Gym: A Comprehensive Defense Approach against DNN Backdoor Attacks

Yi Zeng[1],Han Qiu[2], Shangwei Guo[3], Tianwei Zhang[3], Meikang Qiu[4], Bhavani Thuraisingham[5]

[2]Telecom Paris, Institut Polytechnique de Paris, Palaiseau, France, 91120.
[1]University of California San Diego, CA, USA, 92122.
[3]Nanyang Technological University, Singapore, 639798.
[4]Columbia University, New York, USA, 10027.
[5]The University of Texas at Dallas, Texas, USA, 75080.

Abstract

Public resources and services (e.g., datasets, training platforms, pre-trained models) have been widely adopted to ease the development of deep learning-based applications. However, if the third-party providers are untrusted, they can inject poisoned samples into the datasets or embed backdoors in those models. Such an integrity breach can cause severe consequences, especially in safety- and security-critical applications. Various backdoor attack techniques have been proposed for higher effectiveness and stealthiness. Unfortunately, existing defense solutions are not practical to thwart those attacks in a comprehensive way.

In this paper, we propose Gym, a novel and effective defense solution to defeat different types of backdoor attacks and enhance DL models' robustness. The key innovations of our approach are two preprocessing functions: (1) an intensive function is used to transform clean images for fine-tuning of the infected model. This can invalidate the effects of embedded backdoors; (2) a lightweight function is adopted to invalidate triggers during inference. The combination of these two functions in two stages can achieve reliable and comprehensive protection of backdoored models. Extensive experiments show that our solution can effectively mitigate six different kinds of backdoor attacks and outperform four state-of-the-art defense solutions for various DNN models and datasets.

Introduction

The past several years have witnessed the rapid development of Deep Learning (DL) technology. Various DL models today are widely adopted in many scenarios, e.g., image classification (Chan et al. 2015), speech recognition (Deng and Platt 2014), natural language processing (Collobert and Weston 2008). These applications significantly enhance the quality of life and work efficiency. With the increased complexity of Artificial Intelligence tasks, more sophisticated DL models need to be trained, which require large-scale datasets and a huge amount of computing resources.

To reduce the training cost and effort, it is now common for developers to leverage third-party resources and services for efficient model training. Developers can download state-of-the-art models from the public model zoos or purchase them from model vendors. They can also download or purchase valuable datasets from third parties and train the models by themselves. A more convenient way is to utilize public cloud services (e.g., Amazon SageMaker (Liberty et al. 2020), GoogleVision AI (Hosseini, Xiao, and Poovendran 2017), Microsoft Computer Vision (Han et al. 2013), etc.), which can automatically deploy the training environment and allocate hardware resources based on users' demands.

However, new security threats are introduced to DNN models when the third party is not trusted. One of the most severe threats is the DNN backdoor attacks (Li et al. 2020a): the adversary injects a backdoor into the victim model, causing it to behave normally over benign samples, but predict the samples with an attacker-specified trigger as wrong labels desired by the adversary. Typically, a backdoor injection can be achieved by directly modifying the neurons (Liu et al. 2017) or poisoning the training datasets (Gu, Dolan-Gavitt, and Garg 2017). In practice, the developer may obtain a poisoned dataset if the source is untrusted. It is hard to detect such a threat as a very small ratio of malicious samples can lead to a backdoored model. When the developer outsources the model training task to an untrusted cloud provider, the adversary can inject the backdoor by either dataset poisoning or parameter modifications. It will then be difficult for the developer to detect the existence of backdoors, as the model only has anomalous predictions on samples with triggers, which are agnostic to the developer.

It is of paramount importance to have an effective method to address these severe threats. Past works proposed some approaches to detect the existence of backdoors or eliminate them from the infected models. Unfortunately, most of them have certain limitations. First, some solutions require the defender to have poisoned data samples (Chen et al. 2018; Du, Jia, and Song 2019), or knowledge of the attack techniques (Xu et al. 2019) and triggers (Chou et al. 2018). This assumption is not held when the defender is only given the victim model. Second, these solutions are not comprehensive to cover all different types of attack techniques and trigger patterns. For instance, (Wang et al. 2019) is effective against the single target attack but fails to identify the all-to-all attack where there is more than one target label for the malicious samples (Gu, Dolan-Gavitt, and Garg 2017). We will empirically validate this in our evaluation section. (Liu et al. 2019) cannot defeat attacks with complex trigger patterns (e.g., watermarking in the background), as claimed in that paper. More importantly, most of these defense works only consider traditional backdoor attacks, while ignoring the recently discovered advanced attacks (e.g., invisible backdoor attacks (Li et al. 2019)). As one of our contributions, we will categorize past defense solutions and analyze their limitations in this paper.

Motivated by the gap between the severity of backdoor attacks and the limitations of existing solutions, we propose GYM, a novel and comprehensive method to defeat various DNN backdoor attacks. A successful backdoor attack relies on both the backdoor in the infected model and effective triggers hidden in the malicious samples. Thus, the key idea of our approach is the integration of model fine-tuning (which is used to weaken the effects of the backdoor) and input preprocessing (which is used to affect the impact of triggers). Given an infected model, our solution has two steps. During the fine-tuning phase, it retrains the model for a few epochs using some intensive preprocessed clean data samples[1]. During the inference phase, each data sample (either clean one or trigger-patched one) is first transformed by a lightweight preprocessing function and then fed into the fine-tuned model for prediction. With these two steps, the model will correct the labels of malicious samples while maintaining high performance for normal data.

