MEDIC : Model Backdoor Removal by Importance Driven Cloning

Abstract

We develop a novel method to remove injected backdoors in Deep Learning models. It works by cloning the benign behaviors of a trojaned model to a new model of the same structure. It trains the clone model from scratch on a very small subset of samples (i.e., 5%) and aims to minimize a cloning loss that denotes the differences between the activations of important neurons across the two models. The set of important neurons varies for each input, depending on their magnitude of activations and their impact on the classification result. Our experiments show that our technique can effectively remove seven different types of backdoors with minor benign accuracy degradation, outperforming the state-of-the-art backdoor removal techniques that are based fine-tuning, knowledge distillation, and neuron pruning.

1 INTRODUCTION

Backdoor attack is a prominent threat to applications of Deep Learning models. Misbehaviors, e.g., model misclassification, can be induced by an input stamped with a *backdoor trigger*, such as a patch with a solid color or a pattern [Gu et al., 2019b, Liu et al., 2018b]. A large number of various backdoor attacks have been proposed, targeting different modalities and featuring different attack methods [Zhang et al., 2021, Kurita et al., 2020, Rezaei and Liu, 2020, Wang et al., 2018, Tolpegin et al., 2020, Bagdasaryan et al., 2020]. An important defense method is hence to remove backdoors from pre-trained models. For instance, Liu et al. [2018a] proposed to remove backdoor by fine-tuning a pre-trained model on clean inputs. Distillation [Li et al., 2021] removes neurons that are not critical for benign functionalities. Model connectivity repair (MCR) [Zhao et al., 2020a] interpolates weight parameters of a poisoned model and its finetuned version. ANP [Wu and Wang, 2021] adversarially perturbs neuron weights of a poisoned model and prunes those neurons that are sensitive to perturbations (and hence considered compromised). While these techniques are very effective in their targeted scenarios, they may fall short in some other attacks. For example, if a trojaned model is substantially overfit on a trigger, fine-tuning may not be able to remove the backdoor without lengthy retraining. Distillation may become less effective if a neuron is important for both normal functionalities and backdoor behaviors. And pruning neurons may degrade model benign accuracy. More discussions of related work can be found in Section 2.

In this paper, we propose a novel backdoor removal technique MEDIC [1] as illustrated in Figure 1 (A). It works by cloning the benign functionalities of a pre-trained model to a new sanitized model of *the same structure*. Given a trojaned model and a small set of clean samples (i.e., 5% of the original data in this paper), MEDIC trains the clone model from scratch. The training is not only driven by the cross-entropy loss as in normal model training, but also by forcing the clone model to generate the same internal activation values as the original model *at the corresponding internal neurons*, i.e., steps ①, ②, and ④ in Figure 1 (A). Intuitively, one can consider it essentially derives the weight parameters in the clone model by resolving the activation equivalence constraints. There are a large number of such constraints even with just a small set of clean samples. However, such faithful cloning likely copies the backdoor behaviors as well as it tends to generate the same set of weight values. Therefore, our cloning is further guided by importance (step ③). Specifically, for each sample x, we only force the important neurons to have the same activation values. A neuron is considered important if (1) it tends to be substantially activated by the sample, when compared to its activation statistics over the entire population (the *activation criterion* in red in Figure 1 (A)), and (2) the large activation value has substantial impact on the classification result (the

[1] MEDIC stands for "**M**od**E**l *Backdoor Removal by Importance* **D**r**I**ven **C**oning".

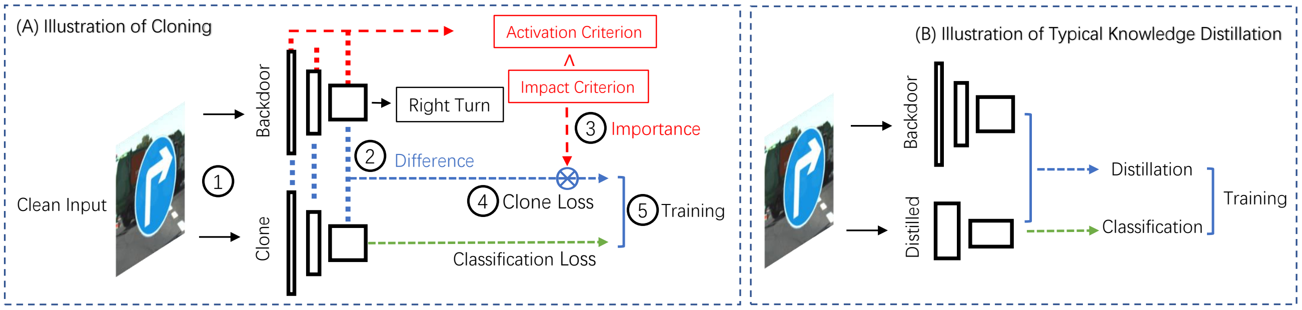

Figure 1: Workflow of MEDIC and comparison with typical knowledge distillation [Hinton et al., 2015]

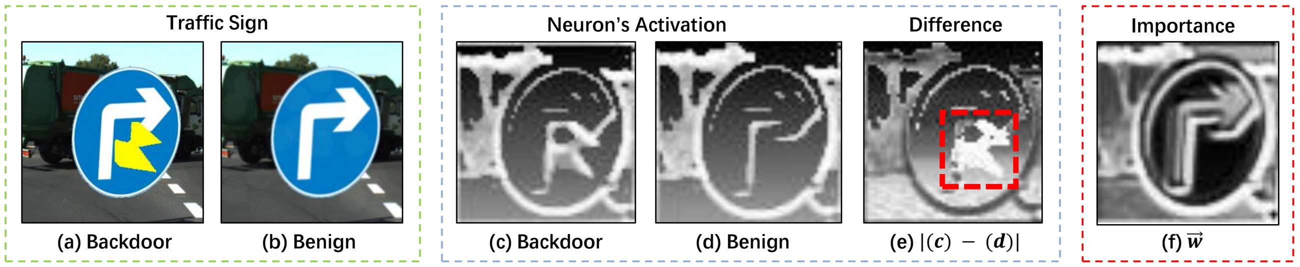

Figure 2: Example, with (a) and (b) the clean and trojaned inputs, (c)-(f) the internal activations of the trojaned input and the clean input, their differences, and the importance values in eq.(4) for the clean input, respectively. We resize and rescale the internal activation for better visualization.

impact criterion) . The latter can be determined by analyzing the output gradient regarding the neuron activation. By constraining only the important neurons and relaxing the others, the model focuses on cloning the behaviors related to the clean inputs and precludes the backdoor. The set of weight values derived by such cloning are largely different from those in the original model. Intuitively, it is like solving the same set of variables with only a subset of equivalence constraints. The solution tends to differ substantially from that by solving the full set of constraints.

Example. Figure 2 shows an example by visualizing the internal activations and importance values for a trojaned model from TrojAI [NIST, 2019]. Image (a) shows a right-turn traffic sign stamped with a polygon trigger in yellow, which induces the classification result of stop sign. Image (c) visualizes the activation values for the trojaned image whereas (d) shows those for its clean version. The differences between the two, visualized in (e), show that the neurons fall into the red box are critical to the backdoor behaviors. A faithful cloning method would copy the behaviors for these neurons (and hence the backdoor). In contrast, our method modulates the cloning process using importance values and hence precludes the backdoor. The last image shows the importance values with the bright points denoting important neurons and the dark ones unimportant. Observe that it prevents copying the behaviors of the compromised neurons *for this example* as they are unimportant. One may notice that the unimportant compromised neurons for *this example* may become important for another example. Our method naturally handles this using per-sample importance values. □

Our technique is different from fine-tuning as it trains from scratch. It is also different from *knowledge distillation* [Li and Hoiem, 2016] shown in Figure 1 (B), which aims to copy behaviors across different model structures and constrains logits equivalence (and sometimes internal activation equivalence [Zagoruyko and Komodakis, 2016a] as well). In contrast, by using the same model structure, we have a one-to-one mapping for individual neurons in the two models such that we can enforce strong constraints on internal equivalence. This allows us to copy behaviors with only a small set of clean samples.

