoor attacks on deep neural networks via steganography and regularization. *IEEE Transactions on Dependable and Secure Computing*, 18(5):2088–2105, 2020a.

Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Neural attention distillation: Erasing backdoor triggers from deep neural networks. In *International Conference on Learning Representations*, 2020b.

Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Anti-backdoor learning: Training clean models on poisoned data. *Advances in Neural Information Processing Systems*, 34:14900–14912, 2021.

Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International symposium on research in attacks, intrusions, and defenses*, pp. 273–294. Springer, 2018.

Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. 2017.

Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1265–1282, 2019.

Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection backdoor: A natural backdoor attack on deep neural networks. In *European Conference on Computer Vision*, pp. 182–199. Springer, 2020.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1765–1773, 2017.

Tuan Anh Nguyen and Anh Tran. Input-aware dynamic backdoor attack. *Advances in Neural Information Processing Systems*, 33:3454–3464, 2020.

Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Tbt: Targeted neural network attack with bit trojan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13198–13207, 2020.

Jonathan S Rosenfeld, Amir Rosenfeld, Yonatan Belinkov, and Nir Shavit. A constructive prediction of the generalization error across scales. *arXiv preprint arXiv:1909.12673*, 2019.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Guangyu Shen, Yingqi Liu, Guanhong Tao, Shengwei An, Qiuling Xu, Siyuan Cheng, Shiqing Ma, and Xiangyu Zhang. Backdoor scanning for deep neural networks through k-arm optimization. In *International Conference on Machine Learning*, pp. 9525–9536. PMLR, 2021.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32:323–332, 2012.

Mahesh Subedar, Nilesh Ahuja, Ranganath Krishnan, Ibrahima J Ndiour, and Omesh Tickoo. Deep probabilistic models to detect data poisoning attacks. *arXiv preprint arXiv:1912.01206*, 2019.

Di Tang, XiaoFeng Wang, Haixu Tang, and Kehuan Zhang. Demon in the variant: Statistical analysis of {DNNs} for robust backdoor contamination detection. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 1541–1558, 2021.

Guanhong Tao, Guangyu Shen, Yingqi Liu, Shengwei An, Qiuling Xu, Shiqing Ma, Pan Li, and Xiangyu Zhang. Better trigger inversion optimization in backdoor scanning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13368–13378, 2022.

Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. *Advances in neural information processing systems*, 31, 2018.

Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Clean-label backdoor attacks. 2018.

Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*, 2019.

Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 707–723. IEEE, 2019.

Ren Wang, Gaoyuan Zhang, Sijia Liu, Pin-Yu Chen, Jinjun Xiong, and Meng Wang. Practical detection of trojan neural networks: Data-limited and data-free cases. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII*, pp. 222–238, 2020.

Dongxian Wu and Yisen Wang. Adversarial neuron pruning purifies backdoored deep models. *Advances in Neural Information Processing Systems*, 34:16913–16925, 2021.

Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A Gunter, and Bo Li. Detecting ai trojans using meta neural analysis. In *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 103–120. IEEE, 2021.

Pu Zhao, Pin-Yu Chen, Payel Das, Karthikeyan Natesan Ramamurthy, and Xue Lin. Bridging mode connectivity in loss landscapes and adversarial robustness. In *International Conference on Learning Representations*, 2020a.

Shihao Zhao, Xingjun Ma, Xiang Zheng, James Bailey, Jingjing Chen, and Yu-Gang Jiang. Clean-label backdoor attacks on video recognition models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14443–14452, 2020b.

Shihao Zhao, Xingjun Ma, Yisen Wang, James Bailey, Bo Li, and Yu-Gang Jiang. What do deep nets learn? class-wise patterns revealed in the input space. *arXiv preprint arXiv:2101.06898*, 2021.

Runkai Zheng, Rongjun Tang, Jianze Li, and Li Liu. Data-free backdoor removal based on channel lipschitzness. In *European Conference on Computer Vision*, pp. 175–191. Springer, 2022.

Songzhu Zheng, Yikai Zhang, Hubert Wagner, Mayank Goswami, and Chao Chen. Topological detection of trojaned neural networks. *Advances in Neural Information Processing Systems*, 34: 17258–17272, 2021.

Haoti Zhong, Cong Liao, Anna Cinzia Squicciarini, Sencun Zhu, and David Miller. Backdoor embedding in convolutional neural network models via invisible perturbation. In *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*, pp. 97–108, 2020.

Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *Proceedings of the IEEE*, 2023.

A Implementation details of Backdoor attacks and defense

In this section, we describe the experimental setup used in applying various backdoor attack and backdoor defense methods.

A.1 Attack Details

We utilized the Adam optimizer with an initial learning rate of 0.001, zero weight decay, and Random Crop (padding = 4) with Random Horizontal Flip for data augmentation when training models on the poisoned trainsets. The CosineAnnealing learning rate decay scheduler was applied for 80 training epochs. The target class for all backdoor attacks on CIFAR-10 and GTSRB datasets was uniformly designated as '1', whereas on the ImageNet-subset dataset, the target class for all backdoor attacks was consistently set to '0'. To ensure the success rate of CL and Signal attacks on the GTSRB dataset, we increased the injection rate to 9.5% instead of 8%. The specifics of the backdoor triggers are summarized in Table 2.

Table 2: Training setup for six backdoor attacks.

