Anti-Backdoor Learning: Training Clean Models on Poisoned Data

Anonymous Author(s)
Affiliation
Address
email

Abstract

Backdoor attack has emerged as a major security threat to deep neural networks (DNNs). While existing defense methods have demonstrated promising results on detecting and erasing backdoor triggers, it is still not clear if measures can be taken to avoid the triggers from being trained into the model in the first place. In this paper, we introduce the concept of *anti-backdoor learning*, of which the aim is to train clean models out of backdoor-poisoned data. We frame the overall learning process as a dual-task of learning the clean portion of data and learning the backdoored portion of data. From this view, we identify two weaknesses from the inherent features of backdoor attacks: 1) the models learn backdoored data at a much faster rate than learning clean data, and the stronger the attack the faster the models converge on the backdoored data; and 2) the backdoor task is tied to a specific class (the backdoor target class). Based on these two weaknesses, we propose a general learning scheme, Anti-Backdoor Learning (ABL), to automatically break backdoor attack during training. ABL introduces a two-stage gradient ascent mechanism into standard training to 1) help isolate backdoored data at an early training stage, and 2) break the correlation between the backdoored data and the target class at a later training stage. Through extensive experiments on multiple benchmark datasets against 6 state-of-the-art attacks, we empirically show that ABL-trained models on backdoor-poisoned data achieve almost the same performance as they were trained on purely clean data.

1 Introduction

Backdoor attacks are a type of training-time data poisoning attacks that implant backdoor triggers into machine learning models by injecting the trigger patterns into a small proportion of the training data [1]. The objective of backdoor attacks is to trick the model to learn a strong but task-irrelevant correlation between the trigger pattern and a target class, and aim to optimize three objectives: stealthiness of the trigger pattern, injection (poisoning) rate and attack success rate. A backdoored model performs normally on clean test data yet consistently predicts the target class whenever the trigger pattern is attached to a test example. Studies have shown that the widely adopted deep neural networks (DNNs) are particularly vulnerable to backdoor attacks [2]. Backdoor triggers are generally easy to implant but hard to detect or erase, posing significant security threats to deep learning.

Existing defense methods against backdoor attacks can be categorized into two types: detection methods and erasing methods. Detection methods exploit representation statistics or model properties to determine whether the model is backdoored [3, 4], or whether a training/test example is a backdoored example [5, 6]. Whilst detection can help identify potential risks, the backdoored model still needs to be purified. Erasing methods [7, 8, 9] take one step further and remove triggers from backdoored models. While existing defenses have demonstrated promising results, it is still not clear in the current literature whether any behavioral differences exist when models learn on backdoored data

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

instead of clean data. The explorations of these aspects lead to a fundamental yet so far overlooked question "*Is it possible to train a clean model on poisoned data*"?

Intuitively, if the backdoored data can be identified during training, measures can be taken to prevent this data from being learned. However, we find that this is not an easy task. One reason is that we do not know the distribution of backdoored data in advance. As shown in Figure 1, a high attack success rate can still be achieved on CIFAR-10 by different attacks even though the poisoning rate is less than 1%. This significantly increases the difficulty of backdoor data detection as the model's learning behavior may stay the same with or without a few examples. Even worse, the dataset might be completely clean and we may accidentally remove a lot of valuable data. One more important reason is that the backdoor may have already been learned by the model even if the backdoor examples are identified at a later training stage.

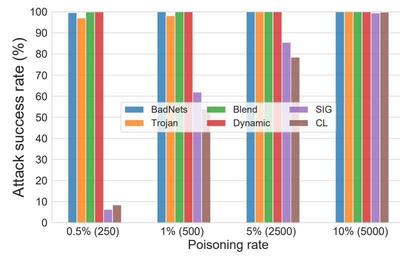


Figure 1: Attack success rate (ASR) of 6 backdoor attacks under different poisoning rates on CIFAR-10. 4 out of the 6 attacks can achieve nearly 100% ASR with only 0.5% poisoning rate.

In this paper, we frame the overall learning process of models on backdoor-poisoned datasets as a dual-task learning problem, with the learning of the clean portion as the original task and the learning of the backdoored portion as the backdoor task. By investigating the distinctive learning behaviors of the model for these two tasks, we identify two inherent features of backdoor attacks as their weaknesses. **First**, the backdoor task is a much easier task compared to the original task. Consequently, the training loss of backdoored portion drops abruptly in early epochs of training, whereas the loss of clean examples decreases in a steady pace. We also find that the stronger the attack, the faster the loss on backdoored data drops. This finding indicates that the backdoor correlations imposed by stronger attacks are easier and faster to learn, and marks one unique learning behavior on backdoored data. **Second**, the backdoor task is tied to a specific class (i.e., the backdoor target class). This indicates that the correlation between the trigger pattern and the target class could be easily broken by simply randomizing the class target, for instance, shuffling the labels of a small proportion of examples with low loss.

Inspired by the above observations, we propose a principled Anti-Backdoor Learning (ABL) scheme that enables the training of clean models without the prior knowledge of the distribution of backdoored data in datasets. ABL introduces a *gradient ascent* based anti-backdoor mechanism into the standard training to help isolate low-loss backdoor examples at an early training and unlearn the backdoor correlation once backdoor examples are isolated. In summary, our main contributions are:

- We present a novel view of the problem of robust learning with poisoned data and reveal two inherent weaknesses of backdoor attacks: faster learning on backdoored data and target-class dependency. The stronger the attack the more easily it can be detected or disrupted.
- We propose a novel Anti-Backdoor Learning (ABL) method that is capable of training clean models on poisoned data. To the best of our knowledge, ABL is the *first* method of its kind in the backdoor defense literature, complementing existing defense methods.
- We empirically show that our ABL is robust to 6 state-of-the-art backdoor attacks. The models trained using ABL are of almost the same clean accuracy as they were directly trained on clean data and the backdoor attack success rates on these models are close to random guess.

