FEATURE GRINDING: EFFICIENT BACKDOOR SANITATION IN DEEP NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Training deep neural networks (DNNs) is expensive and for this reason, third parties provide computational resources to train models. This makes DNNs vulnerable to backdoor attacks, in which the third party maliciously injects hidden functionalities in the model at training time. Removing a backdoor is challenging because although the defender has access to a clean, labeled dataset, they only have limited computational resources which are a fraction of the resources required to train a model from scratch. We propose Feature Grinding as an efficient, randomized backdoor sanitation technique against seven contemporary backdoors on CIFAR-10 and ImageNet. Feature Grinding requires at most six percent of the model's training time on CIFAR-10 and at most two percent on ImageNet for sanitizing the surveyed backdoors. We compare Feature Grinding with five other sanitation methods and find that it is often the most effective at decreasing the backdoor's success rate while preserving a high model accuracy. Our experiments include an ablation study over multiple parameters for each backdoor attack and sanitation technique to ensure a fair evaluation of all methods. Models suspected of containing a backdoor can be Feature Grinded using limited resources, which makes it a practical defense against backdoors that can be incorporated into any standard training procedure.

1 INTRODUCTION

Deep neural networks (DNNs) are large and complex. Many systems deployed in the real world make use of DNNs, such as surveillance systems (Singh et al., 2018; Wang et al., 2017), self-driving cars (Bojarski et al., 2016; Dosovitskiy et al., 2017), and biometric authentication systems (Boles & Rad, 2017; Liu et al., 2018b). Training DNNs is expensive and for this reason, computation is often outsourced to third parties or publicly available, pre-trained DNNs are re-used. This convenience comes at the cost of security, as these third parties may act maliciously.

A critical security threat are *backdoor attacks*. Thereby, the third-party embeds hidden functionality into a *trojan* model that forces targeted misclassifications when a trigger is present in an input. The model functions normally for inputs without a trigger. In practice, backdoors can lead to crashes in self-driving cars (Versprille, 2015), surveillance systems with blind spots (Cooper, 2014), and biometric authentication systems granting access to unauthorized persons (Lovisotto et al., 2020).

Backdoor attacks and defenses are a well-studied subject in the field of secure machine learning. Backdoor attacks have the goal to remain effective by achieving a high attack success rate, being hard to detect and robust against model modification and sanitation. Existing backdoor attacks assume various capabilities of an attacker, such as (i) poisoning the training dataset (Gu et al., 2017; Liu et al., 2017; 2020; Pang et al., 2020a; Shokri et al., 2020), (ii) modifying the model's training code (Turner et al., 2018; Saha et al., 2020) or (iii) controlling the trojaned model's architecture (and parameters) (Hong et al., 2021; Yao et al., 2019; Tang et al., 2020).

Backdoor defenses aim to decrease the attack's success rate as much as possible to make the exploitation of a backdoor unreliable in practice. Thereby, the defender has access to a set of non-poisoned, *clean* data with ground-truth labels and is given a model suspected of containing a backdoor. Defenses can be deployed at the model's inference or training stage. Defenses deployed at inference time either pre-process inputs with the goal to render triggers unrecognizable (Cohen et al., 2019; Meng & Chen, 2017), or they run a detection algorithm for every input to predict whether it contains

a trigger (Chen et al., 2018; Udeshi et al., 2019). Defenses deployed during training either preemptively sanitize a model suspected of containing a backdoor (Liu et al., 2018a; Li et al., 2021), or they run a backdoor detection algorithm before sanitation (Wang et al., 2019; Guo et al., 2019).

Existing backdoor defenses are evaluated with a focus on their effectiveness at sanitizing a backdoor while maintaining the model's utility. We find that evaluating the defense's efficiency is often neglected. For example, the runtime of Neural Cleanse (Wang et al., 2019) scales proportionally with the number of classes, which can be feasible for 10 classes, but becomes infeasible for 1k classes. In practice, the motivation of a defender to engage with third parties and rely on their pre-trained models or computational resources is often rooted in a lack of adequate resources in the defender's control. Maintaining high-performance hardware may be more expensive than booking resources on-demand from third parties (Saiyeda & Mir, 2017). Pre-trained models are readily available online at low or no cost[1]. Defenses have to be executed in a trusted environment on resources available to the defender. The decision of whether to use a defense is bounded by the defender's available resources. We believe that a simple, minimal defense leveraging as few computational resources as possible while remaining effective is missing from related work.