Our method is effective against various backdoor attacks and trigger patterns and does not need any prior knowledge about the attack techniques or poisoned samples. We conduct comprehensive evaluations to validate our solution. We consider different datasets (Cifar10, GTSRB, PubFig) and models (ResNet-18, LeNet-8, VGG-16). We implement different attack techniques (BadNet (Gu, Dolan-Gavitt, and Garg 2017), Neural Trojan (Liu et al. 2017), invisible backdoor (Li et al. 2019)), different triggers (Square, watermark, adversarial perturbation) and different modes (single target, all-to-all). Our method can successfully defeat all these threats. Evaluations also show that our method can outperform four state-of-the-art works (Neural Cleanse (Wang et al. 2019), Fine-pruning (Liu, Dolan-Gavitt, and Garg 2018), FLIP, and ShrinkPad-4 (Li et al. 2020b)) in defeating different backdoor attacks and enhancing the model robustness.

Background about Backdoor Attacks

Given a DNN model f_θ with parameters θ, a backdoor attack can be formulated as a tuple $(\Delta\theta, \delta)$, where $\Delta\theta$ is the backdoor injected by the adversary to the model parameters, and δ is an attacker-specified trigger. Then the compromised model $f_{\theta+\Delta\theta}$ still has state-of-the-art performance for normal samples: $f_{\theta+\Delta\theta}(x) = f_\theta(x), \forall x \in \mathcal{X}$. However, for an input sample containing the trigger, the model will predict a label different from the correct one: $y' = f_{\theta+\Delta\theta}(x+\delta) \neq f_{\theta+\Delta\theta}(x), \forall x \in \mathcal{X}$. y' can be a fixed label pre-determined by the attacker, or an arbitrary unmatched label.

The adversary has multiple ways to embed the backdoor into the DNN model. (1) Data poisoning (Gu, Dolan-Gavitt, and Garg 2017; Chen et al. 2017): the adversary generates a number of poisoned samples with the desired labels and incorporates such samples into the clean training set to train a backdoor model. (2) Direct modification (Liu et al. 2017): the adversary can select critical neurons and weights for modification via model retraining. (3) Transfer learning (Yao et al. 2019): if the adversary injects backdoor into a teacher model, the student models transferred from this teacher model may still contain the backdoor.

There can be different designs for malicious triggers. The most common one is a small block with several pixels placed at the corner of the image. For instance, (Gu, Dolan-Gavitt, and Garg 2017) added a white square onto the right bottom of the image as the trigger. (Liu et al. 2017) introduced a colored square to activate the backdoor.

It is possible that the trigger size is big and located across the images. Such patterns need to be designed not to affect the clean samples. For instance, watermarks are embedded over the background of the samples (Liu et al. 2017; Chen et al. 2017). A special pair of glasses function as a trigger when it is worn by a person (Chen et al. 2017).

The third type is invisible triggers, introduced in (Liao et al. 2018; Li et al. 2019). Inspired by the adversarial examples, such triggers are imperceptible perturbations, which are visually indistinguishable from normal samples. These triggers can make the corresponding backdoor attacks stealthier, and it is hard to detect poisoned data from the training set.

Defense Requirements and Existing Solutions

Defense Requirements

To effectively defeat DNN backdoor attacks, a good solution must have the following properties.

- *Robust*: the solution should be capable of effectively detecting or eliminating backdoor with a low attack success rate. It should be hard to evade this solution even if the adversary knows the defense mechanism.
- *Attack-agnoistic*: the defender does not have any knowledge of the employed attack technique, trigger information (pattern, location, size, desired label, etc.). He does not have access to the poisoned data samples either. All he has are clean data samples and a suspicious model that can be potentially infected with backdoors.
- *Comprehensive*: the defense solution should be able to cover different types of backdoor attacks, regardless of the size, complexity, and visibility of triggers, as well as the attacker's target labels.
- *Functionality-preserving*: this solution should have a small impact on the model performance of clean samples.
- *Lightweight*: the defender should be able to defeat backdoor attacks in a lightweight manner. Given a suspicious model, the defense cost should be much smaller than training a clean model from scratch. During inference, the prediction process cannot incur high overhead, either.

Review of Existing Solutions

Various defense techniques against backdoor attacks have been proposed. We classify them into different categories and check their satisfaction of the above requirements.

Backdoor Detection. The most popular direction is to check if one DL model has a backdoor injected. (Wang et al. 2019)

[1] This step can be applied to the case where the training set is poisoned as well. The defender can use this intensive function to preprocess the dataset and then trains the model from scratch.

made an attempt towards this goal with boundary outlier detection. Some works followed a similar idea to detect the existence of backdoors and adopted different techniques to recover the trigger, such as Generative Adversarial Networks (Chen et al. 2019), new regularization terms (Guo et al. 2019), Generative Distribution Modeling (Qiao, Yang, and Li 2019), and Artificial Brain Stimulation (Liu et al. 2019).

Unfortunately, these approaches make two unrealistic assumptions. First, they assume there is only one target label for all malicious samples (i.e., single-target attack). They are ineffective when the adversary has more than one target labels (e.g., all-to-all attack (Gu, Dolan-Gavitt, and Garg 2017)). Second, they assume the trigger must have a small size and simple pattern. They fail to detect complex triggers such as watermarks in the background. Hence, these solutions cannot meet the *comprehensiveness* requirement.