2 Related Work

Backdoor Attack. Existing backdoor attacks aim to optimize three main objectives: 1) making the trigger pattern stealthier; 2) reducing the injection (poisoning) rate; 3) increasing the attack success rate. Creative design of trigger patterns can help with the stealthiness of the attack. These can be simple patterns such as a single pixel [5] and a black-white checkerboard [1], or more complex patterns including blending backgrounds [10], natural reflections [11], invisible noise [12, 13, 14, 15], and adversarial patterns [16]. Backdoor attacks can be further divided into two categories: dirty-label attacks [1, 10, 11] and clean-label attacks [17, 18, 19, 16, 15]. Clean-label attacks are arguably stealthier as they do not change the labels. Backdoor attackers can also inject patterns via retraining the victim model on a reverse-engineered dataset without accessing the original training data [20]. Most of these attacks can achieve a high success rate (e.g., $> 95\%$) by poisoning only 10% or even less of the training data. A recent study by Zhao *et al.*[21] showed that even models trained on clean data can have backdoors, highlighting the importance of anti-backdoor learning.

Backdoor Defense. Most existing backdoor defenses fall under the categories of either detection-based methods or erasing-based methods. Detection-based methods aim to detect anomalies in input data [6, 5, 22, 23, 24, 25] or whether a model is backdoored [3, 4, 26, 27]. These methods typically show promising accuracies; however, the potential impact of backdoor triggers remains uncleared in the backdoored models. On the other hand, erasing-based methods take a step further and aim to purify the adverse impacts on models caused by the backdoor triggers. The current state-of-the-art erasing methods are Mode Connectivity Repair (MCR) [8] and Neural Attention Distillation (NAD) [9]. MCR mitigates the backdoors by selecting a robust model in the path of loss landscape, while NAD leverages knowledge distillation techniques to erase triggers. Other previous methods, including standard finetuning, traditional denoising, and fine-pruning [7], have been demonstrated to be insufficient against the latest attacks [28, 29, 11].

In this paper, we introduce the concept of *anti-backdoor learning*. Unlike existing methods, our goal is to train clean models directly out of the poisoned datasets without altering the models or the input data further. This requires a more in-depth understanding of the distinctive learning behaviors on backdoored data. However, such information is not available in the current literature. Anti-backdoor learning methods may replace the standard training to prevent potential backdoor attacks in real-world scenarios where data sources are not 100% reliable, and the distribution or even the existence of backdoor examples are unknown.

3 Anti-Backdoor Learning

In this section, we first formulate the anti-backdoor learning (ABL) problem, then reveal the distinctive learning behaviors on clean versus backdoor examples and introduce our proposed ABL method. Here, we focus on classification tasks with deep neural networks.

Defense Setting. We assume the backdoor adversary has pre-generated a set of backdoor examples and has successfully injected these examples into the training dataset. We also assume the defender has full control over the training process but has no prior knowledge of the proportion of backdoor examples in the given dataset. The defender's goal is to train a model on the given dataset (clean or poisoned) that is as good as models trained on purely clean data. Moreover, if an isolation method is used, the defender may identify only a subset of the backdoor examples. For instance, in the case of 10% poisoning, the isolation rate might only be 5% or even less.

Problem Formulation. Consider a standard classification task with a dataset $\mathcal{D} = \mathcal{D}_c \cup \mathcal{D}_b$ where $\mathcal{D}_c$ denoting the subset of clean data and $\mathcal{D}_b$ denoting the subset of backdoor data. The standard training trains a DNN model f_θ by minimizing the following empirical error:

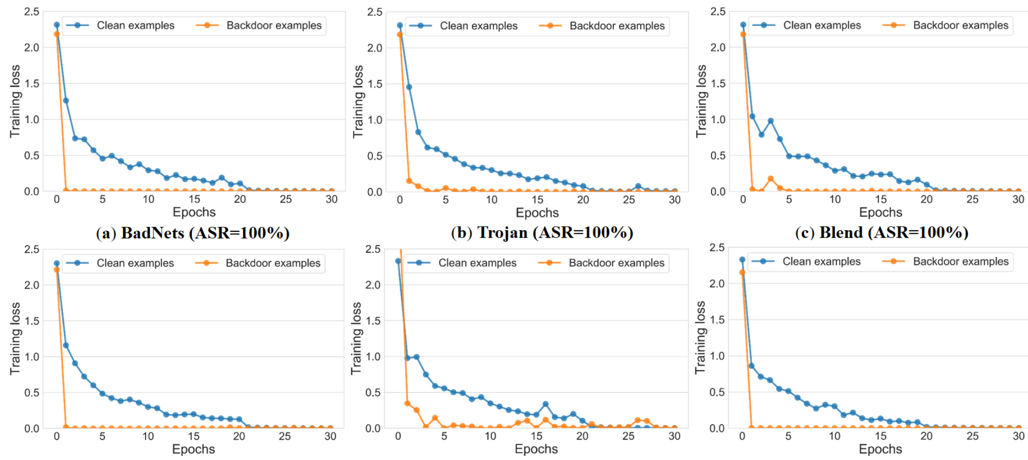
$$\mathcal{L} = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}[\ell(f_\theta(\boldsymbol{x}), y)] = \underbrace{\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_c}[\ell(f_\theta(\boldsymbol{x}), y)]}_{\text{clean task}} + \underbrace{\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_b}[\ell(f_\theta(\boldsymbol{x}), y)]}_{\text{backdoor task}}, \tag{1}$$

where $\ell(\cdot,\cdot)$ denotes the loss function such as the commonly used cross entropy loss. The overall learning task is decomposed into two tasks where the first *clean task* is defined on the clean data $\mathcal{D}_c$ while the second *backdoor task* is defined on the backdoor data $\mathcal{D}_b$. Since backdoor examples are often associated with a particular target class, all data from $\mathcal{D}_b$ may share the same class label. The above decomposition indicates that the standard learning approach tends to learn both tasks, resulting in a backdoored model.