We propose *Feature Grinding* as an efficient backdoor sanitation method. Feature Grinding requires low computational resources compared with four state-of-the-art backdoor sanitation approaches and achieves similar effectiveness. Our defense acts as a regularization method on the penultimate layer, also referred to as the *feature layer*, of the trojan DNN. The goal is to apply a transformation that increases the distance between predicted features of clean and trojan samples from the same target class. Feature Grinding requires only access to clean samples. Our experiments on the image classification datasets CIFAR-10 (Krizhevsky et al., 2009) and ImageNet (Deng et al., 2009) demonstrate that these transformations can be (i) learned quickly by the trojan model and (ii) that they sanitize backdoors effectively.

1.1 Contributions

In summary, we claim the following contributions.

1. We propose an efficient backdoor sanitation method called *Feature Grinding* that can be performed with limited computational resources.
2. Feature Grinding sanitizes trojan models from all seven surveyed backdoor attacks.
3. We conduct an extensive evaluation on the image classification datasets CIFAR-10 and ImageNet, comparing Feature Grinding with four other backdoor sanitation methods.
4. We propose a metric to compare sanitation methods called *Area Under the Pareto-Optimal Curve* (AUPOC). AUPOC shows the trade-off between the clean data accuracy (CDA) and the attack's success rate (ASR). We ablate over sets of parameters for each defense and compute the AUC for the best (i.e., Pareto-optimal) parameters given the CDA and ASR.

2 Related Work

2.1 Backdoors Attack

A deep neural network (DNN) contains a backdoor if an attacker can add a secret *backdoor trigger* to inputs presented to a *victim model*, which causes targeted misclassifications. These triggers are typically hidden (e.g., small or imperceptible) and are only known to the attacker. Existing backdoor attacks assume different capabilities of an attacker, which can be summarized as follows.

- **Poisoning**: The attacker can inject poisoned samples into the training dataset. In *clean label* attacks, the attacker can only poison inputs, but cannot control their target labels.
- **Training Code**: The attacker can modify the training code (e.g., the model's loss function).
- **Model Architecture**: The attacker has control over the victim model's architecture.

[1] `https://modelzoo.co`

We study seven contemporary backdoor attacks from related work. In this paper, we focus on attacks that assume the attacker can (i) poison the training data or (ii) modify the training code. The seven surveyed backdoor attacks from related work can be summarized as follows.

Badnet (Gu et al., 2017) assumes that an attacker can poison the training data, but not modify the training code. The authors propose injecting samples using static trigger patterns such as a white square with poisoned labels. Clean-Label (Turner et al., 2018) is the first poisoning attack that does not require changing the poisoned input's target labels. This makes it more difficult for the defender to remove poisoned inputs from their dataset before training the model. They stamp a nearly opaque trigger on inputs from the target class and adversarially perturb them to impede the victim model's ability to learn from the image's content and instead learn to associate the trigger with the target class. Refool (Liu et al., 2020) is a clean label attack that uses natural reflections as a trigger, blending other images with images from the target class, to embed their backdoor.

TrojanNN (Liu et al., 2017) requires control over the model's training code for fine-tuning the victim model. They generate a modified trigger pattern optimized for exciting a subset of the victim model's internal neurons and then fine-tune the model to embed a backdoor with these triggers. The Latent backdoor (Yao et al., 2019) has been designed to withstand the transfer learning process, in which a pre-trained, trojan teacher is fine-tuned for a related, but different task. They assume full control over the teacher's training process and add a loss term that encourages embedding the backdoor in the lower layers of the teacher model. The Adversarial Backdoor Embedding (ABE) method (Shokri et al., 2020) modifies the model's loss function during training to minimize the distance between poisoned and clean samples in the victim model's feature layer. The Input-Model Co-Optimization (IMC) method (Pang et al., 2020a) optimizes both the victim model and the trigger used to implement the backdoor. The authors formulate an objective to craft a backdoor in conjunction with a trojan model and use alternating optimization of the model and inputs.

2.2 Backdoor Defenses

The goal of a backdoor defense is to prevent an attacker from exploiting their backdoor by either suppressing it through input preprocessing or by sanitizing the model. Following the categorization by Pang et al. (2020b), there are four categories of backdoor defenses.

- **Input Reformation**: Each input to the victim model is pre-processed with the goal to disable a trigger by rendering it unrecognizable to the victim model.
- **Input Filtering**: A binary classification is made for each input to the victim model whether it contains a trigger. Inputs that are predicted to contain a trigger can be discarded.
- **Model Sanitation**: The victim model is modified with the goal of losing the backdoor's functionality while preserving the task's functionality. Sanitation is done preemptively without prior detection of a backdoor.
- **Model Inspection**: An extension of model sanitation methods that first discover and reverse-engineer backdoors before sanitizing the model.

