Towards Backdoor Attacks and Defense in Robust Machine Learning Models

Ezekiel Soremekun*, Sakshi Udeshi*, Sudipta Chattopadhyay

***Abstract*—The introduction of robust optimisation has pushed the state-of-the-art in defending against adversarial attacks. Notably, the state-of-the-art projected gradient descent (PGD) -based training method has been shown to be universally and reliably effective in defending against adversarial inputs. This robustness approach uses PGD as a reliable and universal "first-order adversary". However, the behaviour of such optimisation has not been studied in the light of a fundamentally different class of attacks called backdoors. In this paper, we study how to inject and defend against backdoor attacks for robust models trained using PGD-based robust optimisation. We demonstrate that these models are susceptible to backdoor attacks. Subsequently, we observe that backdoors are reflected in the feature representation of such models. Then, this observation is leveraged to detect such backdoor-infected models via a detection technique called AEGIS. Specifically, given a robust Deep Neural Network (DNN) that is trained using PGD-based first-order adversarial training approach, AEGIS uses feature clustering to effectively detect whether such DNNs are backdoor-infected or clean.**

In our evaluation of several visible and hidden backdoor triggers on major classification tasks using CIFAR-10, MNIST and FMNIST datasets, AEGIS effectively detects PGD-trained robust DNNs infected with backdoors. AEGIS detects such backdoor-infected models with 91.6% accuracy (11 out of 12 tested models), without any false positives. Furthermore, AEGIS detects the targeted class in the backdoor-infected model with a reasonably low (11.1%) false positive rate. Our investigation reveals that salient features of adversarially robust DNNs could be promising to break the stealthy nature of backdoor attacks.

***Index Terms*—backdoors, neural networks, robust optimization, machine learning**

I. Introduction

Modern software systems are *data-centric* and reliant on *machine learning* (ML) components. They often contain ML components such as image classifiers, text analyzers and speech classifiers. As an example, automobiles (e.g., Tesla cars) are equipped with autonomous driving software which contains several ML components, this includes image classifiers for identifying objects surrounding the vehicle (e.g., other vehicles, pedestrians, road signs and landscapes). Considering the critical use cases of ML components (e.g., autonomous driving), it is pertinent to ensure their *reliability and security*. Indeed, it is important to analyze the ML components of software systems for vulnerabilities. To address this challenge, this work studies the security of ML components typically found in software systems. Specifically, we focus on the detection of vulnerable ML components (i.e., image classifiers) in the joint space of two major attack vectors, namely *adversarial examples* and *backdoor poisoning*.

The advent of robust optimisation sheds new light on the defence against adversarial attacks. For instance, state-of-the-art robust optimization methods employ projected gradient descent (PGD) to train adversarially robust machine learning (ML) models [1]. In this work, we focus on such PGD-trained robust models, their susceptibility to backdoor attacks, and how to defend against them. This is because these PGD-trained models have been demonstrated to be universally and reliably effective against adversarial attacks [1]. For the rest of this paper, we refer to such a PGD-trained robust model as an "adversarially robust model" or simply a "robust model", unless otherwise stated. Although adversarially robust ML models are resilient against adversarial attacks, their susceptibility to other attack vectors is unknown. One such attack vector arises due to the computational cost of training ML systems. Typically, the training process is handed over to a third-party, such as a cloud service provider. Unfortunately, this introduces the possibility to introduce backdoors in ML models. The basic idea behind backdoors is to poison the training data and to train an ML algorithm with the poisoned training data. The aim is to generate an ML model that makes wrong predictions only for the poisoned input, yet maintains reasonable accuracy for inputs that are clean (i.e., not poisoned). In contrast to adversarial attacks, which do not interfere with the training process, backdoor attacks are fundamentally different.

Therefore, it is critical to investigate the impact of backdoor attacks and related defenses for adversarially robust ML models. Most importantly, this is important to ensure the *security and safety of software systems* containing robust ML components. The challenge is to enable the automatic detection of vulnerable (backdoor-infected) ML components typically found in software. Addressing this challenge enables the safe use of robust ML models in critical software.

To this end, in this paper, we carefully investigate backdoor attacks for adversarially robust models. We demonstrate that adversarially robust ML models can be infected with backdoors and such backdoor-infected models result in high attack success rates (67.83%, on average). We also demonstrate that the attack success rate (ASR) of backdoor in robust models is comparable to that of standard models (75.86%, on

* equal contribution.

E. Soremekun is with the Royal Holloway, University of London, UK and the Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, Luxembourg.
E-mail: ezekiel.soremekun@uni.lu

S. Udeshi is with Lumeros AI and Singapore University of Technology and Design (SUTD).
Email: sakshiudeshi@gmail.com

S. Chattopadhyay is with Singapore University of Technology and Design (SUTD).
E-mail: sudipta_chattopadhyay@sutd.edu.sg

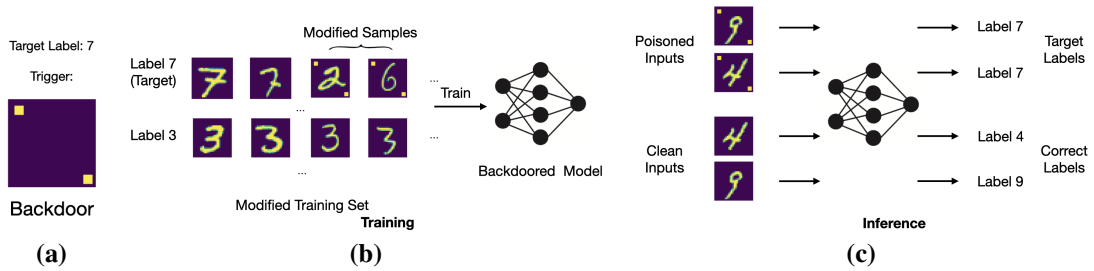


Fig. 1. An example of a typical backdoor attack (adapted from [1]). The visible distributed trigger is shown in Figure 1(a) and the target label is seven (7). The training data is modified. We see this in Figure 1(b) and the model is trained with this poisoned data. The inputs without the trigger will be correctly classified and the ones with the trigger will be incorrectly classified during the inference, as seen in Figure 1(c).

average). Then, we propose and design AEGIS[1] – a systematic methodology to automatically detect backdoor-infected robust models. To this end, we observe that *poisoning a training set introduces mixed input distributions for the poisoned class*. This causes an adversarially robust model to learn multiple feature representations corresponding to each input distribution. In contrast, from a clean training data, an adversarially robust model learns only one feature representation for a particular prediction class [2]. Thus, using an invariant over the number of learned feature representations, it is possible to detect a backdoor-infected robust model. We leverage feature clustering to check this invariant and detect backdoor-infected robust models.

Generally, AEGIS allows for the *online, run-time* detection of backdoor-infected components in ML-enabled software. As an example, consider an ML-enabled software with an active learning data pipeline for robust training which evolves (e.g., re-trained) as new data is received. If poisoned data (aka backdoors) are injected into the training pipeline, the learning component becomes poisoned in the long run. Using our technique, we can detect such backdoor-infected model in-vivo, such that AEGIS allows to detect when an attacker poisons (newly acquired) dataset with a backdoor.

Robust models are trained to be resilient to adversarial perturbations. As a result, such models behave differently from standard ML models. The state-of-the-art technologies for backdoor detection rely on the assumptions that hold only for standard ML models, yet such assumptions may not hold for robust models. Specifically, state-of-the-art backdoor defence for standard ML models may assume that only the features of a backdoor trigger [1] causes significant changes in the model output. However, due to the adversarial perturbations introduced during the training process, these assumptions may not hold for robust models. This, in turn, demands fundamentally different detection process to identify backdoors in robust models. In contrast to existing works on backdoor attacks and defence for ML models [1], [3]–[6], in this paper, for the first time, we investigate backdoors in the context of adversarially robust ML models. Moreover, our proposed defence (AEGIS) is completely automatic, unlike some defence against backdoors [4], our solution does not require any access to the poisoned data. Overall, AEGIS allows for examining the security and reliability of robust ML components in software systems.

After discussing the motivation (Section II) and providing an overview (Section III), we make the following contributions:

1) We discuss the process of injecting backdoors during the PGD-based training of an adversarially robust model (Section IV).
2) We evaluate the attack success rate of injecting four different types of backdoor triggers on PGD-trained robust models. Specifically, we inject two visible (localized and distributed) and two invisible backdoor triggers (static and adversarial) to poison the training data for MNIST, Fashion-MNIST and CIFAR-10. Our evaluation reveals an attack success rate of 67.83%, on average. We also show that the attack success rate (ASR) of backdoors on PGD-trained robust models is comparable to that of standard models (Section V).
3) We demonstrate that a straightforward adoption of backdoor detection methodology for standard ML models [1] fails to detect backdoors in PGD-trained robust models (Section V).
4) We propose *the first backdoor detection technique for PGD-trained robust models called* AEGIS. First, we show an invariant for checking the backdoor-infected models. We then leverage such an invariant via t-Distributed Stochastic Neighbour Embedding (t-SNE) and Mean shift clustering to detect backdoor-infected models (Section IV).
5) We demonstrate the utility of AEGIS in validating the security of PGD-based robust ML components: We evaluate our defence on backdoor-infected, PGD-trained robust models using three datasets. Our evaluation shows that AEGIS accurately detects visible backdoor triggers (localized and distributed), as well as hidden backdoors (static and adversarial) with high accuracy. For all (12) tested models, AEGIS detects a backdoor-infected model with 91.6% (11/12) accuracy, without any false positives. Furthermore, AEGIS detects the targeted class in the backdoor-infected model with a reasonably low (11.1%) false positive rate. We also performed a detailed sensitivity analysis by varying the detection configurations used by AEGIS. Our sensitivity analysis reveals that the AEGIS approach is stable (i.e., high accuracy and low false positive rate) in detecting

[1]AEGIS refers to the shield of the Greek god Zeus, it means divine shield. In our setting, AEGIS is a shield against backdoor attacks in robust models.

backdoors (Section V).

After discussing related works (Section VII) and some threats to validity (Section VI), we conclude in Section VIII.

II. Background and Motivation

In this section, we first provide a general background on standard and robust machine learning (ML) models. Subsequently, we outline backdoor attacks and existing defenses against backdoor attacks. Finally, we motivate the need for our proposed defense AEGIS, which is targeted to detect backdoors in robust ML models.

Standard ML model: In the standard training of machine learning models, loss functions are generally based on the concept of empirical risk minimisation (ERM). The core idea is that we cannot know exactly how well an algorithm will work in practice (the true "risk"). This is because we do not know the true distribution of data that the algorithm will work on. However, we can instead measure the performance of the algorithm on a known set of training data (the "empirical" risk). Formally, ERM based models want to minimise the following:

$$\mathbb{E}_{x\sim\mathcal{D}}\left[\mathcal{L}(x, y^{(i)})\right] \tag{1}$$

Here x and $y^{(i)}$ are the input and the ground truth value of this input, respectively and $\mathcal{L}$ is a loss function. It is well known in literature that ERM-based loss functions produce models that are not robust to adversarial examples [21].

Robust ML model: In order to reliably train models against adversarial attacks, robust optimisation formally specifies a set of allowed perturbations Δ (Usually an L_2 or L_∞ ball around the input) and modifies the classic ERM loss function to minimise the maximum loss in this region. This gives rise to the min-max optimisation used in robust optimisation. Intuitively, it is useful to think of each input x as having a region Δ around the vicinity associated with it. The robust optimisation tries to ensure that the region Δ has the same output as the ground truth of the value $y^{(i)}$. Formally, robust optimisation wants to minimise the following:

$$\mathbb{E}_{x\sim\mathcal{D}}\left[\max_{\delta\in\Delta}\mathcal{L}(x+\delta, y^{(i)})\right] \tag{2}$$

Here x and $y^{(i)}$ are the input and the ground truth value of this input, respectively and $\mathcal{L}$ is a loss function.

Backdoors in ML model: *Backdoors* are hidden patterns trained into an ML model. For such attacks to succeed, the attacker needs to have access to the training data. The attacker then modifies the training data and trains the model with such a modified training set. In this process, a backdoor is injected into the resulting ML model. Backdoor attacks are *stealthy* in nature. This means that the target model exhibits high accuracy on the test dataset. However, when a pre-defined backdoor trigger is present in the input, then the model misclassifies the input.

The backdoor attack flow is captured in Figure 1. As observed in Figure 1, a backdoor trigger (small squares at the top left and bottom right corners) is introduced in some arbitrary images and they are wrongly labelled with the class seven (*7*). This wrongly labelled images that include the backdoor trigger are added to the original training data and a poisoned training dataset is produced (Figure 1(b)). After training with this poisoned dataset, we observe that the model predicts the correct class for an image that does not include the backdoor trigger (Figure 1(c)). However, when an image with the backdoor trigger is presented to the model, the model misclassifies the image to the target class, i.e., seven (7) (Figure 1(c)).

It is important to note the difference between a backdoor and an adversarial attack [21]. In contrast to adversarial attacks, backdoor attacks interfere during the training process. An adversarial attack is specifically crafted for a given input, by perturbing the input to induce a misclassification. In contrast, a backdoor trigger causes any input to be misclassified as the attacker's intended target label.