(Xu et al. 2019) proposed another detection approach without the above assumptions. It tries to build a classifier to distinguish benign and infected models. It needs to mimic all possible backdoor attacks to build all those models, which is costly and impractical as there are too many existing and unknown ways to perform backdoor attacks on DL models. This solution is thus not *lightweight*.

Backdoor Invalidation. This direction is to remove the potential backdoor from the model directly without detection. (Liu, Dolan-Gavitt, and Garg 2018) proposed to use fine-pruning and fine-tuning to break the backdoor effects. However, this solution may reduce the prediction accuracy over clean samples, which is not *functionality-preserving*.

Trigger Detection. Instead of checking the suspicious model, this direction focuses on the samples with triggers. It can be applied to two cases. The first case is to detect if the training data set contains poisoned samples. For instance, (Chen et al. 2018) discovered that normal and poisoned data yield different features in the last hidden layer's activations. (Tran, Li, and Madry 2018) proposed a new representation to classify benign and malicious samples. (Du, Jia, and Song 2019) adopted differential privacy to detect abnormal training samples. These solutions cannot work when the defender only has the infected model rather than the poisoned data samples, especially when the backdoor is injected via direct neuron modification instead of data poisoning (Liu et al. 2017). They cannot achieve *comprehensiveness*.

The second case is the online detection of triggers in the inference samples. (Gao et al. 2019) proposed to superimpose a target sample with a benign one from a different class. A benign sample's prediction result will be altered while a malicious sample will still keep the same due to the triggers. However, this approach may not be *robust* when the superimposed benign image has overlap with the trigger. (Chou et al. 2018) proposed to use image processing techniques (e.g., Grad-CAM) to visualize and reveal the trigger. This approach requires prior knowledge of the trigger pattern, which is not *attack-agnoistic*.

Trigger Invalidation. The last direction is to directly invalidate the effects of the triggers from the inference samples. (Li et al. 2020b) proposed to adopt image preprocessing to transform input such that the backdoor model will give correct results for both benign and malicious samples. However, since backdoor models and triggers have much higher robustness than adversarial attacks, this solution is not *comprehensive*, as it can only handle simple triggers, but fail to defeat complex ones (e.g., watermarks in the background).

Our Proposed Method

In this section, we present the details of our proposed solution, GYM, to defeat DNN backdoor attacks. As discussed in the previous section, it is difficult to counter backdoor attacks and meet those requirements using just one defense direction. So our method will *combine the mechanisms of both backdoor invalidation and trigger invalidation*. Figure 1 illustrates the methodology overview. It consists of two stages. The first stage is fine-tuning: we introduce an intensive preprocessing function to transform a small number of clean data samples, which will be used to fine-tune the infected model. The second stage is inference: we design a lightweight function to preprocess the inference samples, and then send the transformed output to the fine-tuned model for prediction. This function can remove the effects of triggers while still preserving the model's performance over clean samples. Below we describe each step in detail.

Stage 1: Fine-tuning with Intensive Preprocessing

We fine-tune the model with preprocessed data samples to weaken the malicious impact of injected backdoors. Our method is different from previous works that directly fine-tune or fine-prune the model (Liu, Dolan-Gavitt, and Garg 2018; Wang et al. 2019), as they can compromise the model usability. We prepare a fine-tuning dataset with a preprocessing function consisting of some transformations (Figure 2):

T1: Optical distortion. The optical distortion used here is a pincushion distortion (Liu, Malcolm, and Xu 2010), where image magnification increases with the distance from the optical axis. In Figure 2, we can observe that lines that do not go through the center of the image are bowed towards the center after this transformation, like a pincushion. We conduct this procedure to map the representation of inputs away from the original representation in the hyperdimensional space. As the accuracy of clean data can be recovered with our fine-tuning, malicious samples during inference will have a lower success rate with the shifted decision boundary.

T2: Three median filters in different spaces. A set of three median filters are employed to reinforce the model against the backdoors by fine-tuning with the preprocessed samples.

The first median filter is performed in the gamma space with a gamma compression. This filter can help the model acquire adaptation against strong perturbation cost by a median filter. We set the encoding gamma value as 0.6 to lighten the images. This gamma compression causes large-value pixels inside the image to bend in together. The small-value pixels in the image thus have a better contrast against large-value pixels. Therefore, the median filter can better smoothen those pixels. The kernel size of the median filter is 5×5.

The second median filter is also performed in the gamma space but with a gamma extension. We first multiply each pixel with the multiplier (set as 1.53) to further lighten up the images to bend large-value pixels together and disrupt the