We survey four model sanitation or inspection methods from related work. All methods assume access to (i) the trojan model's parameters and (ii) a subset of clean training data, but not to the backdoor's trigger patterns. Fine-Pruning (Liu et al., 2018a) iteratively prunes dormant neurons that are suspected to implement a backdoor. Neural Attention Distillation (NAD) (Li et al., 2021) is a method to quickly distill a student using the teacher's attention maps without retraining the student from scratch. Neural Cleanse (Wang et al., 2019) and TABOR (Guo et al., 2019) are model inspection methods that reverse-engineer a backdoor by an optimization process. The objective is to optimize inputs for a static trigger pattern that adversarially modifies the trojan model's prediction of any input towards a target class. Since the target class is unknown, both methods iterate through each candidate target class and generate at least one trigger per class. TABOR is an extension of Neural Cleanse that specifically allows reverse-engineering large and complex triggers. The authors propose two methods to remove reverse-engineered triggers from a model. The first approach uses *unlearning* in which the model is fine-tuned with the trigger and the ground-truth label. The second approach uses pruning, in which those internal neurons of the trojan model are removed with the highest activation for the reverse-engineered trigger pattern.

3 THREAT MODEL

Training a DNN model is expensive and hence computation is often outsourced to third parties. As input, the defender specifies (i) the training dataset including ground-truth labels, (ii) the training code and (iii) the model architecture. During training, the attacker (i) injects poisoned data into the training dataset and (ii) modifies the training code, but they cannot modify the trojan model's architecture. After training, the attacker sends the trojan model to the defender. The defender's objective is to sanitize any backdoors present in the model using limited computational resources.

The defender's primary goal is to sanitize the backdoor against a *targeted* attacker. A targeted attacker wins if the trojan model has a high *success rate* of predicting a trojan input with the target label. Given a trigger δ, a clean dataset D, a target label t and a model M, the attacker wins if the following condition holds for $\epsilon > 0$.

$$\underset{x \in D}{Pr}[M(x + \delta) = t] > 1 - \epsilon$$

The defender's secondary goal is to defend against a *non-targeted* attacker. A non-targeted attacker wins when the model has a high probability of misclassifying trojan inputs as any target class other than the ground-truth. Defending against non-targeted attacks is more challenging for the defender.

We believe that a defender with limited computational resources is a realistic and practically relevant assumption for multiple reasons. Prices for third party computational resources are decreasing and in many scenarios, it becomes more affordable to occasionally book external resources rather than continually maintaining dedicated hardware. Professional third party hardware often features high-tiered GPUs that allow for rapid development due to low training times. A defender may have the capabilities to run a few training steps themselves on their hardware. However, the aforementioned higher monetary costs and runtimes may deter the defender from performing backdoor sanitation and make them more inclined to accept the risk of a backdoor. The most effective backdoor sanitation approaches are useless if their runtime exceeds the defender's limits. Efficient backdoor sanitation strategies try to minimize the runtime, to allow acting on even a weak suspicion of a backdoor present in their model. We do not put hard constraints on the defender's computational power, but instead, compare sanitation methods by their execution time relative to the model's training time.

In summary, our attacker has access to the entire training dataset and its ground-truth labels, the model's training code, and significant computational resources. However, they cannot modify the model's architecture without the defender taking notice. Our defender has access to the entire training dataset and the ground-truth labels, the trojan model, but only limited computational resources.

4 FEATURE GRINDING

4.1 MOTIVATION

A DNN classifier is composed of a sequence of functions $\sigma(f_m(f_{m-1}(..f_1(x))))$, where f_i represents the i-th hidden layer, $\sigma(\cdot)$ is the softmax activation and $x \in \mathbb{R}^n$ represents the input. Feature grinding is applied to the penultimate layer $f_{m-1}(\cdot)$, of the DNN model, which we refer to as the model's *feature extractor*. The feature extractor of a model M is commonly referred to as $\phi_M(\cdot)$. This layer is a compressed representation of the input and the features form a high-dimensional *feature space*. It has been observed that distances between features are semantically meaningful for well-trained models with respect to the task (Zhang et al., 2018).

Inputs belonging to the same class typically form dense clusters in the feature space. A backdoor's objective is to map a trojan input to a feature vector located within the cluster composing the target class. The goal of a backdoor sanitation method is to *disentangle* the trojan samples from the cluster of the target class by increasing the separation between the clusters of clean and trojan samples. However, the defender does not know the trigger pattern. There are two conceptual approaches to achieve this disentanglement. Either, the trojan examples are reverse-engineered and then sanitized by projecting them to a different area in the feature space, or all clean examples are moved to a different region in the feature space.