To prevent backdoor examples from being learned, we propose anti-backdoor learning to minimize the following empirical error instead:

$$\mathcal{L} = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_c}[\ell(f_\theta(\boldsymbol{x}), y)] - \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_b}[\ell(f_\theta(\boldsymbol{x}), y)]. \tag{2}$$

Note the maximization of the backdoor task is defined on $\mathcal{D}_b$. Unfortunately, the above objective is undefined during training since we do not know the $\mathcal{D}_b$ subset. Intuitively, $\mathcal{D}_b$ can be detected and isolated during training if the model exhibits an atypical learning behavior on the backdoor examples. In the following subsection, we will introduce one such behavior, which we recognize as the first weakness of backdoor attacks.



(a) BadNets (ASR=100%) (b) Trojan (ASR=100%) (c) Blend (ASR=100%)

(d) Dynamic (ASR=100%) (e) SIG (ASR=99.46%) (f) CL (ASR=99.83%)

Figure 2: The training loss on clean versus backdoor examples crafted by 6 backdoor attacks including BadNets, Trojan, Blend, Dynamic, SIG, and CL. This experiment is conducted on CIFAR-10 with poisoning rate 10% and ResNet-18 [32]. ASR: attack success rate.

3.1 Unique Learning Behaviors on backdoor examples

We apply 6 backdoor attacks including BadNets [1], Trojan [20], Blend [10], Dynamic [30], SIG [31], and CL [18] to poison 10% of CIFAR-10 training data. We train a ResNet-18 model [32] on the corresponding backdoored dataset using the standard training method by solving equation (1) for each attack. Each model is trained following standard settings (see Section 4 and Appendix A.2). We plot the average training loss (i.e. cross entropy) on clean versus backdoored training examples in Figure 2. Clearly, for all 6 attacks, the training loss on backdoor examples drops quickly to zero at a very early stage (i.e., before 5 epochs), while the training loss on clean examples decreases to zero until the 20-th epoch. For all attacks except SIG, the training loss reaches almost zero after only two epochs of training. Moreover, according to the attack success rate, the stronger the attack the faster the training loss on backdoor examples drops.

The above observation indicates that the backdoor task is much easier than the clean task. This is not too surprising. In a typical clean dataset, not all examples are easy examples. Thus, it requires a certain number of training epochs to minimize the loss on those examples, even for small datasets like CIFAR-10. On the contrary, a backdoor attack adds a direct correlation between the trigger pattern and the target class to simplify and accelerate the injection of the backdoor trigger. We argue that this is a fundamental requirement and also a major weakness of backdoor attacks. For backdoor attacks to work successfully, the triggers should be easily learnable by the models, or else the attack would lose its effectiveness or require a much higher injection rate, which goes against its key objectives. Therefore, the stronger the attack the faster the training loss on backdoor examples drops to zero, e.g., comparing SIG with other attacks in Figure 2. We also show in Figure 7 in Appendix B.1 that the training loss of the backdoor task drops more rapidly as we increase the poisoning rate.

Based on the above observation, one may wonder if backdoor examples can be easily removed by filtering out the low-loss examples at an early stage (e.g., the 5-th epoch). However, we find that this strategy is not effective due to two reasons. First, the training loss in Figure 2 is the average training loss which means some backdoor examples can still have high training loss. Additionally, several powerful attacks such as Trojan and Dynamic can still succeed even with very few (50 or 100) backdoor examples. Second, if the training progresses long enough (e.g., beyond epoch 20), many clean examples will also have a low training loss, which makes the filtering significantly inaccurate. Therefore, we need a strategy to amplify the difference in training loss between clean and backdoor examples. Moreover, we need to unlearn the backdoor since the backdoor examples can only be identified when they are learned into the model (i.e., low training loss).

3.2 Proposed Anti-Backdoor Learning Method

Suppose the total number of training epochs is T, we decompose the entire training process into two stages, i.e., early training and later training. We denote the turning epoch from early training to later training by T_{te}. Our anti-backdoor learning method consists of two key techniques: 1) *backdoor isolation* during early training; and 2) *backdoor unlearning* during later training. The turning epoch is chosen to be the epoch where the average training loss stabilizes at a certain level.

Backdoor Isolation. During early training, we propose a *local gradient ascent* (LGA) technique to trap the loss value of each example around a certain threshold γ. We use the loss function $\mathcal{L}_{\text{LGA}}$ in equation (3) to achieve this. The gradient ascent is said to be "local" because the maximization is performed only around a fixed loss value γ. In other words, if the loss of a training example goes below γ, gradient ascent will be activated to boost its loss to γ; otherwise, the loss stays the same. Doing so allows us to effectively prevent clean examples from having a loss value smaller than γ, whereas backdoor examples can escape this constraint since their loss values drop significantly faster. The choice of an appropriate γ lies in the core of this strategy, as an overly large γ will hurt the learning of the clean task, while an overly small γ may not be strong enough to segregate the clean task from the backdoor task. We will show the optimal value of γ and its consistency across different datasets and models empirically. At the end of early training, we segregate examples into disjoint subsets: data with the bottom p percent loss will be isolated into the backdoor set $\widehat{\mathcal{D}}_b$ ($p = |\widehat{\mathcal{D}}_b|/|\mathcal{D}|$), and the rest into the clean set $\widehat{\mathcal{D}}_c$ ($\mathcal{D} = \widehat{\mathcal{D}}_b \cup \widehat{\mathcal{D}}_c$). An important note here is that the isolation rate (e.g. $p = 1\%$) is assumed to be much smaller than the poisoning rate (e.g. 10%).