We evaluate our technique on seven different kinds of backdoor attacks. We compare with five latest backdoor removal methods (details in related work). The results show our method can reduce the attack success rate to 9.5% on average with only 2% benign accuracy degradation. It consistently outperforms the other baselines by 25%. It usually takes 15 mins to sanitize a model. We have also conducted an adaptive attack in which the trigger is composed of existing benign features in clean samples. It denotes the most adversary context for MEDIC . Our ablation study shows all the design choices are critical. For example, without using importance values, it can only reduce ASR to 36% on average.

In summary, our main contributions include the following.

- We propose a novel importance driven cloning method to remove backdoors. It only requires a small set of clean samples.
- We theoretically analyze the advantages of the cloning method.
- We empirically show that MEDIC outperforms the state-of-the-art methods.

2 RELATED WORK

There are a large body backdoor attacks [Gu et al., 2019a, Chen et al., 2017, Shafahi et al., 2018, Zhu et al., 2019, Zhao et al., 2020b, Saha et al., 2020, Salem et al., 2020, Nguyen and Tran, 2020, Bagdasaryan and Shmatikov, 2020, Cheng et al., 2021, Nguyen and Tran, 2021]. We focus on a number of representative ones with different natures and used in our experiments. Readers interested in other attacks are referred to a survey [Li et al., 2020, Gao et al., 2020]. Badnet [Gu et al., 2019b] proposes to inject backdoors with a fixed polygon. Clean Label attack [Turner et al., 2018] first adds adversarial perturbations to target-class samples and makes them misclassified. It then adds a square patch on those inputs without changing their ground-truth label. During inference, the square patch can induce misclassification on non-target class samples. SIG [Barni et al., 2019] uses a wave-like pattern as trigger. Composite attack [Lin et al., 2020] combines benign features from two existing classes as the trigger. Reflection attack [Liu et al., 2020] adds the reflection of some external image on the input. Polygon attack [NIST, 2019] injects a polygon-like trigger on the input at specific locations. Filter attack [Liu et al., 2019] utilizes Instagram filters to transform inputs and labels the transformed inputs to the target class. The trigger perturbations hence vary for different inputs.

Researchers have proposed various defense techniques, including backdoor inputs detection that aims to detect and reject input samples with triggers [Gao et al., 2019, Chou et al., 2020, Tran et al., 2018, Veldanda et al.]; backdoor scanning that determines whether a given model has backdoor using a small set of clean inputs [Wang et al., 2019, Liu et al., 2019, Kolouri et al., 2020, Tang et al., 2021]; backdoor certification that certifies the predictions of individual samples with trigger [McCoyd et al., 2020, Xiang et al., 2021a,b, Jia et al., 2020]; model hardening that adversarially trains models using triggers inverted from the subject (clean) model [Tao et al., 2022]; backdoor removal that erases injected backdoors using a small set of clean samples [Li et al., 2021, Wu and Wang, 2021, Liu et al., 2018a]. Our work falls in the last category. Fineprune [Liu et al., 2018a] iteratively prunes neurons with small activation values on clean samples and then finetunes the pruned model. Note that the pruned model has different structure from the original one. In our experience, having the same structure is critical as it provides the optimal neuron alignment. Model connectivity repair (MCR) [Zhao et al., 2020a] interpolates parameters of a poisoned model and its finetuned version. NAD [Li et al., 2021] first finetunes a poisoned model and then performs transfer learning, treating the finetuned model as the teacher and the poisoned model as the student. Its effectiveness hinges on the quality of the first finetuning step. ANP [Wu and Wang, 2021] adversarially perturbs neuron weights of the poisoned model and prunes neurons sensitive to perturbations. We compare MEDIC with the aforementioned approaches in Section 5 and demonstrate that ours outperforms. Note that our method is complementary to adversarial training based methods like MOTH [Tao et al., 2022], which focuses on stronger data argumentation that leverages inverted triggers. These methods are usually orders of magnitude slower as they iteratively invert new triggers.

3 METHOD

Problem Statement. Given a multi-class backdoor classifier $f^* : X \to Y$, where $X \subset \mathbb{R}^d$ and $Y \subset \mathbb{R}^C$, d the input dimension and C the number of classes. Samples (X, Y) are drawn from the joint distribution $\mathcal{S} = (\mathcal{X}, \mathcal{Y})$. The classifier f^* is originally trained on a large dataset $D \sim (\mathcal{X}, \mathcal{Y})^n$ and additional backdoor samples $D_a \sim (\mathcal{X}_a, \mathcal{Y}_a)$. We aim to remove the backdoor behaviors in f^* and retain the benign behaviors. To do so, we leverage a small additional dataset $D_s \sim (\mathcal{X}, \mathcal{Y})^q$ where $|D_s| \ll |D|$. It is used to facilitate the cloning process. Function f^c denotes the cloned and sanitized model using our method.

Our first goal is to clone the (benign) functionalities of the model instead of its parameters. Specifically, f^c is supposed to produce similar outputs as the original model (with backdoor) on clean inputs. This is stated as the *functionality* goal as follows.

$$\mathbb{E}_{x\sim\mathcal{X}}\left(\|f^c(x) - f^*(x)\|_2\right) \approx 0 \qquad \text{(Functionality)}$$

In addition, f^c shall not have the backdoor behavior on samples from the backdoor distribution $(\mathcal{X}_a, \mathcal{Y}_a)$, with a high probability $1 - \delta$. It is stated as the *sanity* goal.

$$\mathbb{E}_{(x,y)\sim(\mathcal{X}_a,\mathcal{Y}_a)}\mathbb{1}\left[f^c(x) \neq y\right] \geq 1 - \delta \qquad \text{(Sanity)}$$

It is worth noting that similar functionalities do not imply a similar set of parameters. In fact, our importance driven cloning derives a set of parameters different from those in the original model to meet the two stated goals.

Cloning. To achieve the first goal of copying normal functionalities using only a small set of benign samples, we propose to train a clone model from scratch that has the same structure, by forcing the clone model to have the same activation values for all the corresponding neurons and also the output logits as the original model. Such cloning is different from transfer learning, in which teacher and student models may be of different structures as it is not that meaningful to copy functionalities to a model with the same structure in transfer learning's application scenarios.