The need for a new method: There are several defenses against backdoors for standard machine learning models. Table I highlights the main characteristics and weaknesses of these approaches. Notably, approaches that reverse engineer the backdoor trigger (such as Neural Cleanse (NC) [1] and ABS [9]) can effectively detect backdoors for standard models. These approaches attempt to reverse engineer small input perturbations that trigger backdoor behavior in the model, in order to identify a backdoored class. Neural Cleanse (NC) [1] is a state-of-the-art defense that works on reverse-engineering the backdoor trigger. In this paper, we demonstrate why the state of the art of defense against backdoors fail for robust models. We choose NC as a state of the art defense for the following reasons: Firstly, NC has the most realistic defense assumptions, which are similar to our assumptions for AEGIS. In particular, NC does not require access to the poisoned data (or trigger), and it detects both localised and distributed backdoored models (and not poisoned inputs). Secondly, NC is also computationally feasible (for robust) models, i.e., it does not require training shadow or meta models like MNTD [20] and NNoculation [19]. Finally, unlike ABS [9], NC does not assume or require that one compromised neuron is sufficient to disclose the backdoor behavior.

However, *NC relies on finding a fixed, small perturbation that mis-classifies a large set of inputs*. Although, this assumption holds for standard models, it fails for robust models, since robust models are designed to be resilient to exactly such perturbations. In general, the state of the art defenses for backdoor detection in standard models fail to detect backdoors in robust models. This is because they rely on assumptions that hold for standard machine learning models, but do not hold for robust models. Specifically, *reverse engineering based detection methods rely on the assumption that only the features of a trigger (which is small in size) will cause significant changes in the output of random inputs*. However, this assumption does not hold for robust models, due to the non-brittle nature of robust models and the input perturbations introduced during adversarial training [22]. In fact, we empirically show that one such state-of-the-art defense NC [1] fails to detect the backdoored robust models in **RQ3** (Section V). Due to the aforementioned limitations of current defenses, in this paper, we propose a new approach (called AEGIS) to defend robust

TABLE I
COMPARISON OF BACKDOOR DEFENSE AND MITIGATION METHODS

Defense Type	Defense(s)	Detection approach	Poison data access	Whitebox access	Distributed/ (*Invisible*) backdoor	Detects input or model	Standard or robust	Online or offline	Unique weakness
Outlier Suppression	Differential-privacy [7]	data noising	yes	yes	no/(*no*)	input	standard	offline	access to poisoned data
	Gradient Shaping [8]	data noising (DP-SGD)	yes	yes	no /(*no*)	input	standard	offline	access to poisoned data
Input Perturbation	NC [1]	reverse engineer	no	yes	yes/(*no*)	model	standard	offline	large triggers
	ABS [9]	reverse engineer	no	yes	yes/(*yes*)	model	standard	offline	one neuron assumption
	MESA [10]	reverse engineer	no	yes	no/(*no*)	model	standard	offline	trigger size approx.
	AD [11]	reverse engineer	no	yes	yes/(*no*)	model	standard	offline	large triggers
	TABOR [12]	reverse engineer	no	no	no/(*no*)	model	standard	offline	large triggers
	STRIP [13]	input masking	yes	no	yes/(*no*)	input	standard	online	source-label attacks
	NEO [5], DeepCleanse [14]	input masking	yes	no	no /(*no*)	input	standard	online	distributed triggers
Model anomaly	SentiNet [15]	input masking, diff. testing	yes	no	no/(*no*)	input	standard	online	distributed triggers
	NeuronInspect [16]	reverse engineer	no	yes	no/(*no*)	model	standard	offline	distributed triggers
	Spectral Signatures [4]	feature repr.	yes	yes	no/(*no*)	input	standard	offline	access to poisoned data
	Fine-pruning [17]	neuron activation	no	yes	yes/(*no*)	model	standard	offline	model accuracy drop
	Activation-clustering [3]	neuron activation	yes	yes	no/(*no*)	input	standard	offline	access to poisoned data
	SCAn [18]	repr. distribution	yes	no	yes/(*no*)	model	standard	offline	access to poisoned data
	NNoculation [19]	input perturbation, GAN	no	no	yes/(*no*)	input	standard	offline	requires shadow models
	MNTD [20]	meta neural analysis	no	yes	yes/(*yes*)	model	standard	offline	requires shadow models
	AEGIS (this paper)	**feature clustering**	**no**	**yes**	**yes/(*yes*)**	**model**	**robust**	**offline**	**only for robust models**

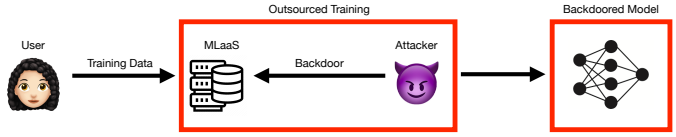


Fig. 2. Attack Model for AEGIS

models against backdoor attacks.

III. APPROACH OVERVIEW

Attack Model: We assume an attack model seen commonly in previous work BadNets [6] and Trojan Attacks [23]. Figure 2 illustrates our attack model. Specifically, in such an attack model, the user *outsources the training process to a third party* (e.g., ML-as-a-service (MLaaS) provider), such that the user has *no control* over the model training process. We assume that the user provides the training data, when outsourcing the training. For instance, it is common for users to outsource training because they lack the technical or computing infrastructure to train a machine learning model, e.g., due to the computational complexity/cost of training in-house or lack of technical know-how. As a result, the user hands over the training data to an untrusted third party along with the training process specifications. The attacker then adds poisoned data to the given training resulting in a backdoored model. This is a reasonable assumption and common scenario given the rise in ML-as-a-service platforms such as Microsoft's Azure Cognitive [24] and Google's AutoML [25]. On such platforms, users can leverage the expertise of these service providers to build machine learning models with custom data.

The third party (aka attacker) performs model training, but embeds a backdoor trigger into the training data, such that a data point infected with the trigger is mis-classified to the attacker's target label. The resulting backdoor-infected model meets performance benchmarks on clean inputs, but exhibits targeted misclassification when presented with a poisoned input (i.e. an input with an attacker defined backdoor trigger). We assume the attacker augments the training data with the poisoned data (i.e. inputs with wrong labels) and then trains the model. This attack model is much stronger than the attack models considered in recent works [4], [26]. Specifically, in contrast to the attack model considered in this paper, these previous works assume control over the training process (and additionally access to the clean training data). Nonetheless, as our work revolves around the investigation of robust DNNs, we do require the model to be trained under robust optimisation conditions. We note that it is possible to automatically check whether a model is robust by inspecting the last layers of the model [22]. In addition, we assume for the targeted class, that poisoned inputs form an input distribution that is distinct from the distribution of the clean (training) images, this is in line with previous works [6], [23].

User Goals and Capabilities: The user wishes to deploy a robust machine learning model and has the necessary dataset, but does not have the technical knowledge nor the computing infrastructure to train the model. Note that it is significantly more expensive to train a robust model than a standard model, e.g. the robust training time is 25 times as much as that of the standard model training in our evaluation. Thus, we assume the user outsources the model training to an untrusted third party, and consequently needs to ascertain that there is no backdoor in the resulting robust model. To this end, the user has access to clean training data, clean testing data and white-box access to the model trained by the untrusted third party. This is in line with previous work [6], [23], where the user does not possess the computational infrastructure and technical knowledge required to train the model, but has access to clean training data and clean testing data.

Image Translation: Image translation is an active area of research in computer vision; several approaches have been developed for image to image translation [27]–[30]. Recently, it has been established that generative adversarial networks (GANs) not only learn the mapping from input image to output image, but also learn a loss function to train this

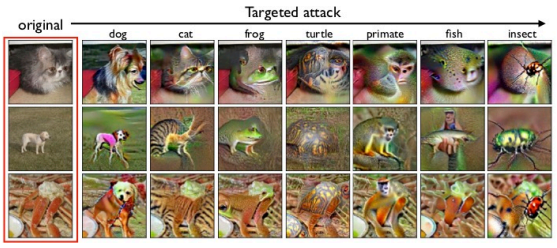


Fig. 3. Image Translation using a robust model. This figure was taken from [2]

mapping [29]. Interestingly, this behavior has also been seen in robust classifiers [2], [31], [32]. This finding enables robust classifiers to translate images from one class to another. In this paper, we apply image translation on robust classifiers to generate the perceptually-aligned representation of the image of a class. In particular, we use the adversarial robust training of [2] because it provides a means to train models that are more reliable and universal against a broader class of adversarial inputs. For instance, the images seen in Figure 3 are generated by a single CIFAR-10 classification model using first order methods, such as projected gradient descent based adversarial attacks [22]. This result is achieved by simply maximising the probability of the translated images to be classified under the targeted class.

Key Insight: If there exists a mixture of distributions in the training dataset, for a particular class, then the model will learn multiple distributions. Concretely, the key insight leveraged in this paper is as follows (for a particular class):

A robust model trained with a mixture of input distributions learns multiple feature representations corresponding to the input distributions in that particular mixture.

In this paper, we visualise the aforementioned insight in two ways. First order methods (e.g. projected gradient descent based adversarial attacks [22]) are used to generate a set of inputs $X_{y^{(i)}}$ of a particular class with label $y^{(i)}$. Let us assume these inputs are generated (by translation) via a model that has been trained using a mixture distribution containing multiple input distributions in a class with label $y^{(i)}$. Then, multiple types of inputs will be observed in the generated inputs $X_{y^{(i)}}$. Such types of inputs should correspond to the different distributions in the mixture distribution for the class with label $y^{(i)}$. Consequently, if we visualise the feature representations of the generated inputs $X_{y^{(i)}}$, then we should observe that the feature representations are distinct corresponding to the distinct distributions in the mixture distribution for the class with label $y^{(i)}$.

Formalising the insight: Let f be a robust classifier that we train. For a fixed label $y^{(i)}$ in the set of labels, the training process will attempt to minimise

$$\mathbb{E}_{x\sim\mathcal{D}}\left[\max_{\delta\in\Delta}\mathcal{L}(x+\delta,y^{(i)})\right] \tag{3}$$

Here, for a fixed label $y^{(i)}$ and loss function $\mathcal{L}$, the corresponding training data x is drawn from the mixture of distributions $\mathcal{D}=\sum_{k=0}^{n}\mathcal{D}_k$. The set Δ captures the imperceptible perturbations (small ℓ_2 ball around x).

Let us assume we attempt to generate a set of samples $X'_{y^{(i)}}$ for the class with label $y^{(i)}$ using the classifier f. We first take an appropriate seed distribution $\mathcal{G}_y$. Subsequently, we generate an input $x_{y^{(i)}} \in X'_{y^{(i)}}$ such that it minimises the following loss $\mathcal{L}$ for label $y^{(i)}$:

$$x_{y^{(i)}} = \underset{||x'-x_0||_2\leq\epsilon}{arg\,min}\ \mathcal{L}(x',y^{(i)}), \qquad x_0\sim\mathcal{G}_y \tag{4}$$

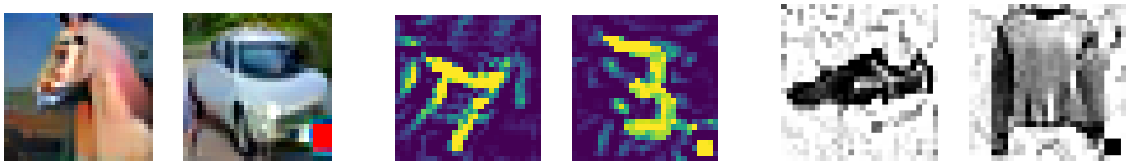
We posit that the set $X'_{y^{(i)}}$ will contain generated inputs that belong to each distribution $\mathcal{D}_0,\mathcal{D}_1,\dots\mathcal{D}_n$, which is part of the mixture of distributions $\mathcal{D}$.

Visualising the insight: To visualise this insight, we present Figure 4. The images shown in Figure 4 were generated via a model by taking random images from the corresponding dataset: CIFAR-10 for Figure 4 (a-b), MNIST digit for Figure 4 (c-d) and Fashion-MNIST for Figure 4 (f-g). This model was trained under robust optimisation conditions with poisoned training data to infect the model with backdoors. Random training data images are used to generate images of the target class in a robust backdoor-infected classifier. The classes are *Horse* in CIFAR-10, the digit *7* in MNIST-digit and the class *Sneaker* in Fashion-MNIST.

We observe the features that are maximised in Figure 4 (a, c, e) correspond to the actual classes. Whereas the counterparts seen in Figure 4 (b, d, f) correspond to the backdoor trigger (the small square at the bottom right corner of the image) used during training. We note that all images shown in Figure 4 were generated via the first order methods, as described in Santurkar et al [2], only on a backdoor-infected robust model. This led us to observe both types of images (i.e. perceptually aligned and poisoned).

In addition to the aforementioned insight, the feature representations of the poisoned images form clusters that are distinct from the clusters of feature representations of clean images [3]. However, existing works exploit this [3] via accessing both the clean and the poisoned data set. Having access to the poisoned data set is impractical for defense, as the attacker is unlikely to make the poisoned data available. In this work, we observe that the set of translated images, for a backdoor-infected robust model, contain both the clean (training) images and poisoned images. Thus, the feature representations of these images form different clusters. We use this observation to automate the detection of classes with a backdoor, without any access to the poisoned images or the training process.

Figure 5 captures the feature representations of a backdoor-infected robust model. The feature representations are the outputs of the last hidden layer of a DNN. We reduce the dimensions of the feature representations and visualise them using t-SNE [33]. In this case, we trained a robust network with a backdoor and the feature representations in Figure 5 belong to the target class (*Sneaker*). The images for this class (as generated via translation) have multiple feature representations (i.e. using projected gradient descent based adversarial attacks [22]). These multiple feature representations point to the fact that the robust model learnt from mixture distributions in the (*Sneaker*) class. Thus, a quick check of the translated images reveals two types of images – one



(a) (b) (c) (d) (e) (f)

Fig. 4. Translated images generated from mixed distributions by backdoor-infected robust model for the class *Horse* (a-b), *7* (c-d) and *Sneaker* (e-f). These are the target classes in the backdoor attack.