Backdoor Unlearning. With the clean and backdoor sets, we can then continue with the later training. Note that at this stage, the backdoor has already been learned by the model. Given the above low isolation rate, an effective backdoor unlearning method is required to make the model unlearn the backdoor with a small subset $\widehat{\mathcal{D}}_b$ of backdoor examples while simultaneously learning the remaining (unisolated) backdoor examples in the clean set $\widehat{\mathcal{D}}_c$. We make this possible by exploiting the second weakness of backdoor attacks: the backdoor trigger is usually associated with a particular backdoor target class. We propose to use the loss $\mathcal{L}_{\text{GGA}}$ defined in equation (2) for this purpose. In $\mathcal{L}_{\text{GGA}}$, a *global gradient ascent* (GGA) is defined on the isolated subset $\widehat{\mathcal{D}}_b$. Unlike the local gradient ascent, it is not constrained to be around a fixed loss value. We will show in Section 4.2 that a low isolation rate of 1% is able to effectively unlearn the backdoor trigger against a high poising rate up to 70%.

The loss functions used by our ABL for two training stages are summarized as follows,

$$\mathcal{L}_{\text{ABL}}^t = \begin{cases} \mathcal{L}_{\text{LGA}} = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\big[\operatorname{sign}(\ell(f_\theta(\boldsymbol{x}),y)-\gamma)\cdot\ell(f_\theta(\boldsymbol{x}),y)\big] & \text{if } 0 \le t < T_{te} \\ \mathcal{L}_{\text{GGA}} = \mathbb{E}_{(\boldsymbol{x},y)\sim\widehat{\mathcal{D}}_c}\big[\ell(f_\theta(\boldsymbol{x}),y)\big] - \mathbb{E}_{(\boldsymbol{x},y)\sim\widehat{\mathcal{D}}_b}\big[\ell(f_\theta(\boldsymbol{x}),y)\big] & \text{if } T_{te} \le t < T, \end{cases} \tag{3}$$

where $t \in [0, T-1]$ is the current training epoch, $\operatorname{sign}(\cdot)$ is the sign function, γ is the loss threshold for LGA and $\widehat{\mathcal{D}}_b$ is the isolated backdoor set with isolation rate $p = |\widehat{\mathcal{D}}_b|/|\mathcal{D}|$. During early training ($0 \le t < T_{te}$), the loss will be automatically switched to $-\ell(f_\theta(\boldsymbol{x}),y)$ if $\ell(\cdot,\cdot)$ is smaller than γ by the sign function; otherwise the loss stays the same, i.e., $\ell(f_\theta(\boldsymbol{x}),y)$. Note that $\mathcal{L}_{\text{LGA}}$ loss may also be achieved by the flooding loss proposed in [33] to prevent overfitting: $|\ell(f_\theta(\boldsymbol{x}),y)-b|+b$ where b is a flooding parameter. Additionally, we will show that a set of other techniques may also achieve backdoor isolation and unlearning, but they are far less effective than our ABL.

4 Experiments

Attack Configurations. We consider six backdoor attacks in our experiments, including four dirty-label attacks: BadNets [1], Trojan attack [20], Blend attack [10], Dynamic attack [30], and two clean-label attacks: Sinusoidal signal attack(SIG) [31] and Clean-label attack(CL) [18]. We follow the settings suggested by [9] to configure these attack algorithms. All attacks are evaluated on three benchmark datasets, CIFAR-10 [34], GTSRB [35] and an ImageNet subset [36], with two classical model structures including WideResNet (WRN-16-1) [37] and ResNet-34 [32]. No data augmentations are used for these attacks since they hinder the backdoor effect [11]. We omit some attacks on GTSRB and ImageNet datasets due to the failure of reproduction following their original papers. The detailed settings of six backdoor attacks are summarized in Table 4 (see Appendix A.2).

Defense and Training Details. We compare our ABL with three state-of-the-art defense methods: Fine-pruning (FP) [7], Mode Connectivity Repair (MCR) [8], and Neural Attention Distillation

Table 1: The attack success rate (ASR %) and the clean accuracy (CA %) of 4 backdoor defense methods against 6 backdoor attacks. *None* means the training data is completely clean.