While inputs often have a fixed bound, internal activation values have dynamic ranges. If we directly use activation value differences in the clone training loss, such range variations cause difficulties in convergence. Therefore, we normalize activation values before using them in the loss function. Formally speaking, given m internal neurons of a backdoored model, denoted by functions $f_i^* : R^d \to R$ with $i \in [1, m]$[2] and the corresponding ones in the clone model f_i^c, we aim to enforce the corresponding neuron functions to produce the same values. Let μ_i and σ_i be the mean and standard deviation of the random variable $f_i^*(\mathcal{X})$ and $w_i(x)$ an importance weight. We use $\overrightarrow{f(x)}$, $\overrightarrow{w(x)}$, $\vec{\sigma}$ and $\vec{\mu}$ to represent their concatenated vectors over the m neurons, respectively. We can derive the following loss function.

$$\mathcal{L}_{\text{clone}} = \mathbb{E}_{x\sim\mathcal{X}}\left[\sum_i w_i(x)\left(\frac{f_i^*(x) - f_i^c(x)}{\sigma}\right)^2\right] \quad (1)$$

where

$$w_i(x) \geq 0$$

Intuitively, we minimize the differences between the outputs of the corresponding neurons. For the moment, one can assume $w_i(x) = 1/m$, meaning that we enforce such constraints equally on all neurons. We will explain how we change $w_i(x)$ to preclude backdoor behaviors later in the section.

Our results in Figure 3 show that we can faithfully clone the normal functionalities with only 2% samples. We also formally analyze the essence of cloning in Section 4.

Pruning Backdoor by Importance Driven Cloning. To preclude backdoor while cloning, we leverage the importance value $w_i(x)$ computed for each sample and each neuron. A neuron is important for an sample, if and only if the neuron is substantially activated (compared to its activation statistics over the entire population) and has a large impact on the final output.

To determine if a neuron is substantially activated, we calculate the absolute difference between its activation and the expected activation as the indicator in Eq.(2). While ideally we would just select the neuron with the largest magnitude, to ensure the loss function is differentiable, we use softmax as an alternative to the max operation. Here τ is the temperature used in the softmax function, which can be used to control the strictness of the operation, namely, a high temperature resembles the strict max operation (that has a one-hot output), whereas a lower temperature suggests smoother results.

$$w^{\text{activate}}(x) = \text{SoftMax}\left(\frac{\tau|\overrightarrow{f(x)} - \vec{\mu}|}{\vec{\sigma}}\right) \quad (2)$$

[2]We consider each element in a CNN feature map a separate neuron.

It is worth noting that this formula essentially corresponds to the intrinsic normalization in many models (e.g., those with batch normalization), where the statistics are readily accessible. When the subject model does not have any normalization layer, one can collect the statistics from the available samples.

To evaluate the impact of activations on classification results, we use the magnitude of gradients as an indicator. However, the gradients are not directly comparable at many layers, especially those that do not have normalization, because the varying magnitude of the activation values at those layers affects the gradients and makes direct comparison inaccurate. To tackle the problem, we introduce variables to denote normalized activations at all layers, called *proxy variables*, and derive gradients regarding those variables. The resulted gradients are hence comparable. Consider a proxy variable $h_i^*(x)$, where the neuron $f_i^*(x) = h_i^*(x) * \sigma_i$. Let l be the loss function for classification based on $f_i^*(x)$, we can thus calculate the gradient of proxy variable as follows.

$$g_i^*(x) = \frac{\partial l(f_i^*(x), ...)}{\partial h_i^*(x)} = \sigma_i \frac{\partial l(f_i^*(x), ...)}{\partial f_i^*(x)}.$$

We then select the neuron with the largest gradient, leveraging the aforementioned soft version of max operation.

$$w^{\text{impact}}(x) = \text{SoftMax}(\tau g^*(x)) \quad (3)$$

The next step is to identify the neurons satisfying both Eq.(2) and (3). In a discrete world, we could perform a set intersection. However, our softmax operations do not select the largest weight values. Instead, they produce vectors denoting the importance of all neurons. Hence, we perform a soft version of set intersection, which is differentiable and hence usable during training. Assume $w_i^{\text{activate}}(x),\ w_i^{\text{impact}}(x) \in [0, 1]$ denote the importance values for a neuron i, computed by the two respective softmax operations in Eq.(2) and (3), with 1 indicating "Important" and 0 "Unimportant". We can derive the neuron's overall importance as follows.

$$w_i(x) = w_i^{\text{activate}}(x) \wedge w_i^{\text{impact}}(x) = \sqrt{w_i^{\text{activate}}(x) \cdot w_i^{\text{impact}}(x)} \quad (4)$$

The operation $\wedge$ is essentially a soft version of "boolean conjunction". For example, when $a = 1$ and $b = 1$, $a \wedge b = 1$. The square root ensures the output is in the same magnitude as the inputs.

Difference from Neuron Pruning based Methods. There are existing works [Liu et al., 2018a, Wu and Wang, 2021] that determine a subset of neurons compromised by poisoning and simply remove those neurons. However, they usually lead to non-trivial benign accuracy degradation (see Section 5). The reason is that a neuron in a compromised model may be important for the classifications of both benign and poisoned inputs. In contrast, our method does not statically decide if a neuron is compromised. If a neuron is important

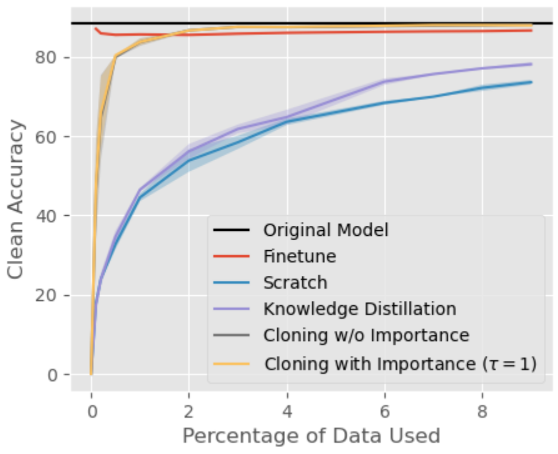

Figure 3: Benign accuracy of models after backdoor removal versus the data used (for a CIFAR-10 model poisoned with CleanLabel backdoor). Observe that knowledge distillation has a larger improvement over training from scratch given more samples. Also observe that training from scratch and knowledge distillation cause non-trivial benign accuracy degradation. In contrast, cloning causes negligible degradation of original backdoor model using even just 2% of data.

for both benign and poisoned samples, we (only) copy its behaviors for benign inputs.

4 FORMAL INTERPRETATION OF CLONING

The effectiveness of MEDIC hinges on cloning. In this section, we study a critical property of cloning, which states that *cloning can copy the functionalities of original model using only a small set of clean samples*. We also formally show that cloning is superior to training from scratch using the small set, and to knowledge distillation. That is, cloning tends to have a smaller loss value. Backdoor removal using importance values is merely an application of the property as we essentially leverage importance analysis to select a subset of functionalities only relevant to clean sample classifications for cloning. Figure 3 shows empirical evidence using an example.

We mainly utilize the *Rademacher complexity bound argument* [Bartlett and Mendelson, 2002] to derive an upper bound for the testing loss of a classifier with a high probability. Intuitively, *Rademacher complexity* [Bartlett and Mendelson, 2002] represents the complexity of a *function family* (i.e., the set of possible functions generated by a training algorithm). We hence show that cloning has a smaller upper bound (of loss) compared to distillation by analyzing their Rademacher complexities. We also show distillation tends to work better than training from scratch using a small set of samples, by studying the factors affecting their performance. In the following, we first introduce the notations, terms, and a set of lemmas. We then formally show that distillation is better than training, and then cloning is better than distillation.

Notations. We consider the original model f^*, model f^s trained from scratch using D_s (i.e., the small dataset), clone model f^c and model f^k by knowledge distillation [Hinton et al., 2015]. They belong to the function families $\mathcal{F}^*$, $\mathcal{F}^s$, $\mathcal{F}^c$, and $\mathcal{F}^k$, respectively. These families are determined by the specific optimization and learning algorithms.