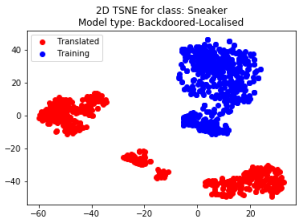


Fig. 5. Feature representations of translated images and training images (for the class *Sneaker*) for a poisoned Fashion- MNIST classifier

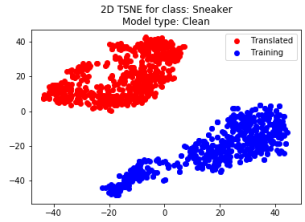


Fig. 6. Feature representations of translated images and training images (for the class *Sneaker*) for an unpoisoned Fashion-MNIST classifier

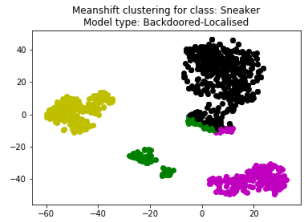


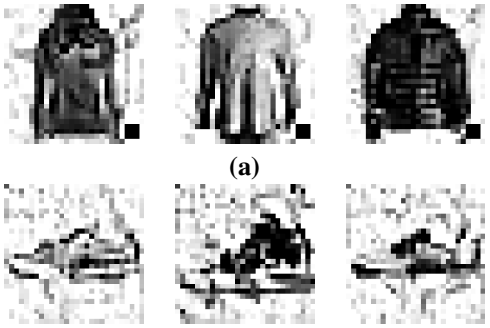
Fig. 7. Mean shift clustering of the feature representations of translated images and training images (for the class *Sneaker*) for a poisoned Fashion-MNIST classifier. The black cluster (on the top right) represents the clean training images, the purple cluster refers to the (translated) poisoned images (i.e., Figure 8 (a)), and the yellow cluster represents the (translated) clean images (i.e., Figure 8 (b)).

corresponding to the actual class *Sneaker* and one to the backdoor as seen in Figure 4 (e-f).

In contrast, Figure 6 captures the feature representations of a clean, yet robust model. The feature representations of the translated images for class *Sneaker* form only one cluster. This is expected behaviour, because the clean model learns only one distribution in *Sneaker* class. Consequently, the translated images also form only one representation that maximises the probability to be categorised in *Sneaker* class.

We observe, there are two clusters for every untargeted or clean class, specifically, the training set cluster and the translated image cluster. The translated images form a different cluster from the training set because they maximise the class probability of the training images. As a result they exaggerate the feature representations of the training set most effectively [2]. Intuitively, the translated cluster represents the "learned" representation that is influenced not only by the members of the class but also the members of every other class in the dataset. Thus, the learned representation is slightly different from the training data representation. This phenomenon leads to the translated images forming a separate cluster. It is important to note that this behavior is in line with the behaviour seen in the *robust* models in existing work [34]. We also observe this in Figure 16.

Feature Clustering: We automate the detection of clusters of feature representations by leveraging the mean shift clustering algorithm [35]. An example of applying mean shift can be seen in Figure 7, where the mean shift algorithm predicts three classes for the translated images, as generated by a backdoor-infected robust model. We further investigated the content inside these clusters by checking the images associated with the feature representations that make up these clusters. Specifically, the purple cluster (*see Figure 7*) contained inputs seen in Figure 8(a). These are the translated inputs which exhibit the backdoor. In contrast, the inputs seen in the yellow cluster (*Figure 7*) contained translated images seen in Figure 8(b). These images correspond to the features of the actual training images in class *Sneaker*.



(a)

(b)

Fig. 8. Inputs in the clusters seen in Figure 7. The purple cluster contains inputs seen in (a), where as the yellow cluster represents contains inputs seen in (b). It is important to note that these images were generated in the same instantiation of the projected gradient descent based adversarial attacks [22].

IV. Detailed Methodology

Backdoor Injection: We show that despite being highly resilient to known adversarial attacks [22], robust backdoor models are still susceptible to backdoor attacks. It takes very few poisoned training images (as little as 1% for visible backdoors) for the backdoor to be successfully injected. We use backdoor injection techniques similar to the one seen in [6] for visible backdoors and seen in [36] for invisible backdoors. We randomly select and poison one percent of the training images at random from each dataset (e.g. 500 images for CIFAR-10) for visible backdoor attacks and thirty percent (e.g. 15000 images for CIFAR-10) for invisible backdoors. The poisoning of 30% of training images for invisible backdoors is in line with the configuration in Zhong et al. [36]. We poison these images by adding the respective backdoor trigger (visible or invisible) to the images and augment them to the training data. Once this modified dataset is ready, we train the model using this data.

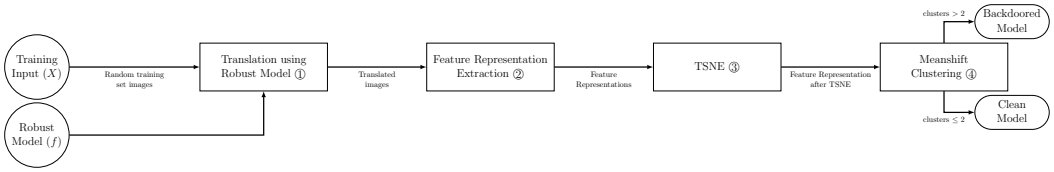


Fig. 9. Overview of the detection technique

f	The robust machine learning classifier under test.
Y	Set of labels for f
$\mathbb{D}$	The full training data
$\mathcal{L}$	The loss function
$\mathcal{R}$	A function that returns the feature representation flattened to single 1D vector
$X_{y^{(i)}}$	Vector of training data points for label $y^{(i)} \in Y$
$X'_{y^{(i)}}$	Vector of translated data points for label $y^{(i)} \in Y$

TABLE II
NOTATIONS USED IN OUR APPROACH

Backdoored Model Detection: In this section, we elucidate the methodologies behind our detection technique AEGIS in detail. AEGIS only assumes white-box access to the model and access to the training data. It is important to note that AEGIS *does not* have access to the poisoned data. In Section IV, we introduce some notation to help us illustrate our approach.

Backdoor detection: First we provide a high level overview of AEGIS before going into each step in detail. Typically, the data points of a particular class follow a single distribution and as a result, form only one cluster after undergoing t-SNE [33]. However, when a backdoor attack is carried out, the adversary inadvertently injects a mixture of distributions in one class, resulting in more than one cluster. The identification of a mixture distribution in a class is the main intuition behind our approach.

The hypothesis is that the image generation process for robust models, as seen in Santurkar et al. [2], will follow similar distributions as the training data. Since the target class in a backdoor model will be learning from multiple distributions, there will be multiple distributions of feature representation of the translated images (generated via first order adversarial methods). Our aim is to detect these multiple feature distributions. To detect such multiple distributions, we leverage t-SNE and Mean shift clustering. We also conduct an ablation study (RQ7 in section V) to compare the effectiveness of our design choices with closely-related alternative dimensionality reduction techniques and clustering methods.

For each label $y^{(i)} \in Y$, Algorithm 1 generates translated images via first order-based adversarial methods (*see Figure 9* Step 1). Then, it extracts the feature representations from the training and translated images for the label $y^{(i)}$ (*see Figure 9* Step 2). Next, the dimensions of the extracted features are reduced using t-SNE (*see Figure 9* Step 3). Mean shift is then employed to calculate the number of clusters in the reduced feature representations (*see Figure 9* Step 4). Finally, the number of resulting clusters is used to flag the backdoor-infected model (and poisoned class) as suspicious, if necessary.

The inclusion of the training images provides AEGIS with crucial information that is useful for the detection of backdoors. We note that the feature representation of backdoor images

Algorithm 1 Backdoor Detection using AEGIS

Input: Robust ML classifier f, Sample of training data points X, Sample of translated data points X', bandwidth for the mean shift algorithm b
for $y^{(i)} \in Y$ **do**
  $\triangleright$ $\mathcal{R}$ returns the activations of the last hidden layer flattened to a single 1D vector
  $R_{X_{y^{(i)}}} = \mathcal{R}(f, X_{y^{(i)}})$
  $R_{X'_{y^{(i)}}} = \mathcal{R}(f, X'_{y^{(i)}})$
  $R_{y^{(i)}} = concatenate(R_{X_{y^{(i)}}},\ R_{X'_{y^{(i)}}})$
  $\triangleright$ $tsne$ reduces the feature dimensions
  $\hat{R}_{y^{(i)}} = tsne(R_{y^{(i)}}, b)$
  $predicted\_classes = meanshift(\hat{R}_{y^{(i)}})$
  $analyseForBackdoor(\hat{R}_{y^{(i)}}, predicted\_classes)$
end for

is distinct from the feature representations of *both the clean training images and translated images (without the backdoor trigger) associated with the class*. Consequently, adding the training images in the detection process helps us avoid false positives. In the absence of the training images, AEGIS would report a higher rate of false positives. An example of such false positives is seen in Figure 17.

Step 1 - Image to Image Translation: To effectively analyse a model for backdoors, a vector of translated images $X'_{y^{(i)}}$ where $y^{(i)} \in Y$ needs to be built. In robust classifiers, image translation leads to perceptually aligned images [2]. This image translation is done for all $y^{(i)} \in Y$. The following function is minimised using stochastic gradient descent (and the probability of the target class $y^{(i)}$ is maximised):

$$x = \underset{||x'-x_0||_2 \leq \epsilon}{arg\,min}\ \mathcal{L}(x', y^{(i)}), \qquad x_0 \in \mathbb{D} \tag{5}$$

AEGIS samples a seed from the training data $\mathbb{D}$ and minimises the loss $\mathcal{L}$ of the particular label $y^{(i)}$ to generate the translated images (*see Figure 9* Step 1). This is done across 500 random seed images to obtain $X'_{y^{(i)}}$. It is important to note that there is no constraint on the labels of the seed images.

Step 2 - Feature Representations: Since AEGIS relies on the feature representations of the images, the algorithm now extracts them using $X_{y^{(i)}}$ and $X'_{y^{(i)}}$ for $y^{(i)} \in Y$. We define $\mathcal{R}$ as a function that maps an input x to a vector $\mathcal{R}(x, f)$ in the representation (penultimate layer) for a robust model f.

Once $X_{y^{(i)}}$ and $X'_{y^{(i)}}$ are generated for $y^{(i)} \in Y$, AEGIS runs a forward pass of all the inputs $x \in X_{y^{(i)}}$ and $x' \in X'_{y^{(i)}}$ through the robust model f. AEGIS extracts the outputs of the last hidden layer and flattens them to form feature representations $R_{X_{y^{(i)}}}$ and $R_{X'_{y^{(i)}}}$, for $X_{y^{(i)}}$ and $X'_{y^{(i)}}$, respectively (*see*

Figure 9 Step 2). These feature representations concatenated into $R_{y^{(i)}}$ for each $y^{(i)} \in Y$.

Step 3 - t-SNE: First introduced in [33], t-distributed stochastic neighbour embedding (t-SNE) is a data visualisation technique . It is a nonlinear dimensionality reduction algorithm, which is primarily used to visualise high dimensional data in a two or three dimensional space. t-SNE is used to visualise the feature representations $R_{y^{(i)}}$ for all $y^{(i)} \in Y$ and to reduce their dimension (*see Figure 9* Step 3). This is done to find any unusual clustering in the translated images. As expected, there are multiple clusters (> 2) of feature representations in the target class of a backdoored model. As seen in Figure 5 for a target class, the feature representations of the translated images show two clusters. This is because the learning process had inputs from two distributions (i.e. clean inputs and poisoned inputs). We have selected t-SNE for AEGIS due to its ability to group data with little assumption about the data distribution. Furthermore, we compare the effectiveness of AEGIS with t-SNE with a closely-related dimensionality reduction method (AEGIS with PCA [37]) in RQ7 (*see* section V).

Step 4 - Detection using Mean shift: To further automate the process of detection, the mean shift algorithm [35] is leveraged by AEGIS. This is a clustering algorithm which is used to identify the clusters automatically. Mean shift tries to locate the modes of a density function. It does this by trying to discover "blobs" in a smooth density of samples (*see Figure 9* Step 4). It updates candidates for centroids to be a mean of points in a given region and then eliminates duplicates to form a final set of points [35]. One can see in Figure 7 that the algorithm identifies four classes. After the mean shift, all the classes that show multiple distributions (clusters > 2) in the translated images are flagged as suspicious. A user can examine the examples in the cluster as seen in Figure 8, which helps the user to determine if the model was poisoned. In addition, we compare the effectiveness of mean-shift clustering in AEGIS to closely-related clustering methods (such as affinity propagation [38] and HDBSCAN [39]) in RQ7 (*see* section V).

V. Evaluation

In this section, we describe the experimental setup for backdoor injection attacks on adversarially robust DNN models, using three major classification tasks and several types of backdoor triggers. Overall, we employ four backdoor attack triggers including localised and distributed visible triggers, as well as static and adversarial invisible triggers. We also present the empirical results of the effectiveness of the different backdoor injection attacks on robust DNN models, as well as the detection accuracy of AEGIS in exposing backdoor attacks in robust models.