Dataset	Types	No Defense		FP		MCR		NAD		**ABL (Ours)**	
		ASR	CA	ASR	CA	ASR	CA	ASR	CA	ASR	CA
CIFAR-10	*None*	0%	89.12%	0%	85.14%	0%	87.49%	0%	88.18%	0%	**88.41%**
	BadNets	100%	85.43%	99.98%	82.14%	3.32%	78.49%	3.56%	82.18%	**3.04%**	**86.11%**
	Trojan	100%	82.14%	66.93%	80.17%	23.88%	76.47%	18.16%	80.23%	**3.81%**	**87.46%**
	Blend	100%	84.51%	85.62%	81.33%	31.85%	76.53%	**4.56%**	82.04%	16.23%	**84.06%**
	Dynamic	100%	83.88%	87.18%	80.37%	26.86%	70.36%	22.50%	74.95%	**18.46%**	**85.34%**
	SIG	99.46%	84.16%	76.32%	81.12%	0.14%	78.65%	1.92%	82.01%	**0.09%**	**88.27%**
	CL	99.83%	83.43%	54.95%	81.53%	19.86%	77.36%	16.11%	80.73%	**0%**	**89.03%**
	Average	99.88%	83.92%	78.50%	81.11%	17.65%	76.31%	11.13%	80.35%	**6.93%**	**86.71%**
GTSRB	*None*	0%	97.87%	0%	90.14%	0%	95.49%	0%	95.18%	0%	**96.41%**
	BadNets	100%	97.38%	99.57%	88.61%	1.00%	93.45%	0.19%	89.52%	**0.03%**	**96.01%**
	Trojan	99.80%	96.27%	93.54%	84.22%	2.76%	92.98%	0.37%	90.02%	**0.36%**	**94.95%**
	Blend	100%	95.97%	99.50%	86.67%	**6.83%**	92.91%	8.10%	89.37%	24.59%	**93.14%**
	Dynamic	100%	97.27%	99.84%	88.38%	64.82%	43.91%	68.71%	76.93%	**6.24%**	**95.80%**
	SIG	97.13%	97.13%	79.28%	90.50%	33.98%	91.83%	**4.64%**	89.36%	5.13%	**96.33%**
	Average	99.38%	96.80%	94.35%	87.68%	21.88%	83.01%	19.17%	87.04%	**7.27%**	**95.25%**
ImageNet Subset	*None*	0%	89.93%	0%	83.14%	0%	85.49%	0%	88.18%	0%	**88.31%**
	BadNets	100%	84.41%	97.70%	82.81%	28.59%	78.52%	6.32%	81.26%	**0.94%**	**87.76%**
	Trojan	100%	85.56%	96.39%	80.34%	6.67%	76.87%	15.48%	80.52%	**1.47%**	**88.19%**
	Blend	99.93%	86.15%	99.34%	81.33%	**19.23%**	75.83%	26.47%	82.39%	21.42%	**85.12%**
	Average	99.98%	85.37%	97.81%	81.49%	18.16%	77.07%	16.09%	81.39%	**7.94%**	**87.02%**

(NAD) [9]. For FP, MCR and NAD, we follow the configurations specified in their original papers, including the available clean data for finetuning/repair/distillation and training settings. The comparison with other data isolation methods are shown in Section 4.3. For our ABL, we set $T = 100$, $T_{te} = 20$, $\gamma = 0.5$ and isolation rate $p = 0.01$ (1%) in all experiments. The exploration of different T_{te}, γ and isolation rate p are also provided in Section 4.1. Three data augmentation techniques suggested in [9]: random crop (padding = 4), horizontal flipping, and cutout, are applied for all defense methods. More details on defense settings can be found in Appendix A.3.

Evaluation Metrics. We adopt two commonly used performance metrics: Attack Success Rate (ASR), which is the classification accuracy on the backdoor test set, and Clean Accuracy (CA), the classification accuracy on clean test set.

4.1 Effectiveness of Our ABL Defense

Comparison to Existing Defenses. Table 1 demonstrates our proposed ABL defense on CIFAR-10, GTSRB, and an ImageNet Subset. We consider 6 state-of-the-art backdoor attacks and compare the performance of ABL with the other three backdoor defense techniques. It is clear that our ABL achieves the best results on reducing ASR against most of backdoor attacks, while maintaining an extremely high CA across all three datasets. In comparison to the best baseline method NAD, our ABL achieves 4.2% (6.93% vs. 11.13%), 11.9% (7.27% vs. 19.17%) and 8.15% (7.94% vs. 16.09%) lower average ASR against the 6 attacks on CIFAR-10, GTSRB and ImageNet subset, respectively. This superiority becomes more significant when compared to other baseline methods.

We notice that our ABL is not always the best when looking at the 6 attacks individually. For instance, NAD is the best defense against Blend attack on CIFAR-10 and against SIG attack on GTSRB, while MCR is the best against Blend on GTSRB and ImageNet subset. We suspect this is because both Blend and SIG attacks mix the trigger pattern (i.e., another image or superimposed sinusoidal signal) into the background of the poisoned images, producing an effect of natural artifacts. This makes them harder to isolate and unlearn, since even clean data can have such patterns [21]. This is one limitation of our ABL that needs further improvement in future works. Note that, for both attacks, our defense can still reduce their ASRs to at least below 25% across all three datasets. We also identify the Dynamic attack to be the strongest in general. For example, on GTSRB dataset, baseline methods

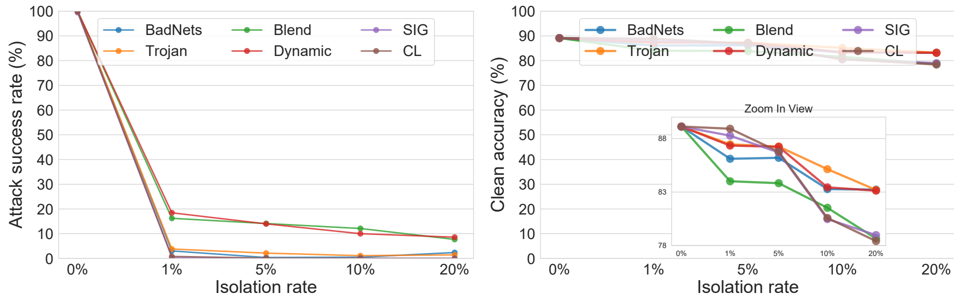


Figure 3: Performance of our ABL with different isolation rate $p \in [0.01, 0.2]$ on CIFAR-10 dataset. Left: attack success rate (ASR); Right: clean accuracy of ABL against 6 backdoor attacks.