Neural Net. To facilitate the formal analysis, we leverage a simple network. Let ψ be the ReLU activation function, l_γ the loss function with the Lipschitz constant 2γ. Without loss of generality, we assume a two-layer network, where the two layers are abstracted to two respective matrices, $G \in \mathcal{G} \subset \mathbb{R}^{d\times p}$ and $H \in \mathcal{H} \subset \mathbb{R}^{p\times C}$, with p the number of hidden units. Let s be the softmax function. The network is represented as $f(x) = s \circ o(x)$, $o(x) = Hg(x)$ and $g(x) = \psi(Gx)$. Putting them together, $f(x) = s(o(H\psi(Gx)))$. Intuitively, $f(x)$ outputs the probability, $o(x)$ is the logits value and $g(x)$ is the hidden layer.

As a common practice in multi-class analysis, we use the margin loss [Bartlett et al., 2017] as l_γ. A smaller loss value indicates better performance.

$$l_\gamma(f(x), y) = \phi_\gamma(f(x)_y - \max_{i\neq y} f(x)_i)$$

$$\text{where } \phi_\gamma(t) = \begin{cases} 1, & t < 0 \\ 1 - \gamma t, & 0 \le t \le 1/\gamma \\ 0, & t > 1/\gamma \end{cases}$$

A learning algorithm $\mathcal{A} : X^{|D|} \to \mathcal{H} \times \mathcal{G}$ is to learn the matrix H and G from data $D \sim \mathcal{S}$.

Radamacher Complexity. Let $\mathcal{F}$ be a family of functions , D a set of samples, and σ uniformly sampled from $\{-1, +1\}^{|D|}$. The empirical Rademacher complexity is defined as follows.

$$\hat{R}(\mathcal{F}|D) = \mathbb{E}_\sigma \sup_{f\in\mathcal{F}} \frac{1}{|D|} \sum_{x_i\in D} \sigma_i f(x_i).$$

Observe that if $\mathcal{F}$ contains only a fixed function, the value is 0 suggesting no complexity (and hence trivial to learn).

Covering Number Bound. It is difficult to directly compute Rademacher complexity in our context. We hence derive its upper-bound using *covering number*. Let $\|\cdot\|_{p,D}$ be a pseudo-norm on function family $\mathcal{F}$ with respect to a vector norm $\|\cdot\|_p$ and data D (definition in Appendix), we define the covering number $\mathcal{N}(\mathcal{F}, \|\cdot\|_{p,D}, \epsilon)$ as the size of minimal-ϵ cover of $\mathcal{F}$ under this pseudo-norm. Intuitively, this number means how many functions are representative such that they can cover all the learning outcomes with the ϵ bound. The covering number of a function family is dependent on the learning method. For example, when we include the loss to minimize the difference between $f^c \in \mathcal{F}^c$ and $f^* \in \mathcal{F}^*$ (during cloning), we essentially reduce the covering number of the function difference family $\mathcal{N}(\{f^c - f^*\}, \|\cdot\|_{p,D}, \epsilon)$.

Lemmas. In the following, we introduce three lemmas that are needed in later analysis. Their proofs can be found in the Appendix. In Lemma 1, we derive the Lipschitz constant of the loss function. In Lemma 2, we bound the Lipschitz of softmax function. In Lemma 3, we bound the Rademacher complexity by a function over the covering number.

Lemma 1. *For any logits $o_1, o_2 \in \mathbb{R}^c$ and $y \in Y$, we have*

$$|l_\gamma(o_1, y) - l_\gamma(o_2, y)| \leq 2\gamma ||o_1 - o_2||_\infty$$

Lemma 2. *For any logits $o_1, o_2 \in \mathbb{R}^c$, we have*

$$||s(o_1) - s(o_2)||_\infty \leq \frac{1}{2}||o_1 - o_2||_\infty$$

Lemma 3 (Rebeschini [2020])**.** *Given function family $\mathcal{F}$ and data D, we define the following function:*

$$B(\mathcal{F}|D) = \inf_{\epsilon \in [0, \epsilon_c/2]} \left\{ \epsilon + \frac{\sqrt{2}\epsilon_c}{\sqrt{n}} \sqrt{\log \mathcal{N}(\mathcal{F}, \|\cdot\|_{1,D}, \epsilon)} \right\}$$

where $\epsilon_c = \sup_{f \in \mathcal{F}} \|f\|_{2,D}$. We then have

$$\hat{R}(\mathcal{F}|D) \leq B(\mathcal{F}|D)$$

4.1 DISTILLATION VERSUS TRAINING WITH A SMALL SET

To understand why the distilled model f^d tends to perform better than model f^s trained from scratch, we study their loss values and their differences.

Using Lemma 1, the expected loss for distilled model f^k can be expressed as follows.

$$\begin{aligned}
&\mathbb{E}_{(x,y)\sim\mathcal{S}}[l_\gamma(f^k(x), y)] \\
=&\mathbb{E}_{\mathcal{S}}[l_\gamma(f^*(x), y)] + \mathbb{E}_{\mathcal{S}}[l_\gamma(f^k(x), y) - l_\gamma(f^*(x), y)] \\
\leq&\mathbb{E}_{\mathcal{S}}[l_\gamma(f^*(x), y)] + 2\gamma\mathbb{E}_{\mathcal{S}}(\|f^k(x) - f^*(x)\|_\infty) \\
\leq&\mathbb{E}_{\mathcal{S}}[l_\gamma(f^*(x), y)] + 2\gamma E_{\mathcal{S}}(\|f^k(x) - f^*(x)\|_2)
\end{aligned}$$

Note that as $|D_s| \ll |D|$, there likely exists a large loss gap between f^* (the original model) and f^s, denoted as $\Delta = E_{(x,y)\sim\mathcal{S}}[l_\gamma(f^s(x), y) - l_\gamma(f^*(x), y)]$. The advantage of f^k over f^s is denoted by their expected loss difference as follows.

$$\begin{aligned}
&\mathbb{E}_{(x,y)\sim\mathcal{S}}[l_\gamma(f^s(x), y) - l_\gamma(f^k(x), y)] \\
=&\Delta - \mathbb{E}_{\mathcal{S}}[(l_\gamma(f^k(x), y) - l_\gamma(f^*(x), y))] \quad (5) \\
\geq&\Delta - 2\gamma\mathbb{E}_{\mathcal{S}}(||f^k(x) - f^*(x)||_2)
\end{aligned}$$

Eq. (5) shows that the advantage of distillation depends on both the gap Δ and the similarity of models' outputs. A large gap and high similarity make distillation favorable. Note that in our context of copying functionalities, as f^s is trained on a very small dataset, the loss gap Δ tends to be large. Distillation uses an alternate loss, i.e., $||f^k(x) - f^*(x)||_2$, to improve output similarity, enlarging the advantage.