Difficulties with the first approach (reverse-engineering) are that (i) formalizing complex triggers spanning multiple regions of the image may be difficult and (ii) it is computationally expensive to

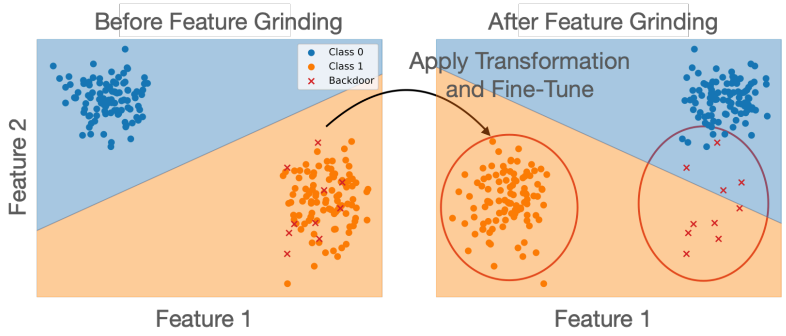

Figure 1: The motivation behind Feature Grinding illustrated at the example of a binary classification problem given two features. Before, the backdoor and clean samples from class 1 are co-located in the same region of the feature space. After Feature Grinding, the model has been fine-tuned on transformed features and the distance between clean and backdoored samples is higher.

optimize for an unknown trigger and target label. Once the trigger has been reverse-engineered, an advantage is that the trojan model can be updated with minimal side-effects through unlearning or pruning with the expectation that the model retains a high test accuracy. This idea motivates model inspection methods such as Neural Cleanse (Wang et al., 2019) and TABOR (Guo et al., 2019).

A different methodology, used in Feature Grinding, is to modify the trojaned model using only clean samples without attempting to reverse-engineer trigger patterns. This method is expected to have greater side-effects to the model, but it requires significantly fewer computational resources. For example, fine-pruning (Liu et al., 2018a) updates the trojaned model purely on its actions on clean data. It prunes neurons that are least active when clean data is passed through the model. The goal of Feature Grinding is to relocate all clean samples to a different region in the feature space, as illustrated in Figure 1. The hypothesis is that by moving clean samples, the trojaned samples retain their feature representation which disentangles them from the clean target samples.

4.2 Grinding and Restoration

The goal of Feature Grinding is to modify the model's feature extractor so that the updated, *grinded* model's feature extractor predicts transformed features. First, the defender resets the parameters of the *model's head*, which refers to the weights and biases of all layers that are on top of the model's feature extraction layer. Then, the defender records all feature activations of the training dataset and perturbs them by applying a static, randomized transformation function before fine-tuning the victim model on these transformed features.

Feature Grinding passes through two phases: *grinding* and *restoration*. In the grinding phase, the victim model is fine-tuned to predict the transformed features given clean samples as inputs. The model may lose some of its test accuracy during the grinding phase. As compensation, we use a restoration phase that focuses on regaining the model's test accuracy. Both phases can be incorporated into any standard training procedure by altering the model's loss function.

Assume the defender receives a trojan model M and wants to derive a model M' that has been sanitized of any backdoor present in M. When using Feature Grinding, the defender adds a term L_f to the model's loss during the grinding and restoration phases. L_f can be described as follows, given some transformation $T(\cdot)$, the feature extractor $\phi_M(\cdot)$ and some input x for model M.

$$L_f(x) = \|\phi_{M'}(x) - T(\phi_M(x))\|$$

The total loss L for both phases is a sum of the task loss L_t and the Feature Grinding loss L_f. We use a parameter $\alpha \in \mathbb{R}$ to trade-off both loss terms. In our experiments, we use $\alpha = 0.8$ during the grinding phase and $\alpha = 0.2$ during the restoration phase.

$$L(x) = \alpha L_f(x) + (1 - \alpha) L_t(x)$$

4.3 TRANSFORMATION

The transformation function is used to perturb the feature space of the victim model. Choosing a transformation function influences the efficiency of Feature Grinding. An effective transformation ensures that the success rate of the backdoor is decreased as much as possible (e.g., by retraining the entire model). An efficient transformation adds an additional constraint by trying to minimize the resources spent, i.e., the number of steps needed for the victim model to learn the transformed feature space, while remaining effective at sanitizing the backdoor.

Proposed Transformations. We design transformation functions that follow our intuition on achieving high efficiency. We experiment with the following transformation functions.

1. **Permute** (T_p): The perturbation consists of a random permutation of all features. The permutation is sampled randomly once and then applied to all features.
2. **Rotate** (T_r): The features are rotated in the n-dimensional space by sampling random rotation matrices using the approach of Stewart (Stewart, 1980).
3. **Rotate-2d** (T_{r_2})): The features are rotated in a randomly sampled, two dimensional plane by an angle θ.

All surveyed transformations are (i) sampled randomly once and (ii) they are automorphisms. Randomization is important because the applied transformation must be kept secret from the adversary to avoid them from adapting their backdoor to the transformation. The defender should post-hoc dispose of their records about the applied transformation. We believe that transformations which are automorphisms are advantageous because they preserve the structure of the high-dimensional space which shortens the restoration phase.