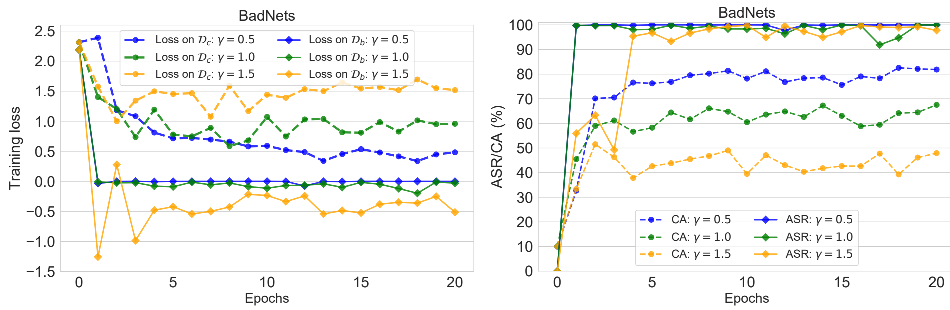


Figure 4: Separation effect of local gradient ascent with different γ on CIFAR-10 against BadNets. Left: Training loss on the ground truth backdoor ($\mathcal{D}_b$) and clean ($\mathcal{D}_c$) subsets; Right: Attack success rate (ASR) and clean accuracy (CA). The gap between the two lines of the same color becomes wider for larger γ, i.e., better separation effect.

NAD, MCR and FP can only decrease Dynamic's ASR to 68.71%, 64.82% and 99.84%, respectively, a result that is much worse than the 6.24% of our ABL.

Clean accuracy is as important as ASR reduction, as the model would completely lose its utility if clean accuracy is much sacrificed by the defense. By inspecting the average CA results in Table 1, one can find that our ABL achieves nearly the same clean accuracy as models trained on 100% clean (shown in row *None* and column 'No Defense') datasets. Particularly, our ABL surpasses the average clean accuracy of NAD by 6.36% (86.71% vs. 80.35%), 8.21% (95.25% vs. 87.04%) and 5.63% (87.02% vs. 81.39%) on CIFAR-10, GTSRB and ImageNet subset, respectively. FP defense decreases model performance even when training data is clean (the *None* row). This makes our ABL defense more practical for industrial applications where performance is equally important as security.

Effectiveness with Different Isolation Rates. Here, we study the correlation between isolation rate $p = |\widehat{\mathcal{D}}_b|/|\mathcal{D}|$ and the performance of our ABL, on CIFAR-10 dataset. We run ABL with different $p \in [0.01, 0.2]$ and show the attack success rate and clean accuracy in Figure 3. There is a trade-off between ASR reduction and clean accuracy. Specifically, high isolation rates can isolate more backdoor examples for the later stage of unlearning, producing much lower ASRs. However, it also puts more examples into the unlearning mode, which harms clean accuracy. In general, ABL with isolation rate $< 5\%$ works reasonably well against all 6 attacks, even though the backdoor poisoning rate is much higher, i.e., 70% (see Figure 5 in Section 4.2). Along with the results in Table 1, this confirms that it is indeed possible to break and unlearn the backdoor correlation with only a tiny subset of correctly-identified backdoor examples, highlighting one unique advantage of backdoor isolation and unlearning approaches.

Effectiveness with Different Turning Epochs. Here, we study the impact of the timing to switch from the learning stage ($\mathcal{L}_{\text{LGA}}$) to the unlearning stage ($\mathcal{L}_{\text{GGA}}$) on CIFAR-10. We compare four different tuning epochs: the 10th, 20th, 30th, and 40th epoch, and record the results of our ABL in Table 5 (see Appendix B.3). We find that delayed turning epochs tend to slightly hinder the defense performance. Despite the slight variations, all choices of the turning epoch help mitigate backdoor attacks, but epoch 20 (i.e., at 20% - 30% of the entire training progress) achieves the best overall results. This tend is consistent on other datasets as well. We attribute these results to the success of

LGA in preserving the difference between clean and backdoor samples over time, which enables us to select the tuning epoch flexibly. More understandings of LGA are discussed in Section 4.2.

4.2 Comprehensive Understanding of ABL

Importance of Local Gradient Ascent. To help understand how LGA works in isolating backdoor data, we visualize and compare in Figure 4 the training loss and the model's performances (ASR and CA) under three different settings where γ is set to 0.5, 1.0, and 1.5. It is evident that LGA can segregate backdoor examples from clean examples to a certain extent under all three settings of γ by preventing the loss of clean examples from converging. Moreover, a larger γ leads to a wider difference in training loss as well as ASR and CA. However, we note that this may cause training instability, as evidenced by the relatively larger fluctuations with $\gamma = 1.5$.

We also examine the precision of the 1% isolated backdoor set under different γ of 0, 0.5, 1.0, and 1.5 on CIFAR-10, GTSRB, and the ImageNet subset. We use BadNets attack with poisoning rate 10% and set the turning (isolation) epoch of ABL to 20. We report the isolation precision results in Table 6 (see Appendix B.4). As can be seen, when $\gamma = 0$, the detection precision is poor; this indicates that it is tough for the model to tell apart backdoor examples from the clean ones without the LGA, which is foreseeable because the clean training loss is uncontrolled and overlaps with the backdoor training loss. Note that as soon as we set $\gamma > 0$, the precision immediately improves on both CIFAR-10 and the ImageNets subset. Additionally, the precision of the isolation task is not sensitive to the change in γ, which again allows the hyperparameter value to be flexibly chosen.