4.2 CLONING VERSUS DISTILLATION

To formally compare cloning and distillation, we study their upper-bounds of loss values. According to the standard Rademacher complexity bound argument [Bartlett and Mendelson, 2002], we have the following loss difference upper bound between the distilled model and the original model with a high probability $1 - \delta$.

$$\begin{aligned}
&E_{(x,y)\sim\mathcal{S}}[l_\gamma(f^k(x), y) - l_\gamma(f^*(x), y)] \\
\leq&\underbrace{\frac{1}{|D_s|}\sum_{(x,y)\in D_s}[l_\gamma(f^k(x), y) - l_\gamma(f^*(x), y)] +}_{\text{Training Error}} \quad (6) \\
&\underbrace{2\hat{R}(\mathcal{L}^k|D_s)}_{\text{Function Complexity}} + 3\underbrace{\sqrt{\frac{log(2/\delta)}{2|D_s|}}}_{\text{Uncertainty}}
\end{aligned}$$

Here, $\mathcal{L}^k$ is the function family of loss gap between the distilled model and the original model, with $\mathcal{L}^k = \{h : X \times Y \to \mathbb{R} \mid h(x, y) = l_\gamma(f^k(x), y) - l_\gamma(f^*(x), y)\}$. The loss upper bound between the clone model and the original model can be similarly derived. Observe that the bound consists of three terms: *training error*, *function complexity* and *uncertainty*. The first and the third terms have a similar effect for both the distilled model f^k and the clone model f^c. Specifically, the training error is computed on training data. Since we only have limited samples and the model is prone to overfit on the training data. This term is close to zero empirically for both models. The uncertainty only depends on the data size $|D_s|$ and the probability δ. Therefore, the difference between the two models mainly lies in their function family complexities. In the following, we will analyze the the upper-bounds of the two function families. We use $\mathcal{O}_i^k = \{h : \mathbb{R}^d \to \mathbb{R} \mid h(x) = o^k(x)_i - o^*(x)_i\}$ to denote the family of differences regarding the i-th logits.

Theorem 1. *For distilled model f^k and class number C, we have*

$$\begin{aligned}
&\hat{R}(\mathcal{L}^k|D_s) \\
\leq&\tilde{O}(\gamma\sqrt{C}) \max_i \hat{R}(\mathcal{O}_i^k|D_s) \quad (7) \\
\leq&\tilde{O}(\gamma\sqrt{C}) \max_i B(\mathcal{O}_i^k|D_s) \quad (8)
\end{aligned}$$

Proof. From Lemma 1 and Lemma 2, we know the Lipschitz constant of $l_\gamma \circ s$ is γ. Using the $\mathcal{L}_\infty$-contraction property of Rademacher complexity [Foster and Rakhlin, 2019], we can expand the complexity of loss function to the complexity of logits values in Eq. (7). Eq. (8) is from Lemma 3. □

Let $\|H\|_{1,\infty} = \max_i \sum_j |H_{i,j}|$ be the entry-wise matrix norm, $\mathcal{I}_i^c = \{h : \mathbb{R}^d \to \mathbb{R} \mid h(x) = g^c(x)_i - g^*(x)_i\}$ the family of differences on an intermediate neuron i (between the clone and original models), and $\mathcal{J}_i^* = \{h : \mathbb{R}^d \to \mathbb{R} \mid h(x) = g^*(x)_i\}$ the function family of intermediate neuron i of the original model. We have the following theorem.

Theorem 2. *For cloned model f^c, let $\omega_c = \sup_{H^c \in \mathcal{H}^c} \|H^c\|_{1,\infty}$, $\beta_c = \sup_{H^c \in \mathcal{H}^c, H^* \in \mathcal{H}^*} \|H^c - H^*\|_{1,\infty}$ and class number C. We have*

$$\hat{R}(\mathcal{L}^c|D_s) \leq \tilde{O}(\gamma\sqrt{C}) \min\Big[\max_i B(\mathcal{O}_i^c|D_s), \max_j \omega_c B(\mathcal{I}_j^c|D_s) + \beta_c B(\mathcal{J}_j^*|D_s)\Big]$$

Proof.

$$\begin{aligned}
&\hat{R}(\mathcal{L}^c|D_s) \\
\leq &\tilde{O}(\gamma\sqrt{C}) \max_i \hat{R}(\mathcal{O}_i^c|D_s) && (9)\\
\leq &\tilde{O}(\gamma\sqrt{C}) \max_i \hat{R}\big(\{[H^c(g^c(x) - g^*(x)) \\
&\qquad + (H^c - H^*)g^*(x)]_i\}\big) && (10)\\
\leq &\tilde{O}(\gamma\sqrt{C})\{\sup_{H^c} \|H^c\|_{1,\infty} \max_i \hat{R}(\mathcal{I}_i^c|D_s) \\
&\qquad + \sup_{H^c, H^*} \|H^c - H^*\|_{1,\infty} \max_i \hat{R}(\mathcal{J}_i^*|D_s)\} && (11)\\
\leq &\tilde{O}(\gamma\sqrt{C}) \Big[\omega_c \max_i B(\mathcal{I}_i^c|D_s) + \beta_c \max_i B(\mathcal{J}_i^*|D_s)\Big] && (12)
\end{aligned}$$

Eq. (9) is similar to that in Eq. (7). Eq. 10 is by expanding the functions in $\mathcal{O}_i^c$. Eq. (11) is the property of Rademacher complexity under linear transformation. Eq. (12) is from Lemma 3. Similar to Theorem 1, we have $\hat{R}(\mathcal{L}^c|D_s) \leq \tilde{O}(\gamma\sqrt{C}) \max_i B(\mathcal{O}_i^c|D_s)$. Together with Eq. (12), we finish the proof. □

From the two theorems, we can infer that the upper-bound for clone model is always smaller than that of the distilled model given $\mathcal{O}_i^k = \mathcal{O}_i^c$ (i.e., both models have their logits similar to the original model's). Intuitively, the advantage of cloning originates from the additional intermediate constraints, which reduces the covering number of $\mathcal{I}_i^c$, i.e. $\mathcal{N}(\mathcal{I}_i^c, \|\cdot\|_{1,D}, \epsilon)$ in $B(\mathcal{I}_i^c|D_s)$. Also note that the other family $\mathcal{J}_i^*$ is unrelated to cloning.

5 EXPERIMENT

Setting. We conduct experiments on seven representative backdoor attacks including Badnet [Gu et al., 2019b], Clean Label [Turner et al., 2018], SIG [Barni et al., 2019], Composite [Lin et al., 2020], Reflection [Liu et al., 2020], Polygon [NIST, 2019], and Filter [Liu et al., 2019] and three datasets including CIFAR-10 [Krizhevsky, 2009], and larger datasets Kitti-City [Fritsch et al., 2013] and Kitti-Road [Fritsch et al., 2013]. We compare with five latest backdoor removal methods including Finetune, Fineprune [Liu et al., 2018a], NAD [Li et al., 2021], MCR [Zhao et al., 2020a], and ANP [Wu and Wang, 2021]. The details of these attacks and removal methods can be found in Section 2 and Appendix B. We use 5% of training data in cloning.