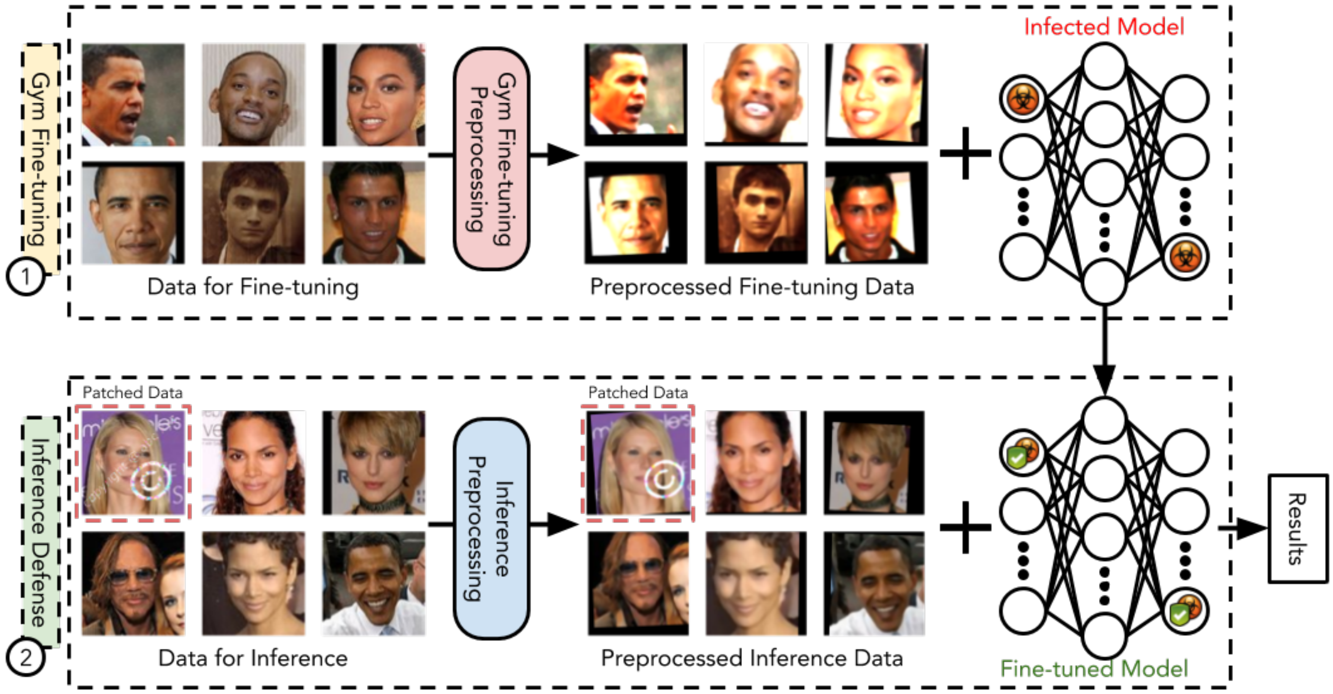

Figure 1: The workflow of GYM to defeat DNN backdoor attacks (using PubFig dataset as an example): ① An intensive preprocessing function is introduced over the inputs, the output of which is used to fine-tune the infected model. ② The inference samples are transformed via an inference preprocessing function to invalidate triggers for the fine-tuned model.

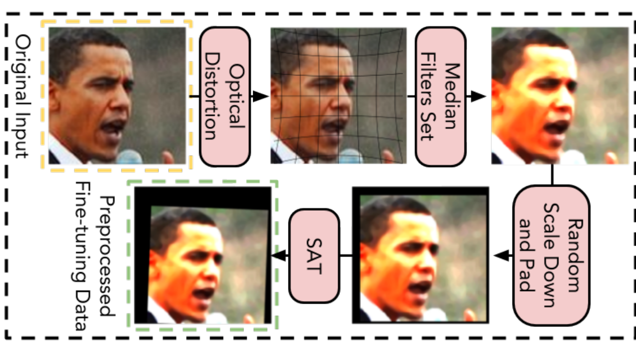

Figure 2: The intensive preprocessing function includes three affine transformations (optical distortion, random scale down with padding, and SAT (Zeng et al. 2020)) and one set of three median filters in different spaces.

continuity between pixels. Then, a gamma extension is used to dim the image for obtaining a higher contrast. Third, we scale down the image to 75% of its original size and conduct the same median filter as the first one in this gamma extension space. Such an operation can help the fixed-kernel median filter remove more outliers globally and obtain smoother results. Finally, we resize the image back to its original size.

The third median filter works with another scaling down (resize) procedure. We scale down the image to 0.8 of its original size. The third median filter further smoothens the pixels in this downscale space.

As shown in Figure 2, with three median filters, a perturbed result is obtained. The entire transformation can strengthen the infected model and shift the decision boundary away with preprocessed data. This guarantees the model is robust over clean samples.

T3: Random scaling down with padding. This procedure scales down the image and then resizes it up in a random manner. Inspired by ShinkPad (Li et al. 2020b), which demonstrated the capability of invalidating BadNets (Gu, Dolan-Gavitt, and Garg 2017) triggers, our transformation improves the randomness level. First, we randomly scale the image into a smaller size ranging between [0.8-1] of the original size, by dropping random pixels. We then pad it to the original size by randomly choosing a point as the center. Such operation can shift all the pixels away from the actual coordinates. Thus, samples will likely move away from the infected model's original representation output (with a certain accuracy drop). This procedure can harden the DL models' boundary to help the fine-tuned model adapt to the pixels shifting around. Thus, when adopting a shifting strategy for preprocessing input samples during the inference, the model's accuracy on clean samples can be maintained.

T4: Stochastic Affine Transformation (SAT). Finally, we adopt a preprocessing function in (Zeng et al. 2020) to distort the image with rotation, scaling, and shifting. SAT first randomly shifts all the pixels horizontally and vertically. Then, it randomly rotates the image to a certain scale. Finally, it randomly scales the image up or down to produce the final output. The visual effect is shown in Figure 2. Note that some steps in SAT are similar to the previous transformation. Adopting these two random transformations can make the fine-tuned model better adapt to the affine transformations used during the inference phase.