5 EVALUATION

In this section, we present our experimental setup and describe our evaluation criteria including the proposed AUPOC metric. Then we show the performance of Feature Grinding against seven contemporary backdoor attacks. We compare Feature Grinding with four other contemporary backdoor sanitation methods: Fine-Pruning (FP) (Liu et al., 2018a), Neural Cleanse (NC) (Wang et al., 2019), TABOR (Guo et al., 2019) and Neural Attention Distillation (NAD) (Li et al., 2021).

5.1 SETUP

In this section, we describe the setup for our experiments.

Hardware and Software. We perform all experiments on a local machine with a single A100 GPU with 40 GByte of VRAM and an AMD EPYC 7302 16-Core Processor. We conduct our experiments using PyTorch 1.9.0 and the trojanvision (Pang et al., 2020b) package version 1.0.10 that implements many of the surveyed backdoor attacks and defenses.

Datasets. We experiment with the following two standard image classification datasets.

- **CIFAR-10** (Krizhevsky et al., 2009): Contains 50k training and 10k testing images with a dimension of 32x32 pixels and 10 class labels.
- **ImageNet** (Deng et al., 2009): Contains 1.23m training and 50k testing images with 1k class labels and we resize and center-crop the images to 256x256 pixels.

Network Architectures. We use standard training procedures for training a ResNet-18 (He et al., 2016) on CIFAR-10. The model is trained for 120 epochs with a learning rate initialized at 0.1. We decrease the learning rate by a factor of 10 when the model's loss does not improve for two epochs and we use random cropping and cutout as data augmentation strategies. Our clean model achieves 96.06% test accuracy which is similar to the value reported in the original ResNet paper. For ImageNet, we rely on a pre-trained ResNet-50 that is made publicly available through the torchvision package[2]. The model has a test accuracy of 76.15%.

[2] `https://pytorch.org/vision/stable/models.html`

	Clean	Fine-Pruning	Neural Cleanse	TABOR	NAD	Feature Grinding
CIFAR-10	34	2	12	14	4	2
ImageNet	4200	65	800	1133	130	68

Table 1: Runtimes are shown in minutes for each backdoor defense. We evaluate the runtime optimized versions of Neural Cleanse and TABOR for ImageNet, in which only classes with a top-100 anomaly index are fully scanned. 'Clean' represents the training time from scratch.

Training Time. We measure the total training time of a clean model on CIFAR-10 and ImageNet as a point of reference to interpret the efficiency of a backdoor defense. For CIFAR-10, we observe that a model can be trained in 34 minutes on our hardware. For ImageNet, we measure the runtime for a single epoch and estimate the model's total training time to be 70 hours for 120 training epochs. Table 1 shows the runtime for each defense. We observe that Feature Grinding is more than $11\times$ faster than the runtime optimized version of Neural Cleanse and $16\times$ faster than TABOR.

Evaluation Metrics. In summary, we empirically measure the following four metrics.

- **CDA** (Clean Data Accuracy): The accuracy of the model on an unseen test dataset.
- **ASR** (Attack Success Rate): The rate at which the trojan model predicts the target label for malicious inputs that contain the backdoor trigger.
- **ARR** (Attack Recovery Rate): The rate at which a trojan model predicts the ground-truth label for malicious inputs that contain the backdoor trigger.
- **Runtime**: The runtime of a defense on our hardware.

AUPOC. We want to compare the effectiveness of two defenses by their CDA and ASR. There is an apparent correlation between both metrics, where a low CDA predicts a low ASR. For example, assume the defender randomly assigns all weights of the victim model (CDA is equivalent to random guessing), then the ASR is expected to be no higher than random guessing as well. For a fair comparison between defenses, we derive a combined value from the CDA and ASR that allows a pairwise comparison of backdoor defenses. We propose using the *Area Under the Pareto-Optimal Curve* (AUPOC) that we record by ablating over multiple sets of parameters for each defense.

For a given defense and attack, the AUPOC can be derived as follows. We identify a set of parameters for the defense to include in an ablation study. Then, for each set of parameters, we record the ASR and CDA as a single data point. We draw the *pareto-frontier* between points that are Pareto-optimal, i.e., those points for which no other points exist that have a lower ASR and a higher CDA. To achieve closure of the curve, we add a point at an ASR of 1.0 with the highest CDA of the Pareto-frontier. Let $f(x)$ be the piece-wise function connecting all the points in the Pareto-frontier, then the AUPOC is simply the integral over that curve.

$$\text{AUPOC} = \int_0^1 f(x)\,\mathrm{d}x$$

Note that a 'perfect' AUPOC of 1.0 means that the defense achieves a CDA of 1.0 at an ASR of 0.0. This is unattainable if the clean model's CDA is lower than 1.0, otherwise, the defense would improve the clean model. AUPOC is a relative measure to compare defenses under the same conditions. It is not useful as an absolute measure due to its sensitivity to the clean model's CDA.