Comparison with Baselines. In Table 1, we show the results in comparison with other backdoor removal methods. The third column shows the accuracy and attack success rate (ASR) of the original (trojaned) model. The following columns show the results after applying various backdoor removal methods. All the methods use the same set of data augmentations. We also adjust the temperature of MEDIC so that the test accuracies of our repaired models are comparable to others. Observe that the baselines are less effective in removing Clean Label, SIG, Composite, Polygon, and Filter backdoors as the ASRs are still high after the removal. In contrast, MEDIC can substantially reduce the ASRs of these attacks. For simple attacks like Badnet and Reflection where Finetune can already remove them, MEDIC has a similar

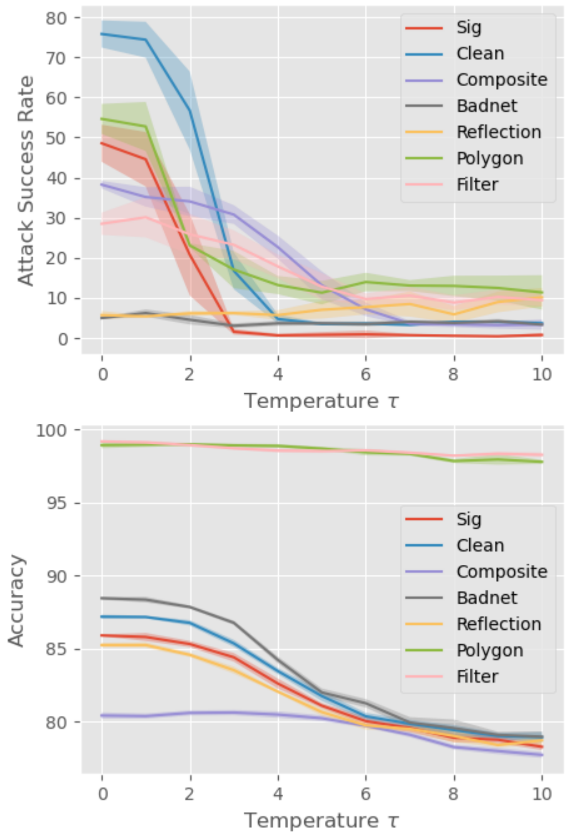

Figure 4: Effectiveness of importance. The shaded area represents standard deviation.

Table 1: Comparing with different methods on seven attacks. ± represents the standard deviation over 5 repeated runs.

Attack	Metric	Original (%)	Method (in percentage %)					
			Finetune	Fineprune	NAD	MCR	ANP	MEDIC
Clean Label	ASR	100.0	89.2 ± 20.2	99.6 ± 0.2	95.9 ± 7.2	90.3 ± 19.4	69.9 ± 17.6	$\mathbf{16.8 \pm 4.6}$
	Accuracy	88.5	85.2 ± 0.5	85.5 ± 0.3	85.3 ± 0.5	85.2 ± 0.4	80.6 ± 1.4	85.3 ± 0.2
SIG	ASR	94.5	68.3 ± 8.3	41.9 ± 23.9	55.8 ± 12.2	56.3 ± 9.0	73.9 ± 8.9	$\mathbf{1.5 \pm 0.7}$
	Accuracy	86.9	85.0 ± 0.2	85.0 ± 0.5	85.2 ± 0.2	84.6 ± 0.2	82.5 ± 0.8	84.4 ± 0.3
Badnet	ASR	100.0	3.2 ± 0.4	5.1 ± 1.6	3.0 ± 1.2	2.7 ± 0.4	$\mathbf{2.3 \pm 1.4}$	3.6 ± 0.6
	Accuracy	88.9	83.1 ± 0.3	84.3 ± 1.2	83.5 ± 0.7	83.6 ± 0.4	83.0 ± 1.5	84.2 ± 0.2
Composite	ASR	79.3	37.1 ± 2.6	36.4 ± 0.8	38.0 ± 3.1	9.4 ± 1.1	33.4 ± 11.3	$\mathbf{7.1 \pm 1.7}$
	Accuracy	83.0	80.2 ± 0.2	79.5 ± 0.5	79.8 ± 0.3	79.4 ± 0.2	76.7 ± 0.7	79.7 ± 0.1
Reflection	ASR	34.2	7.6 ± 1.9	5.7 ± 0.9	8.7 ± 2.2	$\mathbf{4.6 \pm 0.4}$	5.0 ± 4.2	6.2 ± 0.5
	Accuracy	84.3	84.5 ± 0.3	84.4 ± 0.2	84.1 ± 0.6	84.4 ± 0.3	80.3 ± 0.9	83.5 ± 0.2
Polygon	ASR	99.9	60.4 ± 14.6	47.9 ± 19.0	19.4 ± 10.3	39.4 ± 21.0	47.3 ± 8.1	$\mathbf{13.2 \pm 2.3}$
	Accuracy	99.7	99.0 ± 0.2	98.7 ± 0.6	95.7 ± 1.6	98.5 ± 0.5	98.2 ± 0.8	98.9 ± 0.1
Filter	ASR	100.0	62.0 ± 14.9	56.6 ± 9.8	47.9 ± 10.3	72.2 ± 5.5	19.5 ± 4.0	$\mathbf{17.8 \pm 3.0}$
	Accuracy	99.7	99.3 ± 0.3	99.1 ± 0.2	99.0 ± 0.3	99.4 ± 0.2	98.3 ± 0.4	98.6 ± 0.2
Avg. Drop	ASR	-	40.0%	45.0%	40.4%	47.6%	51.0%	**77.4%**
	Accuracy	-	2.1%	2.1%	2.2%	2.3%	4.4%	2.3%

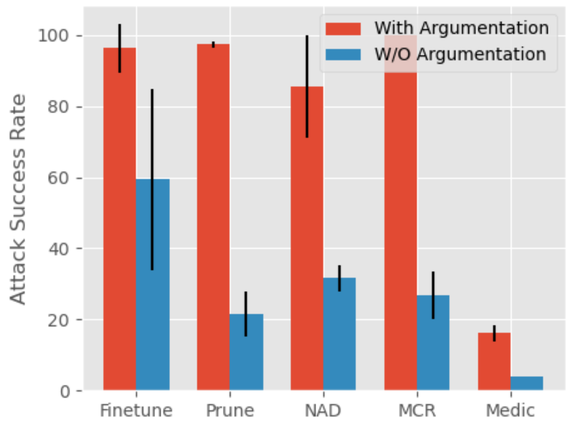

Figure 5: Removal methods on backdoor models trained with and without data augmentation. The black lines denote the standard deviations.

performance to other baselines. In the last row, we show the average ASR reduction and accuracy degradation. Observe that MEDIC achieves 25% more ASR reduction compared to others with minor accuracy degradation.

We also evaluate on an adaptive attack called composite attack, which utilizes multiple natural objects from different classes as the backdoor. Note that it represents the most adversarial context against MEDIC as the backdoor are essentially benign features where cloning would likely copy them. Observe that MEDIC is still effective under this attack as show in Table 1. The reason is that while benign features are used to compose the backdoor, these features are not important for the target class. As such, their behaviors are hence not copied for the target class.

Effectiveness of Using Importance Values. In Figure 4, we show the effects of cloning using different temperatures. A temperature $\tau = 0$ means we do not use the importance and treat all neurons as equal. Observe that a larger temperature prunes more backdoor-related behaviors. The ASR is reduced by at most 70% with at most 4% accuracy drop (with a temperature of 5).This shows our importance values capture the characteristics of benign functionalities.

Data Augmentation in Backdoor Removal. It is a common practice to use data augmentations, such as cropping and rotation, to achieve better benign accuracy. In our experiment, we find that the effectiveness of some baselines is affected by the data augmentations used. Figure 5 shows the effectiveness of different methods on backdoor models trained with and without data augmentations. The model is trojaned by a clean label attack. The red bars denote the ASRs after removal when augmentations are used during attack and the blue bars the ASRs when augmentations are not used. Observe that with augmentations, the baselines can hardly remove the backdoor. In contrast, MEDIC can effectively remove the backdoor with and without augmentations.

6 CONCLUSION

We propose a novel backdoor removal method by importance driven cloning. We empirically and theoretically show that our method is superior to state-of-the-art baselines on a large set of evaluated attacks.