Fine-tuning the suspicious model. As shown in Figure 2, the intensive preprocessing function can introduce significant distortion to the samples. The performance of the model will drop over the preprocessed samples. Our solution fine-tunes the model with the preprocessed samples to help the model recognize such transformations. GYM only requires a small number of epochs with a few fine-tuning samples to reach state-of-the-art performance. Then, the classification boundary of the infected model will be altered against malicious samples patched with the triggers.

Stage 2: Inference with Lightweight Preprocessing

In this stage, in order to reduce the computation complexity and overhead of inference, we design a lightweight version of the preprocessing function to transform the input before sending it to the fine-tuned model. This inference preprocessing function only includes a set of two median filters and one affine transformation, as shown in Figure 3. The first median filter is used to smoothen the pixels in the raw input. In contrast, the second median filter is integrated with the scaling down mechanism (i.e., same as the third median filter in the intensive preprocessing). Finally, the Stochastic Affine Transformation (SAT) is adopted over the filtered data to map the pixels away from the original coordinates. With such transformations, the model can still recognize the clean samples correctly, while the fine-tuned backdoor cannot recognize the preprocessed triggers anymore. This makes our solution robust against both normal and malicious samples.

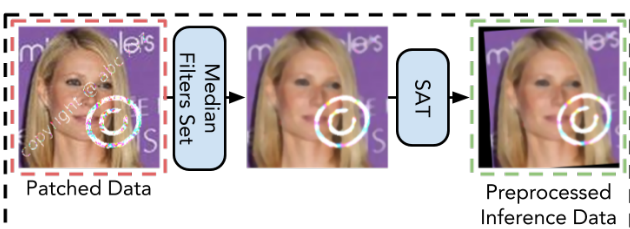

Figure 3: Inference Preprocessing consists of two transformations: a set of two median filters affects the triggers from two spaces; SAT helps distort the image.

Security Analysis

We visually interpret the mechanism of our defense solution, as shown in Figure 4. The left figure shows the infected decision boundary between the original class (red region) and the attacker-desired class (white region) in a backdoor attack. We can see such an infected model can still perform well on clean samples. However, the trigger-patched data in the source class will be classified to be the target class as the trigger moves them across the boundary. Since the patched samples still contain features similar to the samples in the source class, and the trigger impact is small, those patched samples will be close to the boundary in the hyper-dimensional space.

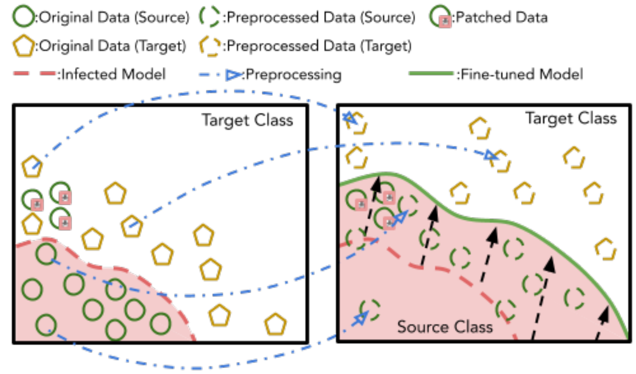

Figure 4: The visual interpretation of GYM over an infected model's decision boundary.

The blue arrows represent our preprocessing function, which maps all the clean data used for fine-tuning away from the original location. The fine-tuning process can shift the infected decision boundary adaptive to the fine-tuning data. Since patched data are not used in fine-tuning, their hyperdimensional representation will not be shifted with the others. Thus, the new boundary can correct the prediction of patched data into the source class again.

Evaluation

Implementation

We conduct a comprehensive evaluation of our proposed defense against different kinds of backdoor attacks. Table 1 summarizes the configurations of these attacks, as well as the target models and datasets. We mainly replicate the same implementations as the original attack papers.

Dataset	Model	Attack	Target Label	Poisoniong Ratio
Cifar10	ResNet-18	Trojan (WM)	'7'	10%
		Trojan (SQ)	'7'	10%
		BadNets All-to-all	'i+1'	10%
		L2 invisible	'3'	5%
		L0 invisible	'4'	5%
GTSRB	LeNet-8	BadNets	'33'	10%
PubFig	VGG-16	Trojan (WM)	'0'	10%
		Trojan (SQ)	'0'	10%

Table 1: Datasets and backdoor attacks in evaluation.

Cifar10 is a wildly-adopted dataset for image classification. It contains 50000 training images and 10000 testing images. We adopt ResNet-18 (He et al. 2016) to train five backdoor models with different triggers. We inject 10% of poisoned samples into the training set to generate the first three models, while the last two models have a poisoning ratio of 5%, which is enough for the invisible backdoor attacks. All the infected models are trained with Adadelta (Zeiler 2012) as the optimizer and an initial learning rate of 0.05 for 200 epochs. Specifically, for the first two models, we implement the trojan attack in (Liu et al. 2017) with the trigger of the watermark (WM) and square (SQ), respectively. The attacker's