Parameter Ablation. Computing the AUPOC for each defense against each backdoor attack requires ablating over multiple sets of parameters for each defense. We rely on the parameters proposed in the author's paper for all surveyed backdoor defenses. Since we are interested in comparing efficiency, we keep the runtime constant and ablate over parameters such as the learning rate, rather than the number of epochs. For Fine-Pruning, we ablate over the pruning rate $\rho \in [0.05, 0.99]$. We optimize triggers in Neural Cleanse and TABOR for ten epochs and ablate over the learning rate $\alpha \in [0.00002, 0.00001]$ used for unlearning a trigger pattern. For NAD, we train the student and teacher model for five epochs each and ablate over the learning rate $\alpha \in [0.0001, 0.001]$. We use the same cutout data augmentation strategy as the authors. We run Feature Grinding for five epochs and ablate over the number of epochs spent in the grinding stage $e \in \{2, 4\}$ and the three transformations proposed in Section 4.2. All results for CIFAR-10 are computed as the mean value

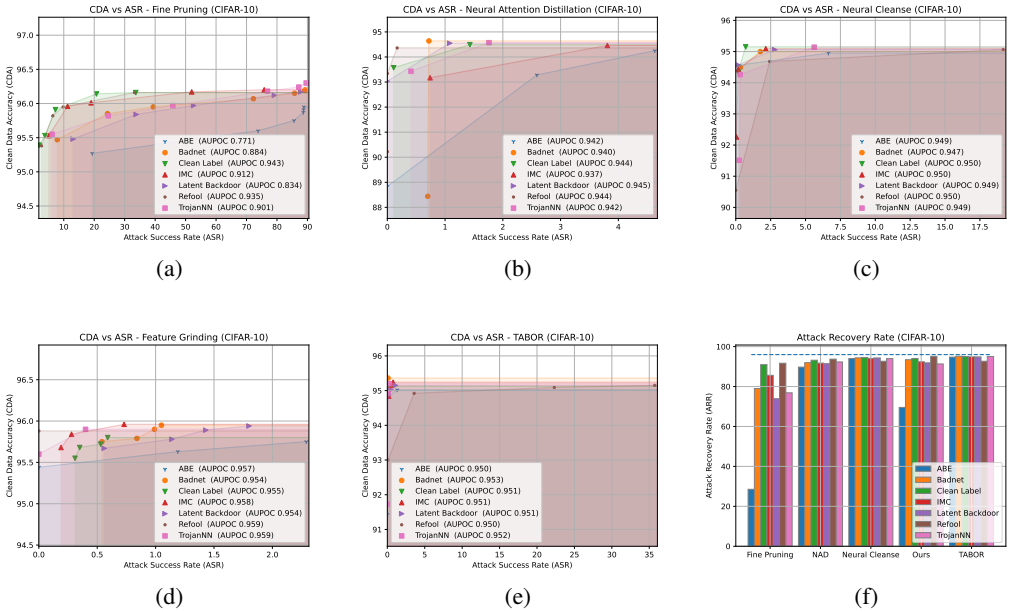

Figure 2: Figures (a-e) show the Area Under the Pareto-Optimal Curve (AUPOC) for each surveyed backdoor sanitation method versus each surveyed backdoor attack on CIFAR-10. A defense with a higher AUPOC is more effective. Figure (f) shows the highest Attack Recovery Rate (ARR) for each backdoor defense and the dashed line indicates the CDA. Each data point represents the mean value over three repetitions with the defense's parameters.

over three repetitions and a single repetition for ImageNet (due to higher computational demands). Since defenses on ImageNet do not require the entire training data, we give each defense access to a random subset of 100k clean training samples rather than the whole dataset of 1.23m records.

5.2 Effectiveness

Our goal is to measure the effectiveness of each backdoor attack against each defense. Figures 2 and 3 show the Pareto-optimal Curves and AUPOC metrics for all five defenses. The plot shows the ASR on the x-axis in relation to the CDA and we show only data points that are members of the Pareto frontier. Defenses with a higher AUPOC are more effective.