In summary, LGD creates and sustains a gap between the training loss of clean and backdoor examples, which plays a vital role in extracting an isolated backdoor set.

Stress Testing: Fixing 1% Isolation Rate While Increasing Poisoning Rate. Now that we know we can confidently extract a tiny subset of backdoor examples with high purity, the challenge remains whether the extracted set is sufficient for the model to unlearn the backdoor. We demonstrate that our ABL is a stronger method to defend against backdoor attacks, even under this strenuous setting. Here, we experiment on CIFAR-10 against BadNets with increasing poisoning rate from 10% to 100% and show the results in Figure 5. Even with a high poisoning rate of 70%, our ABL method can reduce the ASR from 100% to 5.02%. Note that ABL will break when the poisoning rate $\geq 80\%$, however, in this case, we argue that the dataset should not be used to train any models in the first place. As we mentioned before, the correlation between the backdoor pattern and the target label exposes a weakness of backdoor attacks. Our ABL utilizes the GGA to break this link and achieve defense goals effortlessly.

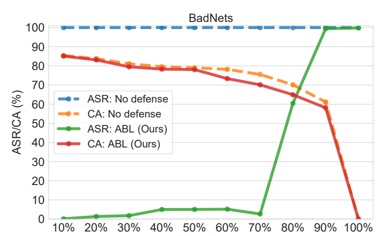


Figure 5: Performance of ABL with isolation rate 1% against different poisoning rates of BadNets on CIFAR-10.

4.3 Exploring Alternative Isolation and Unlearning Methods

Alternative Isolation Methods. In this section, we compare the isolation precision of our ABL with two backdoor detection methods, namely Activation Clustering (AC) [6] and Spectral Signature Analysis (SSA) [5]. The goal is to isolate 1% of training examples into the backdoor set ($\widehat{\mathcal{D}}_b$), and we provide in Figure 8 (see Appendix B.2) the precision of these methods alongside our ABL in detecting the 6 backdoor attacks on CIFAR-10 dataset. We find that both AC and SS achieve high detection rates on BadNets and Trojan attacks, however, perform poorly on the other 4 attacks. A reasonable explanation is that attacks covering the whole image with complex triggers (e.g., Blend, Dynamic, SIG, and CL) give confusing and unidentifiable output representations of either feature activation or spectral signature, making these detection methods ineffective. It is worth mentioning that our ABL is effective against all backdoor attacks with the highest average detection rate. In addition, we find that the flooding loss [33] proposed for mitigating overfitting is also very effective for backdoor isolation. We also explore a confidence-based isolation with label smoothing (LS), which unfortunately fails on most attacks. More details of these explorations can be found in Figure 9 and 10 in Appendix B.5.

Alternative Unlearning Methods. Here we explore several other empirical strategies, including image-based, label-based, model-based approaches, to rebuild a clean model on the poisoned data.

Table 2: Performance of various unlearning methods against BadNets attack on CIFAR-10.

Backdoor Unlearning Methods	Method Type	Discard $\widehat{\mathcal{D}}_b$	Backdoored		After Unlearning	
			ASR	CA	ASR	CA
Pixel Noise	Image-based	No	100%	85.43%	57.54%	82.33%
Grad Noise	Image-based	No	100%	85.43%	**47.65**%	**82.62**%
Label Shuffling	Label-based	No	100%	85.43%	30.23%	83.76%
Label Uniform	Label-based	No	100%	85.43%	75.12%	83.47%
Label Smoothing	Label-based	No	100%	85.43%	99.80%	83.17%
Self-Learning	Label-based	No	100%	85.43%	**21.26**%	**84.38**%
Fine-tuning All Layers	Model-based	Yes	100%	85.43%	99.12%	83.64%
Fine-tuning Last Layers	Model-based	Yes	100%	85.43%	22.33%	77.65%
Fine-tuning ImageNet Model	Model-based	Yes	100%	85.43%	12.18%	75.10%
Re-training from Scratch	Model-based	Yes	100%	85.43%	11.21%	86.02%
ABL	Model-based	No	100%	85.43%	**3.04**%	**86.11**%

These approaches are motivated by the second weakness of backdoor attacks, and are all designed to break the connection between the trigger pattern and the target class. We experiment on CIFAR-10 with BadNets (10% poisoning rate), and fix the backdoor isolation method to our ABL with a hig isolation rate 20% (as most of them will fail with 1% isolation). Table 2 summarizes our explorations. Our core findings can be summarized as: **a)** adding perturbations to pixels or gradients is not effective; **b)** changing the labels of isolated examples is mildly effective; **c)** finetuning some (not all) layers of the model cannot effectively mitigate backdoor attacks; **d)** "self-learning" and "retraining the model from scratch" on the isolated clean set are good choices against backdoor attacks; and **e)** our ABL presents the best unlearning performance. Details of these methods are given in Appendix A.3. The performance of these methods under the 1% isolation rate is also reported in Table 7 in Appendix B.6.