Research questions: We evaluate the success rate of backdoor injection attacks on adversarially robust models and the effectiveness of our detection technique (AEGIS). In particular, we ask the following research questions:

- **RQ1 Attack Success Rate.** How effective are backdoor injection attacks on adversarially robust DNN models? How does the effectiveness of backdoor attacks in robust DNN models compare to that of standard DNN models (i.e., Robust vs Standard)?
- **RQ2 Detection Effectiveness.** How effective is the proposed detection approach, i.e., AEGIS, in detecting all backdoor-infected models?
- **RQ3 Comparison to the state of the art.** How effective is AEGIS in comparison to the state of the art, i.e., Neural Cleanse (NC)? Is NC's performance sensitive to detection parameters, namely epsilon (ϵ), and step size?
- **RQ4 Sensitivity Analysis of Detection Parameters.** Is AEGIS sensitive to detection parameters, namely the epsilon (ϵ), mean shift bandwidth, the random sampling of initial images and the number of initial seed images?
- **RQ5 Attack Comparison.** What is the comparative performance of the different backdoor triggers in terms of attack success rate (i.e., localised vs distributed vs static perturbation vs adversarial perturbation)? Does the type or stealthiness (i.e., visibility) of backdoor triggers have an effect on AEGIS' backdoor detection?
- **RQ6 Detection Efficiency.** What is the performance of AEGIS, in terms of execution time? Is the detection efficiency of AEGIS influenced by the type or stealthiness of backdoor attack type?
- **RQ7 Ablation Study.** What is the effect of our design choices on the effectiveness of AEGIS, in comparison to closely-related alternatives, in terms of visualization (t-SNE versus PCA) and clustering (Mean Shift versus Affinity Propogation versus HDBSCAN)?

TABLE III
Dataset details and complexity of classification tasks

Image Type	Dataset (#labels)	Arch.	Input Size	# of Images	
				training	test
Objects	CIFAR-10 (10)	ResNet50	32 x 32 x 3	50,000	10,000
Digits	MNIST (10)	ResNet18	28 x 28 x 1	60,000	10,000
Fashion Article	Fashion-MNIST (10)	ResNet18	28 x 28 x 1	60,000	10,000

A. Experimental Setup

Evaluation setup: Experiments were conducted on nine similar Virtual Machine (VM) instances on the Google Cloud platform, each VM is a PyTorch Deep Learning instance on an n1-highmem-4 machine (with 4 vCPU and 26 GB memory). Each VM had an Intel Broadwell CPU platform, 1 X NVIDIA Tesla GPU with eight to 16GB GPU memory and a 100 GB standard persistent disk.

Datasets and Models: For our experiments, we use the CIFAR-10 [40], MNIST [41] and Fashion-MNIST [42] datasets. MNIST and Fashion-MNIST have 60,000 training images each, while CIFAR-10 has 50,000 training images (*see Table III*). Each dataset has 10 classes and 10,000 test images. MNIST and Fashion-MNIST models were trained with the standard ResNet-18 architecture, while CIFAR-10 was trained using the standard ResNet-50 architecture [43]. In this work, we have used the ResNet architecture for training all datasets since it is the default architecture supported by our adversarial training approach [22]. All experiments were conducted with the default learning rate (LR) scheduling in the robustness

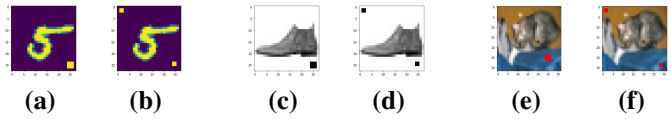


Fig. 10. Visible Triggers for MNIST (a) localised and (b) distributed backdoors, Fashion-MNIST (c) localised and (d) distributed backdoors and CIFAR-10 (e) localised and (f) distributed backdoors.

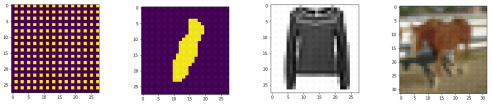


Fig. 11. Details of Static Invisible Backdoor Trigger for each dataset showing (a) the Static Invisible Trigger (note that the image intensity was increased by 25 fold to allow for visibility with the human eye), and example resulting poisoned images for (b) MNIST, (c) Fashion-MNIST and (d) CIFAR-10 showing that the poisoned image is not visible to the human eye

package [34], i.e., the PyTorch StepLR optimisation scheduler. The learning rate is initially set to 0.1 for training (LR) and the scheduler decays the learning rate of each parameter group by 0.1 (gamma) every 50 epochs (default step size). All models were trained with momentum of 0.9 and weight decay of $5e^{-4}$. Only CIFAR-10 models were trained with data augmentation [2], with momentum of 0.9 and weight decay of $5e^{-4}$.

Adversarial Training: Some approaches have been proposed to guarantee adversarial training of machine learning models [22], [44]–[47]. Notably, Wong et al [44], [45] aim to train models that are provably robust against norm-bounded adversarial perturbations on the training data. Sinha et al. [46] and Raghunathan et al. [47] are focused on training and guaranteeing the performance of ML models under adversarial input perturbations. However, the aforementioned approaches either consider very small adversarial perturbation budget epsilon (ϵ), do not scale to larger neural nets or datasets (beyond MNIST) or have a huge computational overhead.

In this paper, we apply the robust optimization approach proposed by Madry et al. [22] for adversarial training. In particular, it is computationally feasible, it provides security guarantees against a wider range of adversarial perturbations and it scales to large networks and datasets (such as CIFAR-10). For our evaluation, all models were trained with robust optimisation based on the adversarial training approach [22] with an l_2 perturbation set. The parameters for robust training are the same for all datasets (*see Table XIV in Appendix A*). In particular, all models were trained with an adversarial attack budget of 0.5 (ϵ), and an attack step size of 1.5 (step size) and set to take 20 steps (# steps) during adversarial attack. All other hyperparameters are set to the default hyperparameters in the robustness package [34]. No hyperparameter tuning was performed for the adversarial training of models.

Training Time: Table IV highlights the average training time for each dataset, model type and backdoor attack trigger. *Robust model training is expensive, it is significantly more expensive to train a robust model than a standard model*. Robust training time is 25 times as much as that of standard model training. It took about 25 hours (1,475 minutes) to train a robust model

[2]This is the default configuration in the robustness package for CIFAR-10

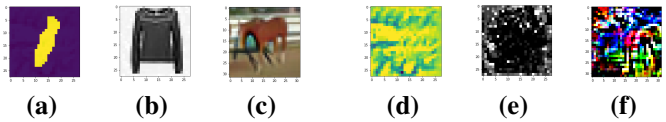


Fig. 12. Poisoned images for Invisible Adversarial backdoors for (a) MNIST, (b) Fashion-MNIST and (c) CIFAR-10 datasets, with their corresponding adversarial triggers (shown in d, e, f), note that the intensity of the triggers were increased by 10 fold to be visible to the human eye

and less than an hour (58 minutes) to train a standard model, on average (*see Table IV*). For backdoor-infected models, robust training time is 23 times as much as that of standard training time, on average. Table IV shows that it took about 21 hours (1,284 minutes) to train a robust backdoor-infected model and less than an hour (56 minutes) to train a standard backdoor-infected model. Meanwhile, robust training time is 34 times as much as standard training time for clean models. In particular, it took about 37 hours (2,238 minutes) to train a robust clean model and about an hour (66 minutes) to train standard clean model (*see Table IV*). Generally, *it is slightly cheaper to inject a backdoor in a model than to train a clean model*. In our experiments, it is less expensive to train a backdoor-infected model in comparison to a clean model.

Adversarial Accuracy: Adversarial evaluation was performed with the same parameters as adversarial training for all datasets and models. In particular, all classifiers were evaluated with an adversarial attack budget of 0.5 (ϵ), and an attack step size of 1.5 and set to take 20 steps during adversarial attack. In addition, for adversarial evaluation, we use the best loss in PGD step as the attack ("use_best": *True*), with no random restarts ("random_restarts": 0) and no fade in epsilon along epochs ("eps_fadein_epochs": 0). Table V shows the average adversarial accuracy of our clean and backdoor-infected trained models for each dataset. *In our evaluation, adversarial training accuracy is not inhibited by the backdoor attack vector.* All trained robust models maintained a similarly high adversarial accuracy for both clean and backdoor-infected models. Specifically, Table V shows that *backdoor-infected robust models have 83.21% adversarial accuracy*, on average. In contrast, *clean robust models have a slightly higher adversarial precision of 86.37%*, on average (*see Table V*).

Visible Backdoor Triggers: For visible backdoor triggers, we employed the backdoor data poisoning approach outlined in BadNets [6] to inject backdoors during adversarial training. For all datasets, we created a set of backdoor infected images by modifying a portion of the training datasets, specifically we apply a trigger to one percent (1%) of the clean images in the training set (e.g., 600 images for the MNIST dataset). Additionally, we modify the class label of each poisoned image to class seven (7) for all datasets and all attack types, then we train DNN models with the modified training data to 100 epochs for Fashion-MNIST and MNIST, and 110 epochs for CIFAR-10.

Invisible Backdoor Triggers: We employed the technique described in Zhong et al. [36] to construct two types of invisible backdoors, namely static and adversarial backdoors (*see Figure 11, Figure 12*). To allow for a reasonable attack success rate for the invisible triggers, we created a set of backdoor infected images for each dataset by modifying 30

TABLE IV
DETAILS OF TRAINING TIME FOR STANDARD VERSUS ROBUST MODELS FOR EACH DATASET, EACH BACKDOOR TRIGGER AND CLEAN MODELS (IN MINS)

Model Type	TRAINING TIME (in mins)															
	MNIST					Fashion-MNIST					CIFAR-10					AVERAGE
	Backdoor-Infected					Backdoor-Infected					Backdoor-Infected					All
	Visible		Invisible		Clean	Visible		Invisible		Clean	Visible		Invisible		Clean	(Clean/
	Local	*Dist*	*Static*	*Adv*		*Local*	*Dist*	*Static*	*Adv*		*Local*	*Dist*	*Static*	*Adv*		*Backdoor-infected*)
Robust	2971	1321	242	220	1800	2971	162	109	132	3031	1871	3276	1183	948	1882	1475 (2238/*1284*)
Standard	20	45	3	2	22	50	62	2	1	41	141	172	108	66	135	58 (66/*56*)

percent (30%) of the clean images in the training set (e.g., 18,000 images for the MNIST dataset) and modifying the class label of each poisoned image to class seven (7). The poisoning of 30% of training images for invisible backdoors is in line with the configuration in Zhong et al. [36]. We then train DNN models with the modified training data to 100 epochs for Fashion-MNIST and MNIST, and 110 epochs for CIFAR-10. For this attack, we employ the most stealthy invisible triggers (least intensity), which has a lower attack success rate (ASR). The resulting ASR is in line with the results seen in [36] which shows ASRs as low as about 30% for static backdoors with the least intensity and about 55% for adversarial backdoors with the least intensity.

Attack Configuration: The triggers for each visible backdoor attack and tasks are shown in Figure 10. The trigger for localised backdoors is a square at the bottom right corner of the image, this is to avoid covering the important parts of the original training image. The trigger for distributed backdoors is made up of two smaller squares, one at the top left corner of the image and another at the bottom right corner. The total size of the trigger is less than one percent of the entire image for both of these visible backdoor triggers.

For the invisible attacks the triggers are seen in Figure 11 and Figure 12. The static backdoor trigger is seen in Figure 11 (a). It is important to note that the trigger image is enhanced to view the trigger with ease. The actual poisoned images for the invisible static backdoor attack are seen in Figure 11 (b, c, d). Similarly, we use the adversarial perturbation-based invisible backdoor attack described in Zhong et al. [36] to generate invisible backdoors which are adversarial in nature. The images with backdoor trigger for MNIST, Fashion-MNIST and CIFAR-10 are seen in Figure 12 (a, b, c) and the enhanced triggers are seen in Figure 12 (d, e, f) respectively.

Detection Configuration: The detection configuration used in our evaluation are shown in Table XV (Appendix A). For all datasets, we have conducted a preliminary controlled experiment of detection parameters (*see* RQ4). This is to determine the best parameter for backdoor detection using AEGIS, without over-fitting. For each dataset, the epsilon (ϵ) ball for input perturbation is fixed. For MNIST and Fashion-MNIST, the parameter ϵ is 100 and it is 500 for CIFAR-10. This places a uniform limit on input perturbation for each dataset. The perplexity for t-SNE is a tuneable parameter that balances the attention between the local and global aspects of the data. The authors suggest a value between five and 50 [33] and as a result we chose 30. The bandwidth in the mean shift algorithm is the size of the kernel function. This value is constant for each dataset, it is automatically computed with the scikit-learn mean shift clustering algorithm. [3] For the backdoor attacks, the resulting bandwidths are 35, 28 and 21 for MNIST, Fashion-MNIST and CIFAR-10, respectively. Additionally, we also test the sensitivity of the AEGIS technique to variance in the bandwidth, and (the number of) initial seed images (*see* **RQ4**). For instance, we run AEGIS with $\pm$ 3 around the respective calculated values for mean shift bandwidth.

Evaluation Metrics: We measure the performance of the backdoor injection attack by computing the *classification accuracy* on the testing data. We compute the *attack success rate* (ASR) by applying the trigger to all test images and measuring the number of modified images that are classified to the attack target label, i.e., classified to class seven (7). We also measure the *adversarial precision* of all robust models. In addition, we measure the classification accuracy of the clean adversarially robust models as a baseline for comparison. We also compare the performance of robust models (i.e., ASR and classification accuracy) to that of standard backdoored (and clean) models. For detection efficacy, we report the *number of feature representation clusters* found for all classes of all robust models.

B. Experimental Results

RQ1 - Attack Success Rate (ASR):

In this section, we present the effectiveness of backdoor injection attack. We illustrate that backdoors can be effectively injected in robust models without significantly reducing the classification accuracy and adversarial precision of the models. Table V highlights the attack success rate (ASR), classification accuracy and adversarial precision of each trained model.