CIFAR-10: Figure 2 show the results for CIFAR-10. We observe that all defenses are effective at sanitizing a majority of the backdoors. Fine-Pruning has a relatively low AUPOC of less than 0.9 against the ABE, Latent Backdoor and Badnet attacks. For example, the best (i.e., lowest) ASR against the ABE backdoor is about 20 percent, which is relatively high compared to other sanitation methods such as Feature Grinding, which achieves a best ASR of 0 percent. The remaining defenses sanitize all backdoors effectively with an AUPOC of at least 0.937. Feature Grinding performs has the highest AUPOCs out of all defenses against every backdoor attack. NAD effectively sanitizes the backdoors but the models CDAs are about 1 percent lower compared to Feature Grinding.

ImageNet: Figure 3 show the results for ImageNet. We observe that it is more difficult to remove backdoors from ImageNet models than from CIFAR-10 models. Fine-Pruning achieves a low AUPOC of less than 0.5 against the Latent Backdoor and the Refool attack. Similarly, NAD has a low effectiveness against the Badnet and IMC attacks. For example, the best ASR that NAD achieves against IMC is still about 57 percent. Neural Cleanse and TABOR are completely ineffective against the Refool attack with an AUPOC of 0.021 and 0.030, which is expected because Refool applies the trigger pattern across the entire image. Feature Grinding achieves relatively high AUPOCs against each backdoor attack. We measure the lowest AUPOC of 0.585 against the IMC attack, meaning

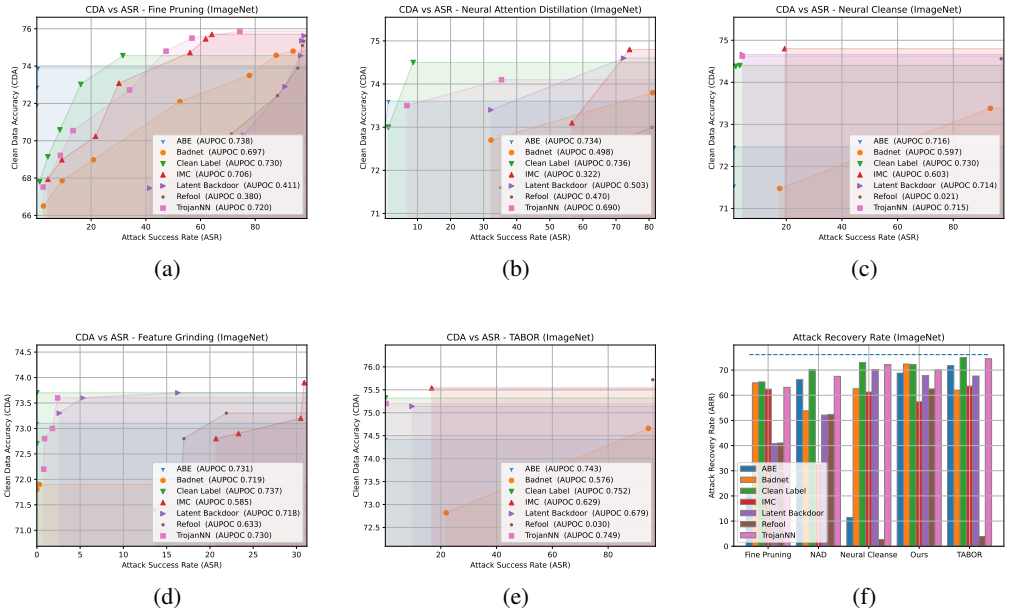

Figure 3: Figures (a-e) show the Area Under the Pareto-Optimal Curve (AUPOC) for each surveyed backdoor sanitation method versus each surveyed backdoor attack on ImageNet. A defense with a higher AUPOC is more effective. Figure (f) shows the Attack Recovery Rate (ARR) for each backdoor defense for the data point with the highest ARR and the dashed line indicates the CDA.

that the best (i.e., lowest) ASR is about 22 percent. Overall, we observe that Feature Grinding has the best worst-case performance against all backdoor attacks compared to all other defenses.

5.3 Attack Recovery

In this experiment, we compare the attack recovery rates (ARR) for each defense. The results for CIFAR-10 and ImageNet are shown in Figures 2f and 3f. The bar charts show the defense on the x-axis and the ARR on the y-axis and the dashed horizontal line represents the clean model's CDA. A high ARR indicates that the sanitized model predicts the correct, ground-truth class for a backdoored input and shows the defense's effectiveness against a non-targeted attacker.

We observe that for both datasets, Feature Grinding has a high ARR against every attack. This supports the hypothesis stated in Section 4.1 that Feature Grinding successfully disentangles clean and backdoored samples. For CIFAR-10, Neural Cleanse and TABOR have the highest ARRs, followed by Feature Grinding and NAD. Fine-Pruning has a significantly lower ARR against multiple backdoors. On ImageNet, Feature Grinding has similar ARRs as Neural Cleanse and TABOR, except for Refool against which Feature Grinding has a significantly higher ARR than all other defenses.