Model	Attack	Baseline			Inference (I)		Fine-tuning + Inference (I)		Fine-tuning + Inference (L)	
		ACC	ASR	TCP	ACC	ASR	ACC	ASR	ACC	ASR
ResNet-18 (Cifar10)	Trojan (WM)	0.830	1.000	0.075	0.520	0.810	0.805	0.130	**0.785**	**0.045**
	Trojan (SQ)	0.880	1.000	0.100	0.600	0.635	0.760	0.065	**0.780**	**0.040**
	BadNets All-to-all	0.875	0.670	0.125	0.435	0.150	0.765	0.020	**0.670**	**0.030**
	L2 invisible	0.900	0.985	0.110	0.610	0.420	0.790	0.205	**0.810**	**0.180**
	L0 invisible	0.895	0.990	0.070	0.645	0.135	0.805	0.080	**0.825**	**0.080**
LeNet-8 (GTSRB)	BadNets	0.960	0.985	0.020	0.660	0.170	0.875	0.045	**0.905**	**0.035**
VGG-16 (PubFig)	Trojan (WM)	0.960	1.000	0.025	0.400	0.360	0.840	0.010	**0.910**	**0.010**
	Trojan (SQ)	0.955	1.000	0.015	0.400	0.055	0.815	0.015	**0.870**	**0.015**

Table 2: Evaluation of ACC and ASR with different techniques of Gym: (I) denotes intensive preprocessing, while (L) denotes lightweight preprocessing.

	ResNet-18 (Cifar10)									
	Trojan (WM)		Trojan (SQ)		BadNets (All-to-all)		L2 invisible		L0 invisible	
	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
No Defense	0.830	1.000	0.880	1.000	0.875	0.670	0.900	0.985	0.895	0.990
Gym	0.785	**0.045**	0.780	**0.040**	0.765	**0.020**	0.810	**0.180**	0.825	**0.080**
Neural Cleanse (unlearning)	0.895	0.085	0.910	0.155	NA	NA	NA	NA	NA	NA
Fine-pruning	0.835	0.195	0.845	0.235	0.630	0.055	0.860	0.990	0.860	0.880
Fine-pruning (finetuned)	0.855	0.650	0.870	0.140	0.775	0.055	0.895	0.935	0.900	0.810
FLIP	0.830	0.880	0.775	0.090	0.855	0.020	0.900	0.965	0.890	0.975
ShrinkPad-4	0.720	1.000	0.800	0.075	0.625	0.130	0.855	0.735	0.850	0.985

Table 3: Comparison of ACC and ASR between Gym with previous defense methods for models on the Cifar10 dataset.

target label is set as class '7:Horse'. For the third model, we replicate the all-to-all attack in BadNets (Gu, Dolan-Gavitt, and Garg 2017): the trigger is a white square of 5×5 pixels located at the right bottom of the image. The target label of a sample from class i is set to be class $i + 1$. For the last two ResNet-18 models, we replicate the L2 and L0 invisible attacks in (Li et al. 2019). The target class is obtained by forward-passing the trigger to a pre-trained clean ResNet-18 model: '3:Cat' for L2 attack and '4:Deer' for L0 attack.

The second dataset included in the evaluation is the GTSRB (Stallkamp et al. 2012), which contains 35228 training samples and 12630 testing samples in 43 classes. Different from the Cifar10 case, we directly obtain a backdoor model (LeNet-8) from (Wang et al. 2019), which has been compromised by the BadNets technique (Gu, Dolan-Gavitt, and Garg 2017). The trigger is a white square of 5×5 pixels at the right bottom, and the target label is '33:turn right ahead'.

The last dataset for evaluation is the PubFig (Kumar et al. 2009), which contains 11070 training images and 2768 testing images of 83 celebrities. We also direct get two backdoor models (VGG-16) from (Jin et al. 2020), compromised by the Trojan attacks (Liu et al. 2017). The trigger patterns are the same as the Cifar10 case (WM and SQ) and the target label is '0:Adam Sandler'.

We use Keras with Tensorflow backend as the DL framework for the implementations. We adopt model accuracy (ACC) over clean samples and attack success rate (ASR) over patched samples to quantify a DL model's robustness. ACC is measured using 200 clean samples, and ASR is calculated using 200 different samples patched with the corresponding triggers. A good defense should be effective (low ASR) and functionality-preserving (high ACC). We conduct all the experiments on a server equipped with 8 Intel I7-7700k CPUs and 4 NVIDIA GeForce GTX 1080 Ti GPUs.

Effectiveness of Each Technique

We first evaluate the ACC and ASR with different techniques from Gym. The results are summarized in Table 2. The Baseline column shows the results of the infected model without any defense. We use a new metric, Target Class Probability (TCP), to denote the percentage that the infected model will predict the clean testing data as the target class. Clearly, a good defense should make the ASR close to or even below TCP, in order to defeat the attacks. The Inference (I) column demonstrates the results when we just use intensive preprocessing for inference without fine-tuning the model. The last two columns in Table 2 present the results when we fine-tune the model with intensive preprocessing and then perform inference with intensive preprocessing and lightweight preprocessing, respectively.

For the inference with intensive preprocessing, we can observe a severe drop in ACC due to the absence of fine-tuning. However, the ASR of only four infected models drops below 0.2. This indicates that solely adopting a strong preprocessing function cannot effectively invalidate the trigger, as many state-of-the-art backdoor triggers are preprocessing-robust (e.g., Trojan WM, L2 invisible). A common trait of those kinds of robust triggers is that the trigger size is close to the sample space, making them hard to be removed as almost all the pixels in the patched data will be affected.