In our evaluation, we found that *robust models are highly vulnerable to backdoor attacks*. Backdoor attacks effectively caused the misclassification of 67.8% of backdoor-infected images to the attacker selected target labels, across all datasets and attack types (*see Table V*). *Visible backdoor triggers are generally more effective than invisible backdoor triggers*, visible triggers are 2.5 times more successful than invisible triggers (*see attack success rate ("ASR") in Table V*). Specifically, visible triggers effectively caused the misclassification of 96.4% of backdoor-infected images to the attacker selected target labels, in comparison, invisible triggers caused the misclassification of only 39.3% of infected images to the

[3] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.estimate_bandwidth.html

TABLE V
DETAILS OF ATTACK SUCCESS RATE (ASR), CLASSIFICATION ACCURACY AND ADVERSARIAL PRECISION FOR EACH DATASET, EACH BACKDOOR TRIGGER AND CLEAN MODELS

Model Type	Dataset	Measure	Backdoor-Infected				Clean	AVERAGE Backdoor-Infected					Clean
			Visible		Invisible			Visible		Invisible		All	
			Local	*Dist*	*Static*	*Adv*		*Local*	*Dist*	*Static*	*Adv*		
Robust Models	MNIST	ASR	99.96	100.00	37.53	59.87	N/A	92.93	99.87	30.65	47.86	67.83	N/A
		Class. Acc.	*99.59*	99.53	*98.94*	*98.31*	*99.61*	*93.74*	*93.85*	*89.71*	*88.89*	*91.55*	*93.96*
		(Adv. Prec.)	*(99.51)*	*(99.49)*	*(97.72)*	*(97.27)*	*(99.55)*	*(86.18)*	*(86.11)*	*(80.79)*	*(79.77)*	*(83.21)*	*(86.37)*
	Fashion-MNIST	ASR	96.26	99.77	33.33	61.00	N/A						
		Class. Acc.	*91.83*	*91.8*	*88.38*	*87.99*	*91.99*						
		(Adv. Prec.)	*(90.78)*	*(90.66)*	*(83.56)*	*(80.66)*	*(90.91)*						
	CIFAR-10	ASR	82.58	99.85	21.08	22.72	N/A						
		Class. Acc.	*89.8*	*90.22*	*81.82*	*80.38*	*90.28*						
		(Adv. Prec.)	*(68.26)*	*(68.17)*	*(61.1)*	*(61.37)*	*(68.64)*						
Standard Models	MNIST	ASR	99.97	99.96	38.59	25.3	N/A	98.87	99.91	60.36	44.29	75.86	N/A
		Class. Acc.	*99.57*	*99.53*	*97.5*	*98.06*	*99.53*	*94.95*	*95.12*	*91.83*	*89.05*	*92.74*	*95.13*
		(Adv. Prec.)	*(99.1)*	*(99.18)*	*(94.92)*	*(96.55)*	*(99.13)*	*(58.98)*	*(61.94)*	*(55.33)*	*(54.70)*	*(57.74)*	*(58.48)*
	Fashion-MNIST	ASR	97.5	99.81	43.47	54.56	N/A						
		Class. Acc.	*91.11*	*91.35*	*86.28*	*86.29*	*91.43*						
		(Adv. Prec.)	*(75.99)*	*(86.44)*	*(69.46)*	*(66.43)*	*(76.29)*						
	CIFAR-10	ASR	99.14	99.97	99.02	53.02	N/A						
		Class. Acc.	*94.18*	*94.47*	*91.72*	*82.79*	*94.42*						
		(Adv. Prec.)	*(1.86)*	*(0.2)*	*(1.62)*	*(1.13)*	*(0.01)*						

target class (*see Table V*). These results suggest that backdoor injection attacks are highly effective on robust models.

Robust DNNs are highly susceptible to backdoor attacks, with a 67.8% attack success rate (ASR), on average.

In our experiments on robust optimization via adversarial training (AT), we observed that *robust models are less susceptible to backdoor attacks than standard models.* Backdoor attacks are more successful on standard models than robust models because adversarial perturbations introduced during adversarial training may influence the shape and dimension of the backdoor trigger. We found that a backdoor attack is 12% more effective on a standard DNN model than on a robust model, with ASR of 67.83% and 75.86% for a robust and standard backdoor-infected model, on average, respectively (*see Table V*). This result holds across attack types and regardless of the stealthiness (or visibility) of the backdoor trigger. For instance, the ASR for invisible static perturbations is 30.7% on robust models, in comparison to 60.4% on standard models. Our results imply that backdoor attacks are more effective in a standard model than a robust model (resulting from AT).

Backdoor attacks are (12%) more effective on standard DNN models than robust models obtained via adversarial training.

Backdoor injection in robust DNNs does not cause a significant reduction in adversarial precision. Backdoor injection in robust models only reduced adversarial precision by about 3.7%, in comparison to clean robust models. Backdoor-infected robust models have an adversarial precision of 83.21% on average, while clean robust models have an adversarial precision of 86.37% on average (*see "Adv. Prec." in Table V*). In particular, the adversarial precision of robust models injected with visible triggers (86.14%) is comparable to that of clean robust models (86.37%). This result suggests that backdoor injection has little or no effect on the adversarial precision of infected robust models.

Backdoors do not significantly reduce the adversarial precision of robust models, they caused only 3.7% reduction, on average.

In our evaluation, backdoor injection in robust DNNs does not cause a significant reduction in classification accuracy for clean images. Overall, backdoor-infected robust models have about 2.6% reduction in classification accuracy in comparison to clean robust models, on average. Despite backdoor injection, robust models still achieved a high classification accuracy (91.55%) for clean images, on average (*see "Class. Acc." in Table V*). In comparison, clean robust models achieved a 93.96% classification accuracy. This is not a significant reduction in classification accuracy. In particular, models trained with visible triggers maintained a higher classification accuracy than models trained with invisible triggers. Models trained with visible triggers had a classification accuracy of 93.80% while models trained with invisible triggers had a lower classification accuracy of 89.30% (*see Table V*). These results imply that backdoor injection in robust models does not significantly influence the classification accuracy of clean images.

Robust backdoor-infected models maintain a high classification accuracy (83.21%), on average.

RQ2 - Detection Effectiveness: In this section, we evaluate the efficacy of our backdoor detection approach (AEGIS). Specifically, we evaluate the technique's efficacy in (a) detecting backdoor-infected robust models, and (b) revealing the backdoor-infected class, for both visible and invisible backdoor triggers. Furthermore, we demonstrate that AEGIS is *specialized to detecting backdoors in robust models* by showing it is ineffective on standard (non-robust) models. In particular,

TABLE VI
BACKDOOR DETECTION EFFICACY: ✓ INDICATES THAT AEGIS DETECTED A BACKDOORED-INFECTED MODEL/CLASS AND ✗ INDICATES THAT AEGIS DID NOT (OR FAILED TO) DETECT THE PRESENCE OF A BACKDOORED MODEL/CLASS, E.G., IN CLEAN MODELS (OR STEALTHY STATIC INVISIBLE BACKDOOR-INFECTED MODELS)

	MNIST					**Fashion-MNIST**					**CIFAR-10**				
	Backdoor-Infected				**Clean**	**Backdoor-Infected**				**Clean**	**Backdoor-Infected**				**Clean**
	Visible		**Invisible**			**Visible**		**Invisible**			**Visible**		**Invisible**		
	Local	*Dist*	*Static*	*Adv*		*Local*	*Dist*	*Static*	*Adv*		*Local*	*Dist*	*Static*	*Adv*	
Backdoor Detection	✓	✓	✗	✓	✗	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗
Backdoor Class Detection	✓	✓	✗	✗	✗	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗
False Positive Class Detection	0	0	0	1	0	0	0	3	1	0	0	0	0	1	0

TABLE VII
EFFECTIVENESS OF AEGIS ON STANDARD (NON-ROBUST) MODELS. WE SHOW THE NUMBER OF CLUSTERS PRODUCED BY AEGIS FOR EACH CLASS, USING A CLEAN STANDARD CIFAR-10 MODEL AND A CIFAR-10 MODEL POISONED WITH A VISIBLE LOCALIZED BACKDOOR TRIGGER. THE NUMBER OF CLUSTERS PRODUCED BY AEGIS FOR THE UNDETECTED POISONED CLASSES (I.E., TWO CLUSTERS FOR THE POISONED CLASS (7)) IS IN **BOLD**. ✗ INDICATES THE BACKDOOR-INFECTED CLASS/MODEL IS UNDETECTED, AND "N/A" MEANS "NOT APPLICABLE".

Standard (non-robust) Model Settings		**Number of clusters produced by AEGIS**										**Backdoor detected**	
		Benign classes									**Poisoned class**		
Dataset	**Backdoor Trigger**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**9**	**7**	**Class**	**Model**
CIFAR-10	Clean	2	2	2	2	2	2	2	2	2	2	N/A	N/A
	Localized (Visible)	2	2	2	2	2	2	2	2	2	**2**	✗	✗

we show that AEGIS did not detect a backdoor for clean standard models, and backdoor-infected standard models.

Visible Backdoor Trigger: In our evaluation, AEGIS effectively detected all visible backdoor-infected robust DNNs, for both localised and distributed backdoors, and all classification tasks. It accurately detected all backdoor-infected models by identifying classes that have more than two feature clusters for the training set and the translated image set. The results showed that all clean untargeted classes of backdoor-infected robust models, as well as all classes of clean robust models have exactly two clusters, while, all targeted classes of backdoor-infected models have more than two clusters. These imply that AEGIS detected all robust models infected with visible backdoor triggers and the corresponding target class. Additionally, there are no false positives. This means that a clean model is not incorrectly predicted as a backdoor-infected model (*see Table VI*).

In particular, for each targeted class, the mean shift clustering of the features of the backdoor-infected models reveals these models consistently have more than two clusters (*see Figure 14* in the Appendix). Notably, these clusters include one cluster for the clean training images and at least two clusters for the translated images. The clusters for the translated images include at least one cluster capturing the image translation for the poisoned images, and another cluster for the translated clean images. Meanwhile, the clean untargeted classes have precisely two clusters of features, one for the training set and another for the translated image set. Likewise, for the clean robust models, each class has exactly two distinct clusters, one cluster for the training set and another cluster for the translated image set (*see Table XVI* in the Appendix).

AEGIS *effectively detected all (100%) visible trigger backdoored robust DNNs.*

Invisible backdoor triggers: Our evaluation results show AEGIS detected five (out of six) invisible backdoor-infected robust DNNs. Specifically, AEGIS was unable to detect the MNIST backdoor model with the invisible static trigger. It accurately detected the backdoor-infected models by identifying classes that have more than two feature clusters for the training set and the translated image set. In terms of the detection of the target backdoored class, AEGIS is able to detect the targeted backdoor class in four out of the six models with invisible backdoors. AEGIS is unable to detect the target class for the MNIST backdoor model with the adversarial static trigger (*see Table VI*). Additionally, for some of the backdoor models AEGIS detected more than two clusters for the non-targeted classes (*see Table XVII* in the Appendix). On average, AEGIS detected a non-targeted class as a backdoored class (false positive detection) 11.1% of the time (*see Table VI*).

AEGIS accurately identified the infected class, for all classification tasks and both visible trigger backdoor attacks (*see Table VI*). The mean shift feature clustering of each class in the backdoor-infected model reveals that only the infected class had more than two clusters, with one cluster for the training set and at least two clusters for the translated images. For invisible backdoor attacks, AEGIS identified five out of six backdoored models and four out of the six targeted classes.

Overall, AEGIS *detected 91.6% (11/12) of backdoor-infected models, across all (12) tested configurations.*

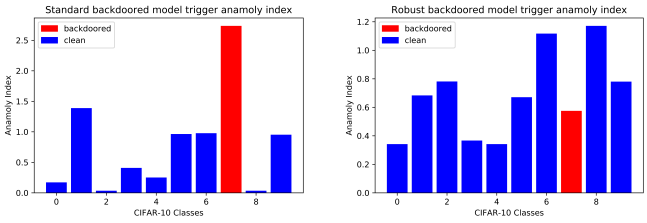
Standard (Non-robust) Models: In this experiment, we investigate if AEGIS detects backdoors in standard (non-robust) models using two standard CIFAR-10 models, namely one clean model, and a poisoned model injected with a localized backdoor trigger.

We found that AEGIS *is specialized for backdoor detection in robust models: It is ineffective in detecting backdoors in standard models*. Table VII provides details of the effectiveness of AEGIS on clean and backdoor-infected standard models. AEGIS correctly predicts clean standard models as benign, i.e., a clean model is not incorrectly predicted as a backdoor-infected model. However, AEGIS does not detect a backdoor-infected model or class for the poisoned model. Specifically, AEGIS produces exactly two clusters for all classes in both models including the poisoned class in the backdoor-infected model (*see* Table VII). Thus, AEGIS does not detect the backdoor-infected model or the poisoned class for standard (non-robust) models. These results imply that AEGIS is not directly amenable to standard models. Even though AEGIS has no false positives (it does not incorrectly classify a clean standard model), it is unable to detect a backdoor-infected standard model. This is expected since AEGIS expects a data distribution typically found in robust models. Unlike standard models, robust models have a different data distribution. In particular, AEGIS is designed to handle the resilience of robust models to perturbations introduced during adversarial training, and such perturbations are uncommon in standard models.

AEGIS *is specialized for backdoor detection in robust models.*
It is not effective in detecting backdoors in standard (non-robust) models.