References

Eugene Bagdasaryan and Vitaly Shmatikov. Blind backdoors in deep learning models. *arXiv preprint arXiv:2005.03823*, 2020.

Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *AISTATS*, pages 2938–2948, 2020.

Mauro Barni, Kassem Kallas, and Benedetta Tondi. A new backdoor attack in cnns by training set corruption without label poisoning. In *ICIP*, pages 101–105. IEEE, 2019.

Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *J. Mach. Learn. Res.*, 3:463–482, 2002.

Peter L. Bartlett, Dylan J. Foster, and Matus Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6240–6249, 2017.

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.

Siyuan Cheng, Yingqi Liu, Shiqing Ma, and Xiangyu Zhang. Deep feature space trojan attack of neural networks by controlled detoxification. In *AAAI*, 2021.

Edward Chou, Florian Tramer, and Giancarlo Pellegrino. Sentinet: Detecting localized universal attack against deep learning systems. *SPW*, 2020.

Dylan J. Foster and Alexander Rakhlin. $\mathcal{L}_\infty$ vector contraction for rademacher complexity. *CoRR*, abs/1911.06468, 2019.

Jannik Fritsch, Tobias Kuehnl, and Andreas Geiger. A new performance measure and evaluation benchmark for road detection algorithms. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2013.

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*, pages 113–125, 2019.

Yansong Gao, Bao Gia Doan, Zhi Zhang, Siqi Ma, Anmin Fu, Surya Nepal, and Hyoungshick Kim. Backdoor attacks and countermeasures on deep learning: A comprehensive review. *arXiv preprint arXiv:2007.10760*, 2020.

Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 2019a.

Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. Certified robustness of nearest neighbors against data poisoning attacks. In *AAAI*, 2020.

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

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Keita Kurita, Paul Michel, and Graham Neubig. Weight poisoning attacks on pre-trained models. In *ACL*, 2020.

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

Yiming Li, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *arXiv preprint arXiv:2007.08745*, 2020.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *CoRR*, abs/1606.09282, 2016.

Junyu Lin, Lei Xu, Yingqi Liu, and Xiangyu Zhang. Composite backdoor attack for deep neural network by mixing existing benign features. In *CCS*, 2020.

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*, pages 273–294. Springer, 2018a.

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

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 *CCS*, pages 1265–1282, 2019.

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

Michael McCoyd, Won Park, Steven Chen, Neil Shah, Ryan Roggenkemper, Minjune Hwang, Jason Xinyu Liu, and David Wagner. Minority reports defense: Defending against adversarial patches. In *International Conference on Applied Cryptography and Network Security*, pages 564–582. Springer, 2020.

Anh Nguyen and Anh Tran. Wanet–imperceptible warping-based backdoor attack. In *ICLR*, 2021.

Tuan Anh Nguyen and Anh Tran. Input-aware dynamic backdoor attack. *NeurIPS*, 33, 2020.

NIST. TrojAI Leaderboard. `https://pages.nist.gov/trojai/`, 2019.

Patrick Rebeschini. Lecture notes in algorithmic foundations of learning: Covering numbers bounds for rademacher complexity, 2020.

Shahbaz Rezaei and Xin Liu. A target-agnostic attack on deep models: Exploiting security vulnerabilities of transfer learning. In *ICLR*, 2020.

Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. Hidden trigger backdoor attacks. In *AAAI*, 2020.

Ahmed Salem, Rui Wen, Michael Backes, Shiqing Ma, and Yang Zhang. Dynamic backdoor attacks against machine learning models. *arXiv preprint arXiv:2003.03675*, 2020.

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.

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)*, 2021.

Guanhong Tao, Yingqi Liu, Guangyu Shen, Qiuling Xu, Shengwei An, Zhuo Zhang, and Xiangyu Zhang. Model orthogonalization: Class distance hardening in neural networks for better security. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022.

Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. Data poisoning attacks against federated learning systems. In *ESORICS*, pages 480–501, 2020.

Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. In *NeurIPS*, pages 8000–8010, 2018.

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

Akshaj Kumar Veldanda, Kang Liu, Benjamin Tan, Prashanth Krishnamurthy, Farshad Khorrami, Ramesh Karri, Brendan Dolan-Gavitt, and Siddharth Garg. Nnoculation: broad spectrum and targeted treatment of backdoored dnns. *arXiv preprint arXiv:2002.08313*.

Bolun Wang, Yuanshun Yao, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. With great training comes great vulnerability: Practical attacks against transfer learning. In *USENIX Security*, pages 1281–1297, 2018.

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*, pages 707–723, 2019.

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

Chong Xiang, Arjun Nitin Bhagoji, Vikash Sehwag, and Prateek Mittal. Patchguard: A provably robust defense against adversarial patches via small receptive fields and masking. In *30th USENIX Security Symposium (USENIX Security 21)*, 2021a.

Chong Xiang, Saeed Mahloujifar, and Prateek Mittal. Patchcleanser: Certifiably robust defense against adversarial patches for any image classifier. *arXiv preprint arXiv:2108.09135*, 2021b.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *CoRR*, abs/1612.03928, 2016a.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016b.

Xinyang Zhang, Zheng Zhang, and Ting Wang. Trojaning language models for fun and profit. In *European S&P*, 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 *CVPR*, 2020b.

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, 2019.

A PROOF

Lemma 1. *Let $l_\gamma(x,y) : \mathbb{R}^c \times Y \to \mathbb{R}$ and $\phi_\gamma(x) : \mathbb{R} \to \mathbb{R}$ where*

$$l_\gamma(x,y) = \phi_\gamma[f(x)_y - \max_{i \neq y} f(x)_i],$$
$$\text{where } \phi_\gamma(x) = \begin{cases} 1, & x < 0 \\ 1 - \gamma x, & 0 \leq x \leq 1/\gamma \\ 0, & x > 1/\gamma \end{cases} . \tag{13}$$

For any $x_1, x_2 \in R^c$ and $y \in Y$, we have

$$|l_\gamma(x_1,y) - l_\gamma(x_2,y)| \leq 2\gamma ||x_1 - x_2||_\infty \tag{14}$$

Proof. It is apparent that $\phi_\gamma(x) : \mathbb{R} \to \mathbb{R}$ is γ Lipschitz. And we have

$$\begin{aligned} &|l_\gamma(x_1,y) - l_\gamma(x_2,y)| \\ =&\Big|\phi_\gamma\Big[(f(x_1)_y - f(x_2)_y) - (\max_{i\neq y} f(x_1)_i - \\ &\max_{j \neq y} f(x_2)_j)\Big]\Big| \\ \leq&\gamma\big|(f(x_1)_y - f(x_2)_y) - (\max_{i\neq y} f(x_1)_i - \\ &\max_{j\neq y} f(x_2)_j)\big| \\ \leq&\gamma\big|(f(x_1)_y - f(x_2)_y)\big| + \gamma\big|\max_{i\neq y} f(x_1)_i - \\ &\max_{j\neq y} f(x_2)_j\big| \\ \leq& 2\gamma||f(x_1) - f(x_2)||_\infty \end{aligned} \tag{15}$$

□

Lemma 2. *Assume the softmax function $s : \mathbb{R}^c \to \mathbb{R}^c$ is defined as follows.*

$$s(x)_i = \frac{\exp(x_i)}{\sum_i \exp(x_i)} \tag{16}$$

Then we have

$$||s(x) - s(y)||_\infty \leq \frac{1}{2}||x - y||_\infty . \tag{17}$$

Moreover we have

$$\begin{aligned} &\big\|(s(x) - s(y)) - (s(x') - s(y'))\big\|_\infty \leq \\ &\frac{1}{2}(\|x - x'\|_\infty + \|y - y'\|_\infty) \end{aligned} \tag{18}$$

In other words, $s(x)$ and its residual form $s(x) - s(y)$ are both $1/2$-Lipschitz with respect to L_∞-norm.