5 Conclusion

In this work, we identified two inherent features of backdoor attacks as their weaknesses: 1) backdoored data are easier for models to learn than clean data, and 2) backdoor learning establishes a stronger correlation between the trigger and the target label. Based on these two findings, we proposed a novel framework - Anti-Backdoor Learning (ABL) - which consists of two stages of learning utilizing local gradient ascent (LGA) and global gradient ascent (GGA), respectively. At the early learning stage, we use LGA to intentionally maximize the training loss gap between clean examples and backdoored examples to isolate out the backdoored data via the low loss value. We use GGA to unlearn the backdoored model with the isolated backdoor data at the last learning stage. Empirical results demonstrate that our ABL is resilient to various experimental settings and can effectively defend against 6 state-of-the-art backdoor attacks. Our work introduces a simple but very effective ABL method for industries to train backdoor-free models on real-world datasets, and opens up a new research direction for backdoor defense.

Broader Impact

Data has been key to the success of deep learning and modern artificial intelligence (AI). However, it is hard to guarantee the quality and purity of data in many cases, and even high-quality datasets may contain backdoors, especially when they are collected from the internet. By introducing this new concept of *anti-backdoor learning* (ABL), our work opens up a new angle looking into the process of learning with data. From a broader perspective, ABL prevents deep learning models from learning (or overfitting) to some easy patterns, and further highlights the fact that easy patterns are not contributing much to the overall performance. Beyond backdoor defense, ABL should be explored as a generic quality-ware learning mechanism in place of traditional quality-agnostic learning. Such a mechanism can help prevent many potential data-quality-related risks, for example, the risk of overfitting, bias, disruptive noise, and backdoor. Although not our initial intention, our work may adversely be exploited to develop more advanced and stealthy backdoor attacks. This essentially requires more advanced defense methods to combat.

References

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

[2] Cheng-Hsin Weng, Yan-Ting Lee, and Shan-Hung Brandon Wu. On the trade-off between adversarial and backdoor robustness. In *NeurIPS*, 2020.

[3] 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 *S&P*. IEEE, 2019.

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

[5] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. In *NeurIPS*, 2018.

[6] 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 *AAAI Workshop*, 2019.

[7] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *RAID*, 2018.

[8] Pu Zhao, Pin-Yu Chen, Payel Das, Karthikeyan Natesan Ramamurthy, and Xue Lin. Bridging mode connectivity in loss landscapes and adversarial robustness. In *ICLR*, 2020.

[9] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Neural attention distillation: Erasing backdoor triggers from deep neural networks. In *ICLR*, 2021.

[10] 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.

[11] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection backdoor: A natural backdoor attack on deep neural networks. In *ECCV*, 2020.

[12] Cong Liao, Haoti Zhong, Anna Squicciarini, Sencun Zhu, and David Miller. Backdoor embedding in convolutional neural network models via invisible perturbation. *CODASPY*, 2020.

[13] Shaofeng Li, Minhui Xue, Benjamin Zi Hao Zhao, Haojin Zhu, and Xinpeng Zhang. Invisible backdoor attacks on deep neural networks via steganography and regularization. *arXiv preprint arXiv:1909.02742*, 2019.

[14] Jinyin Chen, Haibin Zheng, Mengmeng Su, Tianyu Du, Changting Lin, and Shouling Ji. Invisible poisoning: Highly stealthy targeted poisoning attack. In *ICISC*, 2019.

[15] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. Hidden trigger backdoor attacks. In *AAAI*, volume 34, pages 11957–11965, 2020.

[16] Shihao Zhao, Xingjun Ma, Xiang Zheng, James Bailey, Jingjing Chen, and Yu-Gang Jiang. Clean-label backdoor attacks on video recognition models. In *CVPR*, pages 14443–14452, 2020.

[17] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *NeurIPS*, 2018.

[18] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Clean-label backdoor attacks. *https://people.csail.mit.edu/madry/lab/*, 2019.

[19] Chen Zhu, W Ronny Huang, Hengduo Li, Gavin Taylor, Christoph Studer, and Tom Goldstein. Transferable clean-label poisoning attacks on deep neural nets. In *ICML*, pages 7614–7623. PMLR, 2019.

[20] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *NDSS*, 2018.

[21] 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.

[22] 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 *ACSAC*, 2019.

[23] Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A Gunter, and Bo Li. Detecting ai trojans using meta neural analysis. In *S&P*, 2021.

[24] Jonathan Hayase, Weihao Kong, Raghav Somani, and Sewoong Oh. Spectre: Defending against backdoor attacks using robust statistics. In *ICML*, 2021.

[25] Di Tang, XiaoFeng Wang, Haixu Tang, and Kehuan Zhang. Demon in the variant: Statistical analysis of dnns for robust backdoor contamination detection. In *USENIX Security*, 2021.

[26] Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. Universal litmus patterns: Revealing backdoor attacks in cnns. In *CVPR*, 2020.

[27] 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 *ICML*, 2021.

[28] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y Zhao. Latent backdoor attacks on deep neural networks. In *CCS*, 2019.

[29] Yiming Li, Tongqing Zhai, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shutao Xia. Rethinking the trigger of backdoor attack. *arXiv preprint arXiv:2004.04692*, 2020.

[30] Anh Nguyen and Anh Tran. Input-aware dynamic backdoor attack. In *NeurIPS*, 2020.

[31] Mauro Barni, Kassem Kallas, and Benedetta Tondi. A new backdoor attack in cnns by training set corruption without label poisoning. In *ICIP*, 2019.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[33] Takashi Ishida, Ikko Yamane, Tomoya Sakai, Gang Niu, and Masashi Sugiyama. Do we need zero training loss after achieving zero training error? In *ICML*, 2020.

[34] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[35] 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.

[36] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[37] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 1
   (b) Did you describe the limitations of your work? [Yes] See Section 4.1
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 5
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] The code will be released once the paper is accepted.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4 and Appendix A.3
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] The experimental results reported in our paper is already the average results over multiple runs.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix A.3
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]