RQ3 Comparison to the state of the art. In this section we compare our backdoor detection approach (AEGIS) to the state of the art backdoor detection technique called Neural Cleanse (NC) [1]. NC is a reverse engineering approach that assumes *the reverse engineered trigger for the backdoor-infected class is smaller than the median size of the reverse engineered trigger for all classes*. Specifically, NC's outlier detector identifies *a class as backdoor-infected (with 95% probability) if it has an anomaly index that is larger than two*. Although, this assumption holds for standard models because the underlying distribution of data points is normal [1], it does not hold for robust models. Due to the unbrittle nature of robust models [22], the underlying distribution of data points does not form a normal distribution because of adversarial perturbations introduced during robust training.

To compare NC and AEGIS, we run NC to detect localised backdoors in a standard model and a robust model. First, we train standard and robust models for CIFAR-10 that are poisoned with localised backdoors (using the backdoor injection process described in Section IV). We then reverse engineer the trigger for both the standard and robust backdoor-infected models using projected gradient descent on 100 random images from the training set [22], using the default NC detection parameters for both the standard and robust models. Next, we estimate the anomaly index for each class, i.e., the size of the trigger for each class by measuring the average L_1 norm deviation from the original images to the reverse-engineered images (this is equivalent to counting the number of pixels changed). The mean L_1 norms are shown in Figure 18.



(a) Standard model **(b) Robust model**

Fig. 13. Anomaly indices for the reverse engineered triggers for backdoor-infected standard and robust models

Additionally, we repeat the same experiment for six varying values (range = ± 3) for each detection parameter (i.e., epsilon $\epsilon = \{1, 2, 3, 5, 6, 7\}$ and step-size $= \{1, 2, 3, 5, 6, 7\}$) to ensure the obtained results are not due to NC's sensitivity to detection parameters. Table VIII and Table IX highlight the results for NC's effectiveness for varying values of epsilon (ϵ) and step-size, respectively.

Our evaluation results show that *NC detects the poisoned class for standard models, but it fails to accurately detect the poisoned class for robust models*. This result holds for all tested detection parameter configurations. Particularly, Table VIII and Table IX show that NC does not detect the poisoned class for (14) different parameter settings. In contrast, AEGIS detected the backdoor-infected robust model as well as the poisoned class (*see* ***RQ2***). Figure 13 shows the anomaly indices for each class, i.e., the estimated size of the reverse engineered trigger, for a standard backdoor-infected model (a) and for a robust backdoor-infected model (b). The red bar represents the anomaly index for the backdoor-infected class. We found that on standard models, the size of the backdoor-infected class is small and it is indeed detected as anomalous by NC, i.e., the anomaly index of the poisoned class (class seven (7)) is greater than two (2) (*see Figure 13(a)*). However, on robust models, NC fails to detect the poisoned class as anomalous. In fact, the anomaly index of the backdoor-infected class in the robust model is significantly less than two (*see Figure 13(b)*). This result suggests that while NC is suitable for backdoor detection in standard models, it is not suitable for detecting backdoor in robust models.

The state-of-the-art backdoor defense (Neural Cleanse) fails to accurately detect the backdoor-infected class for robust models, for (14) different detection settings with varying values of epsilon (ϵ) and step-size.

RQ4 - Sensitivity Analysis of Detection parameters: We evaluate the sensitivity of AEGIS to varying values of the detection parameters, i.e.,epsilon (ϵ), mean shift bandwidth and (number of) initial seed images. [4] We evaluate the sensitivity of these parameters for all attacks and data sets. For these parameters, we report the *detection accuracy* and the *false positive rate* for all tested values of these detection parameters. Although the mean shift bandwidth was automatically computed using the scikit-learn mean shift clustering algorithm, we

[4]We do not evaluate the sensitivity of the t-SNE perplexity parameter, because this has been shown to be robust between values five and 50 [33].

TABLE VIII
DETAILS OF THE PARAMETER SWEEP WITH VARYING EPSILON (ϵ) VALUES ($\epsilon \in \{1-7\}$) FOR NEURAL CLEANSE. WE SHOW THE ANOMALY INDICES PRODUCED BY NEURAL CLEANSE FOR EACH CLASS, USING A CIFAR-10 ROBUST MODEL POISONED WITH A VISIBLE LOCALISED BACKDOOR TRIGGER. ANOMALY INDICES FOR UNDETECTED POISONED CLASSES (I.E., ANOMALY INDEX LESS THAN TWO FOR THE POISONED CLASS (7)) ARE IN **BOLD**, AS WELL AS THE RESULTS FOR THE DEFAULT PARAMETER SETTING ($\boldsymbol{\epsilon = 4.0}$). ✗ INDICATES THE BACKDOOR-INFECTED CLASS/MODEL IS UNDETECTED BY NC, AND ✓ MEANS THE BACKDOOR-INFECTED CLASS/MODEL IS DETECTED BY NC.

Detection Setting	**Anomaly indices produced by Neural Cleanse**										**Backdoor detected**	
	Benign classes									**Poisoned class**		
Epsilon (ϵ)	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**9**	**7**	**Class**	**Model**
$\epsilon = 1.0$	0.625	0.739	0.514	1.506	0.127	0.400	1.228	0.724	0.127	**1.825**	✗	✗
$\epsilon = 2.0$	0.084	0.148	1.142	0.084	0.383	1.537	0.760	1.214	0.589	**1.611**	✗	✗
$\epsilon = 3.0$	0.475	0.373	0.796	0.291	0.291	1.133	0.733	0.852	0.620	**0.729**	✗	✗
$\boldsymbol{\epsilon = 4.0}$	**0.382**	**0.545**	**0.832**	**0.304**	**0.304**	**0.686**	**0.957**	**1.046**	**0.670**	**0.679**	✗	✗
$\epsilon = 5.0$	0.670	0.679	1.224	0.187	0.187	0.448	1.248	1.453	0.484	**1.293**	✗	✗
$\epsilon = 6.0$	0.823	0.526	1.417	0.349	0.097	0.185	1.079	1.172	0.097	**0.967**	✗	✗
$\epsilon = 7.0$	0.793	0.459	1.903	0.556	0.495	0.266	1.080	1.187	0.266	**0.934**	✗	✗

TABLE IX
DETAILS OF THE PARAMETER SWEEP WITH VARYING STEP-SIZE VALUES (STEP-SIZE$\in \{1-7\}$) FOR NEURAL CLEANSE. WE REPORT THE ANOMALY INDICES PRODUCED BY NEURAL CLEANSE FOR EACH CLASS, USING A CIFAR-10 ROBUST MODEL POISONED WITH A VISIBLE LOCALISED BACKDOOR TRIGGER. ANOMALY INDICES FOR UNDETECTED POISONED CLASS (I.E., ANOMALY INDEX LESS THAN TWO FOR THE POISOINED CLASS (7)) ARE IN **BOLD**, AS WELL AS THE RESULTS FOR THE DEFAULT PARAMETER SETTING (**STEP-SIZE$= 4.0$**). ✗ INDICATES THE BACKDOOR-INFECTED CLASS/MODEL IS UNDETECTED BY NC, AND ✓ MEANS THE BACKDOOR-INFECTED CLASS/MODEL IS DETECTED BY NC.

Detection Setting	**Anomaly indices produced by Neural Cleanse**										**Backdoor detected**	
	Benign classes									**Poisoned class**		
step-size	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**9**	**7**	**Class**	**Model**
step-size$= 1.0$	0.627	0.781	1.140	0.197	0.082	0.082	0.972	0.921	0.114	**0.722**	✗	✗
step-size$= 2.0$	0.809	0.398	1.086	0.181	0.181	0.554	0.795	0.844	0.208	**0.944**	✗	✗
step-size$= 3.0$	0.319	0.682	1.033	0.281	0.281	0.620	1.232	1.334	0.667	**0.817**	✗	✗
step-size$= 4.0$	**0.357**	**0.529**	**0.712**	**0.543**	**0.357**	**0.727**	**0.877**	**0.981**	**0.637**	**0.716**	✗	✗
step-size$= 5.0$	0.402	0.688	0.763	0.477	0.402	0.661	1.147	1.286	0.825	**0.617**	✗	✗
step-size$= 6.0$	0.306	0.590	0.806	0.297	0.297	0.802	1.253	1.204	0.663	**0.686**	✗	✗
step-size$= 7.0$	0.288	0.490	0.793	0.316	0.288	0.872	1.075	1.087	0.665	**0.684**	✗	✗

still examined the sensitivity of the resulting values with a variance of ± 3. For MNIST and FMNIST dataset, we experimented with varying epsilon values of ± 40 around the default value of 100 used, i.e.,between 60 and 140, in particular, $\epsilon \in \{60, 70, 80, 90, 100, 110, 120, 130, 140\}$. For CIFAR-10, we experiment with varying epsilon values of ± 200 around the default value of 500 used, i.e.,between 300 and 700 ($\epsilon \in \{300, 350, 400, 450, 500, 550, 600, 650, 700\}$). For all datasets, we vary the number of initial sample images ± 300 around the default value of 500 used, i.e.,between 200 and 800 ($\{200, 300, 400, 500, 600, 700, 800\}$). We also study the stability of AEGIS' detection by executing five runs for each robust model that has been infected with the visible backdoor trigger.

The epsilon sensitivity results showed that AEGIS *has a very low sensitivity to varying values of epsilon*. For all values of epsilon, AEGIS could identify a backdoor-infected model and the poisoned class for 98% (53 out of 54 configurations) of all configurations, with no false positives (*see Table X*). One backdoor-infected model was undetected, specifically, the distributed backdoor attack on MNIST at $\epsilon = 60$. We found that for the MNIST distributed backdoor attack, the epsilon value at 60 is too low. Thus, we recommend that higher epsilon (ϵ) values be used for (distributed) backdoor detection.

For all values of epsilon (ϵ), AEGIS *detected 98% of the backdoor-infected models, with no false positives.*

For mean shift sensitivity, our evaluation revealed that AEGIS *has a very low sensitivity to varying values of the mean shift bandwidth*. AEGIS detected 94% of the backdoored model for all mean shift configurations, i.e.,17 out of 18 configurations (*see Table X*). In particular, for all tested mean shift values, AEGIS did not detect a backdoored model for one value of the mean shift bandwidth. Specifically, such a mean shift value is 24 for the CIFAR-10 model poisoned with distributed backdoor. This result suggests that for values higher than the computed mean shift bandwidth value, AEGIS may not detect the backdoor-infected class. Besides, AEGIS reported two false positives. In both cases a benign class other than the poisoned class was also misclassified as backdoored by AEGIS. Specifically, false positives were manifested for MNIST localised backdoored and CIFAR-10 distributed backdoored models, both with mean shift bandwidth values less than the computed values. Hence, we recommend to use the computed mean shift bandwidth value for accurate backdoor detection.

TABLE X
SENSITIVITY TO DETECTION PARAMETERS ("#" = "NUMBER OF")

Detection Parameters	**#Config**	**#Detection Accuracy (#)**	**#Failure Rate (#)**	**#False Positive Rate (#)**
Epsilon (ϵ)	54	98.1% (53)	1.9% (1)	0% (0)
Meanshift bandwidth	18	94.4% (17)	5.6% (1)	1.2% (2)
#Images	42	88.1% (37)	11.9% (5)	2.11% (8)
Stability	30	90% (27)	10% (3)	0.7% (2)

AEGIS *has a 94% detection accuracy and a 1.2% false positive rate, for all tested mean shift bandwidth values.*

For the sensitivity of AEGIS to the number of initial seed

images, our investigation reveals that AEGIS *has a fairly low sensitivity to varying values of the number of initial images*. AEGIS detected 37 (88.1%) out of 42 tested configurations of varying number of initial seed images. Specifically, the five configurations AEGIS is unable to detect backdoors includes the MNIST localised model where the number of initial images is 300, as well as poisoned CIFAR-10 models where the number of initial images are 200 and 400 for the localised backdoors, and 200 and 300 initial images for the distributed backdoors. Overall, AEGIS has a low false positive rate of only 2.1% (*see Table X*). Hence we recommend, using at least 500 initial seed images for effective detection of backdoors.

AEGIS *has 88.1% detection accuracy and 2.1% false positive rate, for varying number of initial seed images.*

Our experiments reveal that AEGIS *is a fairly stable algorithm*. To evaluate the stability of AEGIS we run the full technique five times independently on MNIST, Fashion-MNIST and CIFAR-10 models with visible backdoor triggers. We find that out of the 30 runs, AEGIS can detect the backdoor 27 times (90%). AEGIS did not detect two MNIST distributed backdoor runs and one CIFAR-10 distributed backdoor. The false positive rate is extremely low at 0.74%. For maximum effectiveness, we recommend multiple runs of the AEGIS technique.

AEGIS *is a fairly stable algorithm with a 90% detection rate and low false positive rate of 0.74%.*

RQ5 - Attack Comparison: In this section, we compare the effectiveness of all four backdoor attack triggers namely the visible triggers (i.e., localised and the distributed triggers) as well as the invisible triggers (static perturbation and adversarial triggers). Specifically, we compare their attack success rate, and their effect on the classification accuracy and adversarial accuracy of the robust model. We also examine the detection efficacy of AEGIS on each backdoor trigger. Table V highlights the attack success rate (ASR), classification accuracy and adversarial precision of each backdoor trigger.

First, let us compare the effectiveness of backdoor attack triggers based on their stealthiness (i.e., visibility). Our results show that *robust DNN models are less susceptible to invisible triggers* (*see "ASR" Table V*). In addition, we found that visible triggers have less impact on the adversarial precision or classification accuracy of robust models, in comparison to invisible triggers. Robust models injected with visible backdoor triggers have similar adversarial precision and classification accuracy to clean robust models (*see "Adv. Prec." and "Class. Acc." in Table V*). Meanwhile, in comparison to clean robust models, invisible triggers reduce the classification accuracy and adversarial precision of robust models by 5% and 7%, respectively. These results suggest that the stealthiness (i.e.,visibility) of a backdoor trigger influences the effectiveness of the attack, in particular, visible triggers are more effective than invisible triggers.