Attacks	Trigger Type	Trigger Patten	Inject Ratio
BN	Fixed	Grid	10%
Blend	Fixed	Random Pixel	10%
TJ	Fixed	Reversed Watermark	10%
Dynamic	Varied	Mask Generator	10%
L2	Fixed	Regularized Gaussian Noise	10%
CL	Fixed	Grid and PGD Noise	8%(other datasets) 9.5%(GTSRB)
SIG	Fixed	Sinusoidal Signal	8%(CIFAR-10) 9.5%(GTSRB)

A.2 Defense Details

For the ABL defense, we first trained the $f_{\theta_{teacher}}$ model for 20 epochs with a learning rate of 0.1 on CIFAR-10, ImageNet-subset and 0.01 on GTSRB, before reaching the turning epoch. Once we identified 1% of potential backdoor examples, we continued training the $f_{\theta_{student}}$ model for an additional 60 epochs on the entire training dataset, which helped to recover the model's classification accuracy. During the final 20 epochs, we trained the $f_{\theta_{student}}$ model using the LGGA loss with the 1% isolated backdoor examples and a learning rate of 0.0001. As for the Fine-pruning (FP) defense, we pruned the last convolutional layer of the $f_{\theta_{student}}$ model until its CA became similar to that of the other defense baselines.

We utilized the activation clustering defense method from Trojan-zoo's open-source code. Firstly, we trained $f_{\theta_{teacher}}$ for 80 epochs without data augmentation on the ResNet-18 architecture with the poisoned trainset. Then, we extracted the activation values of all data in layer 4 of the training dataset and applied the K-means clustering algorithm to cluster them into two clusters. Finally, we trained a new model from scratch using the cluster with more data. All defense methods were trained with Random Crop (padding = 4) and Random Horizontal Flip.

A.3 Model Architecture

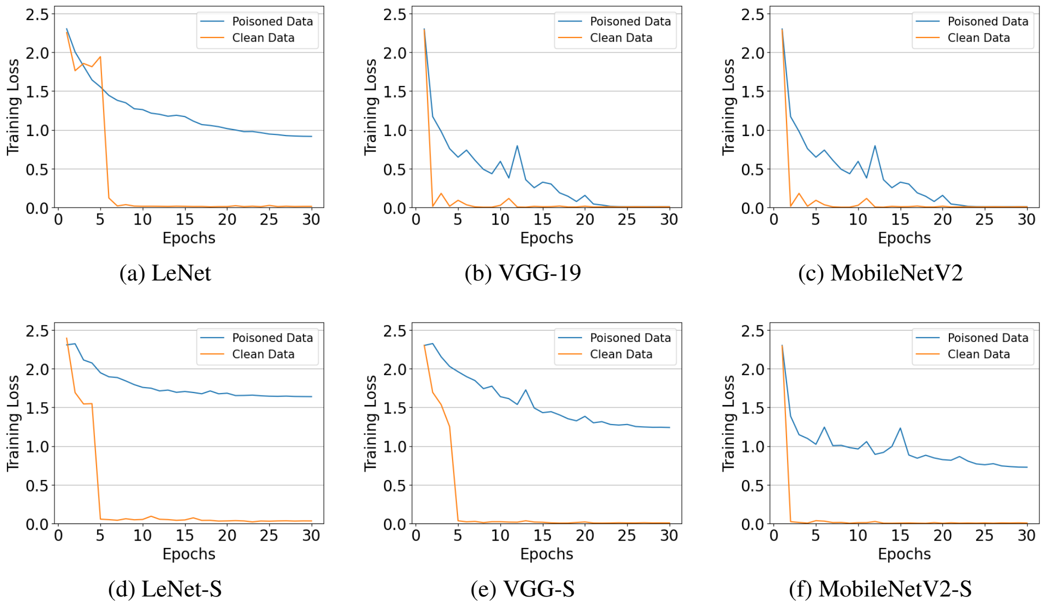
We employed various models in our experiments. In this section, we describe the architectures of all the small-size models we used.

We based our modifications on the ResNet architecture, altering the number of neurons and blocks to create ResNet-S, ResNet-S1, and ResNet-S2. For ResNet-S, ResNet-S1, and ResNet-S2, the first four layers consist of only one block each. In ResNet-S, these blocks contain 2, 2, 2, and 4 neurons respectively. For ResNet-S1, each block in the first four layers contains 1 neuron. In the case of ResNet-S2, the number of neurons in the blocks of the first four layers are 64, 128, 256, and 512 respectively.

For the VGG network, we modified the number of neurons in each convolutional layer to obtain the VGG-S network. Specifically, in the VGG-S network, the number of neurons in each convolutional layer are set as 4, 8, 16, 16, 32, 32, 32, and 512, respectively.

Finally, we made modifications to the MobileNetV2 network and obtained the MobileNetV2-S network. MobileNetV2-S consists of 2 bottlenecks, where bottleneck 1 contains 1 block with 3 convolutional layers, and the number of neurons in each layer is 4, 4, and 8 respectively. Bottleneck 2 contains 2 blocks, each with 3 convolutional layers, and the number of neurons in the two blocks' convolutional layers are 4, 4, 8 and 8, 8, 8 respectively. In addition, there are 2 convolutional layers outside the bottleneck, where the number of neurons is 4 and 16 respectively.

A.4 Exploration Across Various Model Architectures



(a) LeNet

(b) VGG-19

(c) MobileNetV2

(d) LeNet-S

(e) VGG-S

(f) MobileNetV2-S

Figure 6: The training loss of clean data and poisoned data on multiple model architectures, where the poisoned data are generated by BadNets backdoor attack and the models are constructed based on VGG, LeNet and MobileNet, their detiled informations are presented in appendix A.3. The experiments were conducted on the CIFAR-10 dataset with a poisoning rate of 10%.

We constructed low-capacity models on several other model architectures (described in detail in Appendix A.3) for additional exploration. As shown in Figure 6, the results indicate that the distinction in average loss between poisoned and clean data is consistently greater in low-capacity models compared to high-capacity models. This suggests that this phenomenon is prevalent across various model architectures.