Proof. From the definition of Lipschitz continualty, we have $|s(x) - s(y)|_\infty \leq L|x - y|_\infty$. Note that $|s(x) - s(y)|_\infty \leq \sup_i |s_(x)_i - s(y)_i|$.

The gradient is hence the following.

$$\frac{ds(x)_i}{dx_j} = \begin{cases} s(x)_i(1 - s(x)_i), & \text{i = j} \\ -s(x)_i s(x)_j, & \text{i} \neq \text{j} \end{cases} \tag{19}$$

Note that $||\nabla_i s(x)||_1 = 2s(x))_i(1 - s(x))_i) \leq 1/2$. Let $\preceq$ represents the generalized inequality of the nonnegative orthant. By the mean value theorem, for some δ, s.t. $x \preceq \delta \preceq y$, we have

$$s(x)_i - s(y)_i \leq \nabla_i s(\delta)^T(x - y) . \tag{20}$$

By Holder's inequality,

$$\nabla_i s(\delta)^T(x - y) \leq ||\nabla_i s(\delta)||_1 ||x - y||_\infty \leq \frac{1}{2}||x - y||_\infty \tag{21}$$

To prove the second goal, we can find that

$$\begin{aligned} &||(s(x) - s(y)) - (s(x') - s(y'))||_\infty \\ =&||(s(x) - s(x')) - (s(y) - s(y'))||_\infty \\ \leq&||(s(x) - s(x'))||_\infty + (s(y) - s(y')||_\infty \\ \leq&\frac{1}{2}(||x - x'|_\infty + ||y - y'||_\infty) \end{aligned} \tag{22}$$

□

Lemma 3 (Rebeschini [2020]). *We add the proof of this lemma here for completeness. Let $f : \mathbb{R} \to \mathbb{R} \in \mathcal{F}$ be a class of real-value functions, D be a set of samples. We define a pseudo-metric $\| \cdot \|_{p,D}$ on functions $\mathcal{F}$ with respect to vector norm $\| \cdot \|_p$ and n samples $|D| = n$, where*

$$\|f\|_{p,D} = \Big[\frac{1}{n}\sum_{x\in D} |f(x)|^p\Big]^{\frac{1}{p}} . \tag{23}$$

We define the Covering number $\mathcal{N}(\mathcal{F}, \| \cdot \|_{p,D}, \epsilon)$ as the size of the minimal ϵ-cover of $\mathcal{F}$ with respect to pseudo-norm $\| \cdot \|_{p,D}$.

Given

$$\sup_{f\in F} \|f\|_{2,D} \leq s_D , \tag{24}$$

we have

$$\hat{R}(\mathcal{F}|D) \leq \inf_{\epsilon\in[0,s_D/2]} \left\{\epsilon + \frac{\sqrt{2}s_D}{\sqrt{n}}\sqrt{\log \mathcal{N}(\mathcal{F}, \| \cdot \|_{1,D}, \epsilon)}\right\} \tag{25}$$

Proof. For any ϵ and D, let $\mathcal{C}$ be the minimal-ϵ cover of $\mathcal{F}$, which means for each function f, there exists $f_\epsilon \in \mathcal{C}$ such that $\|f - f_\epsilon\|_{1,D} \leq \epsilon$

$$\begin{aligned}
&\hat{R}(\mathcal{F}|D) \\
=&E_\sigma\left[\frac{1}{n}\sup_{f\in\mathcal{F}}\sum_i \sigma_i f(x_i)\right] \\
\leq&E_\sigma\left[\frac{1}{n}\sup_{f\in\mathcal{F}}\sum_i \sigma_i\left(f(x_i)-f_\epsilon(x_i)\right)\right]+ \\
&E_\sigma\left[\frac{1}{n}\sup_{f_\epsilon\in\mathcal{C}}\sum_i \sigma_i f_\epsilon(x_i)\right] \\
\leq&\epsilon + E_\sigma\left[\frac{1}{n}\sup_{f_\epsilon\in\mathcal{C}}\sum_i \sigma_i f_\epsilon(x_i)\right] \\
\leq&\epsilon + \sup_{f\in\mathcal{F}}\sqrt{\sum_{x_i\in D} f(x_i)^2}\,\frac{\sqrt{2\log|\mathcal{C}|}}{n}\text{(Massart's Lemma)} \\
\leq&\epsilon + \frac{\sqrt{2}s_D}{\sqrt{n}}\sqrt{\log\mathcal{N}(\mathcal{F},\|\cdot\|_{1,D},\epsilon)}
\end{aligned}$$

Since this bound stands for any ϵ, Thus we have

$$\hat{R}(\mathcal{F}|D) \leq \inf_{\epsilon\in[0,s_D/2]}\left\{\epsilon + \frac{\sqrt{2}s_D}{\sqrt{n}}\sqrt{\log\mathcal{N}\left(\mathcal{F},\|\cdot\|_{1,D},\epsilon\right)}\right\} \tag{26}$$

□

B EXPERIMENT DETAILS

In this section, we describe the details of our experiments. We use the original code from Badnet [Gu et al., 2019b], Clean Label attack [Turner et al., 2018], SIG [Barni et al., 2019], composite attack [Lin et al., 2020], Reflection attack [Liu et al., 2020], Polygon attack [NIST, 2019], and Filter attack [Liu et al., 2019] to construct backdoored models. For backdoor removal methods, we leverage the code from Model Connectivity Repair (MCR) [Zhao et al., 2020a], NAD [Li et al., 2021], ANP [Wu and Wang, 2021], and Fineprune [Liu et al., 2018a]. During the model training and testing, we use the exact same data augmentations, including resizing and cropping. Wide ResNet [Zagoruyko and Komodakis, 2016b] structure and CIFAR-10 [Krizhevsky, 2009] dataset are used for Clean Label, SIG, Badnet, Composite, and Reflection attacks. For polygon attack, we use ResNet34 [He et al., 2015] and Kitti-City [Fritsch et al., 2013] dataset. For filter attack, we use ResNet34 and Kitti-Road [Fritsch et al., 2013] dataset. We utilize 5% of CIFAR-10 training data and 0.5% of Kitti-City and Kitti-Road training data for the experiments. Note that we use a smaller number of data for large-scale datasets.

During cloning, we clone the outputs from all the convolution, normalization, and fully-connected layers in the network structure. These layers have learnable parameters where we aim to copy the functionalities from. We estimate the mean and standard deviation of internal activations using benign data. We use parameter λ to balance the cloning loss $\mathcal{L}_{\text{clone}}$ in Eq.(1) and the classification loss $\mathcal{L}_{\text{classification}}$ (i.e., cross-entropy loss) as follows. We set $\lambda = 10$ in this paper.

$$\mathcal{L}(x,y) = \mathcal{L}_{\text{classification}}(x,y) + \lambda\mathcal{L}_{\text{clone}}(x,y)$$

In the experiment, we train the model for 60 epochs over the small set of clean data. We use Adam optimizer with a initial learning rate $1\text{e}{-2}$ and apply weight decay of $1\text{e}{-4}$. We align the temperature τ in our method so that the clean accuracy of our model is comparable to others.