TABLE XI
AEGIS EFFICIENCY IN TERMS OF DETECTION RUNTIME

Dataset	AEGIS Runtime			
	Visible Backdoor		Invisible Backdoor	
	Localised	Distributed	Static	Adversarial
	mins (secs)	mins (secs)	mins (secs)	mins (secs)
MNIST	5.08 (304.5)	5.18 (310.5)	5.36 (321.5)	5.24 (314.3)
Fashion-MNIST	5.36 (321.5)	5.32 (319.4)	5.28 (317.3)	5.11 (306.8)
CIFAR-10	9.39 (563.5)	9.34 (560.6)	9.29 (557.9)	9.36 (561.7)

Visible triggers are more effective and have less impact on the (adversarial) accuracy of robust models than invisible triggers.

We compare the effectiveness of the two visible backdoor attack triggers based on the specific trigger types, i.e.,localised vs distributed. *We found that the distributed backdoor attack is more effective than the localised backdoor attack, it has a higher attack success rate.* The distributed attack is 6.95% more successful than the localised backdoor attack, on average (*see Table V*). Additionally, the distributed backdoors have a higher classification accuracy than the localised backdoors, albeit only a slight improvement of 0.12%. Overall, the distributed backdoors performed better than the localised backdoors.

The distributed backdoor attack is (6.95%) more effective than the localised backdoor attack on robust models, on average.

Let us compare the effectiveness of the two invisible backdoor triggers, i.e., the static and adversarial perturbation. Table V shows that adversarial perturbation is 56% more effective than the static invisible perturbation, with 48% vs 31% ASR, on average (*see Table V*). This is because the adversarial perturbation (trigger) is dynamic and more powerful, it is derived from both the model and sample images from the dataset. Besides, the adversarial precision and classification accuracy of both triggers are similar. This result suggests that the quality of the invisible trigger influences the effectiveness of invisible backdoor attacks.

Invisible adversarial backdoor triggers are significantly more effective (56%) on robust models than static backdoor triggers.

In our evaluation, AEGIS detects 91.6% of backdoor attacks (i.e., 11 out of 12 tested backdoor-infected models). For both visible attacks, AEGIS detected the infected class in addition to the backdoor-infected model (*see Table VI*). For invisible backdoors, AEGIS was able to detect five out of the six backdoored models and four out of six poisoned classes. (*see Table VI*). We find that invisible backdoor attacks are *slightly more stealthy* in comparison to visible backdoor attacks.

AEGIS *detects 91.6% (11 out of 12) of backdoor-infected models, across all attack types (visible and invisible).*

RQ6 AEGIS Efficiency. We evaluate the detection time of AEGIS, i.e.,the time taken to run the AEGIS technique on a

backdoor-infected model. Table XI shows the time taken for each attack type and dataset.

AEGIS *is very efficient; it took five to nine minutes to run on average on a backdoor-infected model*. In contrast, the state of the art defenses (for standard models) are known to take hours to days to detect a backdoor-infected model [1], [13]. Furthermore, we observed that the time taken by AEGIS increases as the complexity of the model and dataset increases (*see Table XI*). For instance, AEGIS took almost twice the time taken to run on MNIST models (five minutes) to run on CIFAR-10 (nine minutes). In addition, there is no significant difference in the time taken to detect each attack type, i.e., localised/distributed backdoor (visible trigger) or static/adversarial trigger (invisible trigger) (*see Table XI*). These results illustrate that AEGIS is computationally efficient and its efficiency is not adversely affected by the backdoor attack type.

AEGIS *was reasonably fast, it took five to nine minutes to run on a backdoor-infected model.*

RQ7 Ablation Study. Let us evaluate the effect of our design choices on the effectiveness of AEGIS, especially in comparison to alternative design choices. The goal is to investigate how our design choices compare to closely-related, alternative methods. Particularly, we examine AEGIS's use of t-SNE for dimensionality reduction and data visualization, as well as its use of mean shift clustering. In this RQ, we employed two robust models trained for CIFAR-10 dataset that are poisoned with localized and distributed visible backdoors. Table XII and Table XIII highlight the comparison of the design choices of AEGIS to alternative design choices in terms of dimensionality reduction and clustering, respectively.

Dimensionality Reduction and Data Visualization: In this experiment, we examine the effectiveness of AEGIS with t-SNE, our default dimensionality reduction algorithm (called AEGIS-t-SNE), in comparison to replacing t-SNE with Principal Component Analysis (PCA) (called AEGIS-PCA). We compare to PCA as an alternative since it is the most popular dimensionality reduction technique for sparse datasets [37]. Besides, other dimensionality reduction alternatives are not amenable to our goal since they have strong assumptions about the underlying data distribution, e.g., Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) [48] assumes uniform data distribution.

Our experimental results show that the t-SNE algorithm is more effective in backdoor detection than the PCA algorithm for AEGIS. We found that the default setting of AEGIS (i.e., AEGIS-t-SNE) is more effective than using PCA (AEGIS-PCA). Table XII highlights the clustering provided by default AEGIS (i.e., AEGIS-t-SNE) versus AEGIS-PCA, for both the localised and distributed backdoor-infected robust models. While AEGIS-t-SNE detected the backdoored model and class, we found that AEGIS-PCA does not detect the backdoored model or the backdoored class: For both localised and distributed backdoored robust models, AEGIS-PCA does not detect the backdoored model or class, indeed, it produces two clusters for all classes, including the backdoored class (seven). We attribute the poor performance of AEGIS-PCA to the non-linearity of the dataset, since PCA is more effective when dealing with linear data. This result demonstrates that t-SNE is vital to the effectiveness of AEGIS and it is appropriate for backdoor detection in robust models.

For AEGIS*, t-SNE (*AEGIS-*t-SNE) is more effective in backdoor detection than PCA (*AEGIS-*PCA): Unlike default* AEGIS *(*AEGIS-*t-SNE),* AEGIS-*PCA does not detect the backdoor-infected model or the poisoned class.*

Clustering: To evaluate our choice of clustering algorithm, we examine the effectiveness of *default* AEGIS *with mean-shift clustering* (called AEGIS-MS) to two closely-related alternatives to mean-shift clustering, namely *affinity propagation* [38] and *HDBSCAN* [39]. Specifically, we compare the default AEGIS (AEGIS-MS), with replacing mean-shift clustering with affinity propagation (called AEGIS-AP) or HDBSCAN (called AEGIS-HDBSCAN). We chose these two clustering algorithms because they are state-of-the-art clustering methods that are closely related to mean shift clustering. Besides, they do not require prior knowledge of the (expected or desired) number of clusters unlike alternatives such as K-means [49], spectral clustering [50] or agglomerative clustering [51].

On one hand, *mean shift clustering is more effective for detecting backdoors than affinity propagation for* AEGIS. Specifically, AEGIS using affinity propagation (AEGIS-AP) does not detect a backdoor-infected robust model or a poisoned class, but default AEGIS with mean shift clustering (AEGIS-MS) detects both the backdoor-infected model and the poisoned class. Table XIII shows that unlike default AEGIS (AEGIS-MS), AEGIS-AP does not detect a backdoor-infected robust model or a poisoned class. This result suggests that affinity propagation clustering is not suitable for AEGIS, i.e., AEGIS-AP is not amenable to backdoor detection. We attribute the poor performance of AEGIS-AP to the fact that affinity propagation is inherently a partitioning algorithm which causes its resulting clusters to be easily polluted by noisy or distant data points, such noisy data points are lumped into nearby clusters using affinity propagation. This is particularly a problem for backdoor detection especially for the poisoned class since clusters belonging to the poisoned data points are evidently lumped with the clusters of benign data points (*see* Table XIII).

Affinity propagation is not a viable alternative to mean shift clustering for AEGIS*. Unlike default* AEGIS *(*AEGIS-*MS),* AEGIS *using affinity propagation (*AEGIS-*AP) does not detect a backdoor-infected model or the poisoned class.*

On the other hand, we found that *HDBSCAN is almost as effective as mean-shift clustering for detecting backdoors in robust models*. Table XIII shows that default AEGIS (AEGIS-MS) is comparable to replacing mean-shift clustering with HDBSCAN (AEGIS-HDBSCAN). Both clustering algorithms enable AEGIS to identify a backdoor-infected model and the poisoned class. Results show that AEGIS-HDBSCAN is

TABLE XII
AEGIS-T-SNE VERSUS AEGIS-PCA: EVALUATION OF DIMENSIONALITY REDUCTION/VISUALIZATION DESIGN CHOICE USING LOCALISED AND DISTRIBUTED CIFAR-10 VISIBLE BACKDOOR-INFECTED ROBUST MODELS. TRUE POSITIVES (I.E., CORRECT DETECTION OF THE POISONED CLASS) ARE IN **BOLD**.

Dataset	Type of Visible Backdoor	Detection Setting	Number of Resulting Clusters										Backdoor detected	
			Benign classes									Poisoned class		
			0	1	2	3	4	5	6	8	9	7	Class	Model
CIFAR-10	Localised	AEGIS-t-SNE	2	2	2	2	2	2	2	2	2	**3**	✓	✓
		AEGIS-PCA	2	2	2	2	2	2	2	2	2	2	✗	✗
	Distributed	AEGIS-t-SNE	2	2	2	2	2	2	2	2	2	**3**	✓	✓
		AEGIS-PCA	2	2	2	2	2	2	2	2	2	2	✗	✗

effective in detecting backdoors in robust models: For both localized and distributed backdoor-infected CIFAR-10 models, AEGIS-HDBSCAN correctly identifies the poisoned class, except for the mis-identification of class three (3) as a poisoned class for the localised backdoor (*see* Table XIII). Overall, these results demonstrate that HDBSCAN is similarly as effective as mean-shift clustering. The effectiveness of HDBSCAN is because similar to mean-shift clustering, it makes minimal assumptions about the the underlying dataset. HDBSCAN does not partition the data, and it effectively leaves sparse or noisy data points as independent clusters, hence noisy data points (e.g., poisoned data points) are not lumped with the nearest cluster. Overall, this result suggests that HDBSCAN is an effective alternative to mean-shift clustering for AEGIS.

HDBSCAN is almost as effective as mean-shift clustering for AEGIS. *Similar to default* AEGIS *(*AEGIS-*MS),* AEGIS *with HDBSCAN (*AEGIS-*HDBSCAN) detects the backdoor-infected model as well as the poisoned class.*

C. Discussions and Future Outlook

In this section, we discuss concerns about the application of AEGIS, in particular, how an adaptive attack can evade the detection of AEGIS, and how implicit assumptions of AEGIS (w.r.t. data distribution) can be applied to fool it or influence its performance.

Counter-measures against AEGIS: An attacker that is aware of AEGIS's detection methodology can ensure that clean or backdoored models are trained in a manner that tricks AEGIS and reduces its effectiveness. For instance, instead of the typical backdoor data poisoning attack vector, a powerful attacker can train a backdoor-infected model such that the backdoor image mimics neuron output values (seen in clean models). This powerful attack may evade the detection of AEGIS, such that instead of simply causing a mis-classification of the backdoored image, it fools AEGIS to believe the backdoor neuron representation is similar to the neuron representation of clean images. Likewise, an adaptive attacker can deceptively train clean models to reduce the accuracy of AEGIS. As an example, an attacker may fool AEGIS by ensuring that clean models (similar to backdoored models) also have more than one data distribution. Although, this attack does not affect the detection of backdoor models by AEGIS, it may cause false positives, where AEGIS also detects such deceptive clean models as backdoored models. In the future, we plan to investigate these more powerful attack vectors and explore potential defenses to protect against them beyond AEGIS and our current threat model.

Data Distribution assumption: In this work, we have assumed that *clean models have only one data distribution for each class label*, hence, AEGIS detects backdoors by examining if the backdoored model has more than one data distribution for a class label. Concretely, there is an implicit assumption in our method that the data corresponding to each label in the dataset contains data of only one distribution. Although, this assumption is valid within our threat model, it may not hold in other scenarios. As an example, consider a binary computer vision classifier which detects *dog* images, such that it has two classes or output labels, i.e, *dog*, and *not dog*. Consider that this classifier is trained on a dataset containing multiple animal images (e.g., cat, horse, rat etc.), which may correspond to multiple distributions for the *not dog* class. As a result, our data distribution assumption may not hold in this scenario. Besides, this assumption may lead to wrong detection of clean models as backdoored model, if the clean models also have multiple distributions, e.g., because of the limitations of the training (e.g., local optima or incomplete training), or the complexity of the task/dataset (e.g., for multi-label or multi-output classification). For instance, consider a classifier trained on a (fashion) dataset, where a data point (e.g., an image of a person wearing a piece of clothing) can be classified into multiple labels (e.g., the gender of the person, the type/size of clothing and the color of the cloth). In this scenario, the data distribution assumption may not hold for each label. As an example, consider the distribution of the "shirts" clothing label, which may contain multiple data distributions representing different gender, sizes and colors of shirts. However, in our threat model the user has access to the training data, and can examine the data distribution before-hand (e.g., through methods such as t-SNE). Thus, the user can successfully analyze whether the number of distributions learned in the trained model correspond to the actual data, and if not there may be a backdoor distribution. In a different threat model where the user does not have access to the training data, they may have no means to verify such an assumption. However, note that AEGIS can still detect a backdoored model in the absence of this assumption, i.e., even when this data distribution assumption does not hold, we expect that the backdoored model still has multiple data distributions.