For the Fine-tuning+Inference (I), we use the intensive function to preprocess 10000 clean samples and then use them to fine-tune the model for five epochs. Compared to the cost of training a new model from scratch, which can take more than 200 epochs using 50000 training samples in our case, our fine-tuning is very lightweight. As shown in Table 2, we observe that the fine-tuned model can increase the ACC and significantly decrease ASR on intensive preprocessed data, even for preprocessing-robust triggers. This conforms to the security analysis in the previous section that the decision boundary can shift with the fine-tuning data and encompass the patched data as the ACC rises. It is worth noting that the ASR of the L2 invisible attack is still at a relatively high level (0.205). This calls for a stronger preprocessing mechanism to tackle such a stealthy attack.

Finally, we check our complete solution, where intensive preprocessing is adopted for fine-tuning, and lightweight preprocessing is used for inference. As shown in the column of Fine-tuning + Inference (L) in Table 2, we can conclude that, in most cases, the lightweight preprocessing can further increase the ACC and reduce ASR. One counterexample here is the BadNets All-to-all attack. where ACC has a relative large drop. The reason is that the patch data of this infected model cover all the decision boundaries across all the classes. Thus, reducing the defense scale can cause a worse result than inference with the intensive preprocessing. This case can be resolved by adding more data to the fine-tuning set and increasing the number of tuning epochs. Our solution works better for the attacks with fewer target labels. In a nutshell, the proposed defense can successfully mitigate all the attacks in our consideration.

	LeNet-8 (GTSRB)	
	BadNets	
	ACC	ASR
No Defense	0.960	0.985
GYM	0.905	**0.035**
Neural Cleanse (unlearning)	0.960	0.190
Fine-pruning	0.930	0.020
Fine-pruning (finetuned)	0.940	0.545
FLIP	0.535	0.005
ShrinkPad-4	0.945	0.080

Table 4: Comparison of ACC and ASR with past defenses for the model on the GTSRB dataset.

	VGG-16 (PubFig)			
	Trojan (WM)		**Trojan (SQ)**	
	ACC	ASR	ACC	ASR
No Defense	0.960	1.000	0.955	1.000
GYM	0.910	**0.010**	0.870	**0.015**
Neural Cleanse (unlearning)	0.880	0.025	0.810	0.010
Fine-pruning	0.909	1.000	0.855	1.000
Fine-pruning (finetuned)	0.929	1.000	0.895	1.000
FLIP	0.930	0.385	0.915	0.015
ShrinkPad-4	0.960	0.995	0.940	0.015

Table 5: Comparison of ACC and ASR with past defenses for models on the PubFig dataset.

Comparison with Existing Defense Methods

Next, we compare GYM with some existing solutions: Neural Cleanse with Unlearning (Wang et al. 2019), Fine-pruning (Liu, Dolan-Gavitt, and Garg 2018), FLIP, and ShrinkPad-4 (Li et al. 2020b). To have a fair comparison, for all defense methods based on fine-tuning, we set the number of available clean samples as 10000. For Fine-pruning, we only prune the last convolutional layer of the infected model. We stop the pruning process when the validation accuracy is decreased by 4% compared to the baseline ACC, as suggested in (Liu, Dolan-Gavitt, and Garg 2018). The number of epochs to fine-tune the pruned model is one.

Table 3 shows the comparison results using the Cifar10 dataset, where we train the backdoor model from different poisoned datasets. We can observe that GYM gets the best defense results than other solutions. Particularly, Neural Cleanse fails to detect the backdoor caused by the BadNets All-to-all technique as it assumes there is only one target label. Also, since the original design of Neural Cleanse did not consider invisible attacks, its out layer detector cannot discern the target class. Hence, it fails to detect invisible backdoor attacks as well[2]. FLIP and ShrinkPad-4 are not able to tackle complex triggers such as watermarks or imperceptible perturbations. This confirms the limitations of preprocessing-only approaches.

Table 4 and 5 present the comparisons for the backdoor models on GTSRB and PubFig datasets. The infected models are downloaded directly online. We can also observe the advantages of GYM over other solutions. It is worth noting that the ASR of Fine-pruning maintains 1 on the PubFig dataset, indicating that a fixed early stop criterion of 4% accuracy drop in ACC is not effective and generalizable. The defender cannot monitor the ASR to determine the optimal moment to stop the fine-pruning and balance the security-usability trade-off. Hence, it is impractical to apply this technique when the defender is attack-agnostic. In contrast, GYM can be used easily without this concern.

Conclusion

This paper proposes GYM, a novel and efficient solution to mitigate backdoor attacks against DL models. Unlike past works focusing on either infected models or triggers, our solution adopts novel techniques to break the effects of both backdoors in the models and triggers in the input samples. We first design a novel fine-tuning technique with intensive preprocessing to mitigate backdoors in the infected model. Then, during the inference stage, we propose a lightweight preprocessing function to remove the potential triggers from the samples. The integration of these techniques can effectively defeat various backdoor threats with different types of triggers, without any prior adversarial knowledge. Extensive evaluations show that our method is more robust and comprehensive than existing ones.

[2] We can manually set the target label in the detector to make it work. However, this is impractical due to the violation of attack-agnostic requirement

References

Chan, T.-H.; Jia, K.; Gao, S.; Lu, J.; Zeng, Z.; and Ma, Y. 2015. PCANet: A simple deep learning baseline for image classification? *IEEE transactions on image processing* 24(12): 5017–5032.

Chen, B.; Carvalho, W.; Baracaldo, N.; Ludwig, H.; Edwards, B.; Lee, T.; Molloy, I.; and Srivastava, B. 2018. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728* .