6 Conclusion

We proposed Feature Grinding as an efficient backdoor sanitation method that can be performed using a low amount of computational resources. Our experiments on the image classification datasets CIFAR-10 and ImageNet have shown that Feature Grinding is highly efficient and can sanitize all seven surveyed backdoors. On ImageNet, Feature Grinding is approximately $11\times$ faster than Neural Cleanse and about $16\times$ faster than TABOR with a similar (or better) effectiveness. We propose the AUPOC metric to fairly evaluate the effectiveness of backdoor sanitation methods. Our evaluation shows that other fast sanitation methods from related work, such as Fine Pruning and Neural Attention Distillation, do not achieve the same AUPOC as Feature Grinding against all backdoor attacks. We hope that our work leads to further research into efficient backdoor sanitation methods.

REFERENCES

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Andrew Boles and Paul Rad. Voice biometrics: Deep learning-based voiceprint authentication system. In *2017 12th System of Systems Engineering Conference (SoSE)*, pp. 1–6. IEEE, 2017.

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. *arXiv preprint arXiv:1811.03728*, 2018.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pp. 1310–1320. PMLR, 2019.

Paul Cooper. Meet aisight: The scary cctv network completely run by ai. 2014. URL `http://www.itproportal.com/`.

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

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pp. 1–16. PMLR, 2017.

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

Wenbo Guo, Lun Wang, Xinyu Xing, Min Du, and Dawn Song. Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems. *arXiv preprint arXiv:1908.01763*, 2019.

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

Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. Handcrafted backdoors in deep neural networks. *arXiv preprint arXiv:2106.04690*, 2021.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). 2009. URL `http://www.cs.toronto.edu/~kriz/cifar.html`.

Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Neural attention distillation: Erasing backdoor triggers from deep neural networks. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=9l0K4OM-oXE`.

Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pp. 273–294. Springer, 2018a.

Yi Liu, Jie Ling, Zhusong Liu, Jian Shen, and Chongzhi Gao. Finger vein secure biometric template generation based on deep learning. *Soft Computing*, 22(7):2257–2265, 2018b.

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

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

Giulio Lovisotto, Simon Eberz, and Ivan Martinovic. Biometric backdoors: A poisoning attack against unsupervised template updating. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 184–197. IEEE, 2020.

Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pp. 135–147, 2017.

Ren Pang, Hua Shen, Xinyang Zhang, Shouling Ji, Yevgeniy Vorobeychik, Xiapu Luo, Alex Liu, and Ting Wang. A tale of evil twins: Adversarial inputs versus poisoned models. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 85–99, 2020a.

Ren Pang, Zheng Zhang, Xiangshan Gao, Zhaohan Xi, Shouling Ji, Peng Cheng, and Ting Wang. Trojanzoo: Everything you ever wanted to know about neural backdoors (but were afraid to ask). *arXiv preprint arXiv:2012.09302*, 2020b.

Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. Hidden trigger backdoor attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 11957–11965, 2020.

Anam Saiyeda and Mansoor Ahmad Mir. Cloud computing for deep learning analytics: A survey of current trends and challenges. *International Journal of Advanced Research in Computer Science*, 8(2), 2017.

Reza Shokri et al. Bypassing backdoor detection algorithms in deep learning. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 175–183. IEEE, 2020.

Amarjot Singh, Devendra Patil, and SN Omkar. Eye in the sky: Real-time drone surveillance system (dss) for violent individuals identification using scatternet hybrid deep learning network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1629–1637, 2018.

Gilbert W Stewart. The efficient generation of random orthogonal matrices with an application to condition estimators. *SIAM Journal on Numerical Analysis*, 17(3):403–409, 1980.

Ruixiang Tang, Mengnan Du, Ninghao Liu, Fan Yang, and Xia Hu. An embarrassingly simple approach for trojan attack in deep neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 218–228, 2020.

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

Sakshi Udeshi, Shanshan Peng, Gerald Woo, Lionell Loh, Louth Rawshan, and Sudipta Chattopadhyay. Model agnostic defence against backdoor attacks in machine learning. *arXiv preprint arXiv:1908.02203*, 2019.

Allyson Versprille. Researchers hack into driverless car systems, take control of vehicle. 2015. URL `https://www.nationaldefensemagazine.org/articles/2015/5/1/2015may-researchers-hack-into-driverless-car-system-take-control-of-vehicle`.

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

Ya Wang, Tianlong Bao, Chunhui Ding, and Ming Zhu. Face recognition in real-world surveillance videos with deep learning method. In *2017 2nd international conference on image, vision and computing (icivc)*, pp. 239–243. IEEE, 2017.

Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y Zhao. Latent backdoor attacks on deep neural networks. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2041–2055, 2019.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.