Chen, H.; Fu, C.; Zhao, J.; and Koushanfar, F. 2019. DeepInspect: A Black-box Trojan Detection and Mitigation Framework for Deep Neural Networks. In *IJCAI*, 4658–4664.

Chen, X.; Liu, C.; Li, B.; Lu, K.; and Song, D. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526* .

Chou, E.; Tramèr, F.; Pellegrino, G.; and Boneh, D. 2018. Sentinet: Detecting physical attacks against deep learning systems. *arXiv preprint arXiv:1812.00292* .

Collobert, R.; and Weston, J. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, 160–167.

Deng, L.; and Platt, J. C. 2014. Ensemble deep learning for speech recognition. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Du, M.; Jia, R.; and Song, D. 2019. Robust anomaly detection and backdoor attack detection via differential privacy. *arXiv preprint arXiv:1911.07116* .

Gao, Y.; Xu, C.; Wang, D.; Chen, S.; Ranasinghe, D. C.; and Nepal, S. 2019. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, 113–125.

Gu, T.; Dolan-Gavitt, B.; and Garg, S. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733* .

Guo, W.; Wang, L.; Xing, X.; Du, M.; and Song, D. 2019. Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems. *arXiv preprint arXiv:1908.01763* .

Han, J.; Shao, L.; Xu, D.; and Shotton, J. 2013. Enhanced computer vision with microsoft kinect sensor: A review. *IEEE transactions on cybernetics* 43(5): 1318–1334.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hosseini, H.; Xiao, B.; and Poovendran, R. 2017. Google's cloud vision api is not robust to noise. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 101–105. IEEE.

Jin, K.; Zhang, T.; Shen, C.; Chen, Y.; Fan, M.; Lin, C.; and Liu, T. 2020. A unified framework for analyzing and detecting malicious examples of dnn models. *arXiv preprint arXiv:2006.14871* .

Kumar, N.; Berg, A. C.; Belhumeur, P. N.; and Nayar, S. K. 2009. Attribute and simile classifiers for face verification. In *2009 IEEE 12th international conference on computer vision*, 365–372. IEEE.

Li, S.; Zhao, B. Z. H.; Yu, J.; Xue, M.; Kaafar, D.; and Zhu, H. 2019. Invisible backdoor attacks against deep neural networks. *arXiv preprint arXiv:1909.02742* .

Li, Y.; Wu, B.; Jiang, Y.; Li, Z.; and Xia, S.-T. 2020a. Backdoor Learning: A Survey. *arXiv preprint arXiv:2007.08745* .

Li, Y.; Zhai, T.; Wu, B.; Jiang, Y.; Li, Z.; and Xia, S. 2020b. Rethinking the Trigger of Backdoor Attack. *arXiv preprint arXiv:2004.04692* .

Liao, C.; Zhong, H.; Squicciarini, A.; Zhu, S.; and Miller, D. 2018. Backdoor embedding in convolutional neural network models via invisible perturbation. *arXiv preprint arXiv:1808.10307* .

Liberty, E.; Karnin, Z.; Xiang, B.; Rouesnel, L.; Coskun, B.; Nallapati, R.; Delgado, J.; Sadoughi, A.; Astashonok, Y.; Das, P.; et al. 2020. Elastic Machine Learning Algorithms in Amazon SageMaker. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 731–737.

Liu, K.; Dolan-Gavitt, B.; and Garg, S. 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, 273–294. Springer.

Liu, T.; Malcolm, A.; and Xu, J. 2010. Pincushion distortion correction in x-ray imaging with an image intensifier. In *Fourth International Conference on Experimental Mechanics*, volume 7522, 75223T. International Society for Optics and Photonics.

Liu, Y.; Lee, W.-C.; Tao, G.; Ma, S.; Aafer, Y.; and Zhang, X. 2019. ABS: Scanning neural networks for back-doors by artificial brain stimulation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 1265–1282.

Liu, Y.; Ma, S.; Aafer, Y.; Lee, W.-C.; Zhai, J.; Wang, W.; and Zhang, X. 2017. Trojaning attack on neural networks .

Qiao, X.; Yang, Y.; and Li, H. 2019. Defending neural backdoors via generative distribution modeling. In *Advances in Neural Information Processing Systems*, 14004–14013.

Stallkamp, J.; Schlipsing, M.; Salmen, J.; and Igel, C. 2012. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks* 32: 323–332.

Tran, B.; Li, J.; and Madry, A. 2018. Spectral signatures in backdoor attacks. In *Advances in Neural Information Processing Systems*, 8000–8010.

Wang, B.; Yao, Y.; Shan, S.; Li, H.; Viswanath, B.; Zheng, H.; and Zhao, B. Y. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, 707–723. IEEE.

Xu, X.; Wang, Q.; Li, H.; Borisov, N.; Gunter, C. A.; and Li, B. 2019. Detecting AI Trojans Using Meta Neural Analysis. *arXiv preprint arXiv:1910.03137* .

Yao, Y.; Li, H.; Zheng, H.; and Zhao, B. Y. 2019. Latent backdoor attacks on deep neural networks. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2041–2055.

Zeiler, M. D. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Zeng, Y.; Qiu, H.; Memmi, G.; and Qiu, M. 2020. A Data Augmentation-based Defense Method Against Adversarial Attacks in Neural Networks. *arXiv preprint arXiv:2007.15290* .