In a scenario where the user has no access to the training data for inspection, then the user may not be able to determine whether our data distribution assumptions holds or not. In this threat model, an attacker can further leverage this lack of

TABLE XIII
AEGIS-T-SNE VS. AEGIS-AP VS. AEGIS-HDBSCAN: EVALUATION OF CLUSTERING DESIGN CHOICE USING LOCALISED AND DISTRIBUTED CIFAR-10 VISIBLE BACKDOOR-INFECTED ROBUST MODELS. TRUE POSITIVES (I.E., CORRECT DETECTION OF THE POISONED CLASS) ARE IN **BOLD** AND FALSE POSITIVES (I.E., INCORRECT DETECTION OF A BENIGN CLASS AS POISONED) ARE IN ***bold italics*** (E.G., AEGIS-AP).

Dataset	Type of Visible Backdoor	Detection Setting	Number of Resulting Clusters										Backdoor detected	
			Benign classes									Poisoned class		
			0	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**9**	**7**	**Class**	**Model**
CIFAR-10	Localised	AEGIS-MS	2	2	2	2	2	2	2	2	2	**3**	✓	✓
		AEGIS-AP	***19***	***19***	***34***	***17***	***196***	***21***	***73***	***23***	***145***	105	✗	✗
		AEGIS-HDBSCAN	2	2	2	***3***	2	2	2	2	2	**3**	✓	✓
	Distributed	AEGIS-MS	2	2	2	2	2	2	2	2	2	**3**	✓	✓
		AEGIS-AP	***20***	***116***	***104***	***104***	***72***	***54***	***53***	***20***	***143***	21	✗	✗
		AEGIS-HDBSCAN	2	2	2	2	2	2	2	2	2	**3**	✓	✓

data access to fool AEGIS, e.g., by ensuring that our data distribution assumption does not hold for clean models. As an example, an attacker can ensure that clean models are not properly trained, thus they converge to a local optima, hence, an *adaptive attacker* can ensure that trained clean models have more than one data distribution. Alternatively, the attacker can ensure that backdoored models also converge to a local optima with one data distribution. These attack vectors and threat models can fool AEGIS and reduce its effectiveness. However, we expect that such an adaptive attack will reduce the performance of the model (e.g., in terms of accuracy), since it forces the trained model to be sub-optimally trained. Furthermore, this assumption does not influence the ability of AEGIS to detect backdoor models, although it may cause a false positive detection (i.e., wrongly detect a clean model as a backdoored model because the clean model has more than one data distribution). In the future, we plan to investigate these line of attacks concerning alternative threat models and multiple data distributions, in order to develop potential defenses to mitigate these attack vectors.

Robustness Check: Since AEGIS is designed to detect backdoors in robust models, it assumes the analyzed model is robust, i.e., trained under robust optimization conditions. To ascertain an examined model is robust, there are automatic tests for adversarial robustness. For instance, Madry et al. [22] demonstrates a white-box approach that inspects the last layer of a model to check if a model is robust. Likewise, a brute-force black-box test is to check the performance of the model on adversarial examples within the expected perturbation bound. Both of these approaches are reliable and easy to automate. Thus, using the aforementioned methods, it is possible to check for model robustness to determine the applicability of AEGIS for the model-at-hand. Besides, we have demonstrated that AEGIS is not applicable to standard models: It does not produce false positives for standard models and it does not detect backdoors in backdoor-infected standard models (see RQ2). Thus, in the absence of a robustness check, AEGIS does not detect a backdoor in a standard model (as intended) and it does not classify a clean standard model as backdoor-infected.

Alternative Threat Models: In the following we discuss changes in the threat model that may influence the performance of AEGIS in detecting backdoor in robust models. Firstly, this work assumes the *attacker trains on the entire training dataset* (*see* section III). However, an *adaptive attacker can train on a subset of the training data* such that the model still attains an acceptable performance. In this threat model, such an attack may make it more difficult for AEGIS to detect the backdoor-infected model or class. We expect that achieving an acceptable performance requires a substantial subset of the dataset that preserves the mixed data distribution hypothesis leveraged by AEGIS. However, in the future, we plan to investigate the potential of such adaptive attacks, how much they may influence (e..g, decrease) the performance of AEGIS, and how to (extend AEGIS to) appropriately mitigate them.

We also assume an attacker directly poisons the training examples during outsourced model training, but *a sophisticated attacker may have access to the data preparation pipeline such that he/she can directly poison the objects or scenes captured in the training images*. This attack is much stronger than our default setting because of the potential *naturalness* of the resulting images. Given that the attacker only poisons for a targeted class and the mixed data distribution is preserved, we expect AEGIS to detect the backdoor in the resulting training images from such an attack. However, such a powerful attack may fool AEGIS because of the naturalness of the attack, especially if the attacker has sufficient resources to poison a substantial number of real-world objects/scenes (beyond the targeted class). Hence, such a "natural" attack may require new detection methods. We encourage researchers to investigate this "natural" backdoor attack vector and how to mitigate such attacks.

Finally, AEGIS is designed to be a white-box, post-training backdoor detection method. Hence, it may not be directly amenable to other threat settings, e.g., black-box, pre-training and in-training scenarios. For instance, consider scenarios that require detecting backdoors in poisoned datasets before (e.g., fingerprinting the information content in poisoned examples) or during the training process, AEGIS is not applicable in these scenarios since it requires white-box access to an already trained model and a clean training dataset. In addition, AEGIS may not be directly applicable to scenarios involving an ensemble of models or multiple datasets (e.g. federated learning). These scenarios are beyond the scope of this work and may require fundamentally different backdoor detection methods than AEGIS, since these threat models do not fulfill the requirements/assumptions of AEGIS. In the future, we plan to investigate these alternative threat models and how to mitigate backdoor in these settings. As an example, we plan to investigate the information content of backdoor examples to inform the automatic identification of backdoor in black-box, pre-training or in-training scenarios with no access to

the model. In addition, we plan to investigate the impact of ensemble models and multiple data sources (e.g., federated learning settings) on the effectiveness of backdoor injection and detection. We also encourage researchers to develop practical approaches to defend and mitigate against backdoor in such alternative threat models.

In summary, similar to Athena [52] – a generic defense against adversarial attacks, we encourage researchers to investigate general defense mechanisms against backdoor that are applicable to several threat models. In the same vein, we plan to investigate generic defenses that are applicable to different threat models, e.g., with varying access levels including zero-knowledge, black-box, gray-box, and white-box.

VI. Threats to Validity

Our evaluation is limited by the following threats to validity:

External validity: This refers to the generalisability of our approach and results. In particular, our findings may not generalize to other settings, different from the employed setting, specifically different (image classification) tasks, neural architectures, datasets and robust optimization methods. In the following, we discuss these threats/limitations in detail.

Tasks, Datasets and Class labels: There is a threat that our findings and approach (AEGIS) may not generalize to other classification tasks, datasets or more complex, larger labels. We have mitigated this threat by evaluating the performance of our approach using three major image classification tasks with varying levels of complexity (CIFAR-10, MNIST and FashionMNIST). These tasks have thousands of training and test images, providing confidence that our approach will work on similarly complex tasks and models. Despite this mitigation, our findings are limited to these settings. Indeed, our findings may not be applicable to other object recognition datasets, other image classification tasks (e.g. image segmentation) and other classification tasks (e.g., image captioning). Besides, our experiments involve few ($<=$10) class labels, thus it may not generalize to models with a much larger number of labels or more complex labels (e.g., multi-label classification). In the future, we plan to investigate the applicability of AEGIS to different or more complex tasks. We also plan to develop generic approaches that are applicable across several tasks.

Neural Architecture and Robust Optimization: Our experiments were conducted using a specific neural architecture and robust optimization method, in particular, the ResNet architecture [43], and adversarial training (AT) [22]. Hence, there is a threat that our findings do not generalize to simpler or more complex neural architectures where the model has less or more capacity. Besides, our findings may not generalize to other robust optimization methods beyond adversarial training. In the future, we plan to examine backdoor injection and defense across different neural architectures and robust optimization methods (e.g., adversarial defense via diversity and ensemble models [53]–[56]). For a general evaluation of backdoor in robust models, we encourage researchers to employ a standard and wide range of adversarial defenses under different threat models (e.g., AutoAttack[5] [57], [58] and RobustBench[6] [59]).

Internal validity: This concerns the correctness of our implementation of backdoor attacks and AEGIS' defense. This includes whether we have performed adversarial training rightly, accurately defined (in)visible backdoor triggers, successfully injected backdoors, and correctly implemented AEGIS. We mitigate this threat by thoroughly testing our implementations on sample images to ensure our implementation works as expected. In addition, we provide our implementation, datasets and results for replication and scrutiny.

Construct validity: It is possible that advanced backdoor triggers can be crafted to align to the input distribution of the training dataset. We mitigate this threat by ensuring that our backdoor triggers are similar to the ones described in the literature, as reported in previous related research. We emphasize that for robust models, the success and mitigation of backdoor attack variants such as blind backdoors [60], trojaning [23], [61], [62] and adaptive attacks [13] are open research problems. These attacks have not been investigated for robust models. We consider the investigation of these advanced attacks against robust models as future work.

The backdoor detection scenario employed in this work is a threat to the construct validity of AEGIS. In our attacker model, the attacker injects backdoors in robust models via a *third-party platform* and the user has access to the the clean training data, clean testing data and the trained robust model. We assume an attacker introduces poisoned examples into the training data when the model training is outsourced to a third-party. However, Li et al. [63] has shown that there are alternative processes for injecting backdoors. Specifically, backdoors can also be injected via two alternative scenarios, i.e., (i) *third-party datasets* – where an attacker provides the poisoned dataset to users directly, or (ii) *third-party models* – where the attacker provides trained infected DNNs to the user [63]. We expect that AEGIS is applicable in the *third-party dataset* scenario since the user has the capabilities required by AEGIS, i.e., access to the clean training data, clean testing data and white-box access. However, our approach (AEGIS) may not be directly applicable in the *third-party model* scenario since the user/defender may lack access to the training set and white-box access to the trained models. In the future, we shall investigate the injection and detection of backdoors in alternative attacker/defender scenarios.

Lastly, our design choices pose a threat to construct validity. Specifically, our use of *mean-shift clustering* and *t-SNE for dimensionality reduction* may influence the effectiveness of AEGIS. To mitigate this threat, we have conducted an ablation study investigating the effectiveness of our choices in comparison to alternative design choices (*see* RQ7).

VII. Related Work

Robust Optimization: Adversarial attacks for Neural Networks (NNs) were first introduced in [64]. Researchers have

[5]https://github.com/fra31/auto-attack
[6]https://robustbench.github.io/

introduced better adversarial attacks and built systems that are resilient to these attacks [21], [65]–[67]. A significant leap has been made by introducing robust optimisation to mitigate adversarial attacks [22], [68]–[70]. These defences aim to guarantee the performance of machine learning models against adversarial examples. In this paper, we study the susceptibility of the models trained using robust optimisation to backdoor attacks. Then, we leverage the inherent properties of robust models to detect backdoor attacks.

In this work, we have studied backdoor detection/injection in robust models using adversarial training (AT) as the *only* robust optimization method. Even though adversarial training is an effective and well-known defense against adversarial examples (AE), there are other robust optimization techniques beyond adversarial training which may not be susceptible to backdoor detection or amenable to backdoor detection by AEGIS. A number of researchers have demonstrated that adversarial robustness can be achieved via an ensemble of diverse models [54]–[56], or by detecting unrecognised, potentially adversarial examples [53]. Indeed, we do not know the susceptibility of these robust optimization methods (except AT) to backdoor attacks, and AEGIS may not be sufficient to defend against backdoor injection in these settings. In the future, we plan to investigate the susceptibility of different robust optimization methods to backdoor attacks and how to effectively defend against them.

In addition, our findings and observations about backdoor attacks and AEGIS is *strictly empirical*. We provide no theoretical bound to the susceptibility of robust optimization to backdoor attacks or guarantees of AEGIS's defense. Indeed, it is vital to understand how our empirical observations relate to the robust optimization theory, e.g., in terms of the susceptibility of standard and robust models to backdoor attacks (*see* RQ1). In addition, it is interesting to know the lower and upper bound accuracy of AEGIS on certain poisoning attacks (e.g., (in)visible distributed or localized backdoor triggers), and how this relates to formulating backdoor defense as studying spurious/non-robust features in robust models [71]. In particular, we encourage the theoretical investigation of how backdoor injection and defenses relate to robust optimization theory to provide mathematical insights into our empirical observations.

Backdoor Attacks: Backdoor attacks were introduced in BadNets [6], where an attacker poisons the training data by augmenting it. A pre-defined random shape (called trigger) is chosen for the attack. TrojanNN [23] improves the attack by engineering the trigger and reducing the number of examples needed to insert the backdoor. Yao et al. [72] propose a transfer learning based backdoor. All of these attacks are visible to the human eye. Besides, other variants of backdoor attacks have also recently been developed such as blind backdoors [60], trojaning [23], [61], [62] and adaptive attacks [13]. In addition, Zhong et al. proposed a backdoor attacks where the trigger is hidden [36].

Li et al. [63] provides a systematic literature review of backdoor attack mechanisms. This work demonstrates varying attack/threat models for backdoor attacks, for instance in terms of the access level of the attacker (e..g, access to training set, training schedule, model and/or inference pipeline). The paper also provides a taxonomy of poisoning based attacks (e.g., trigger properties such as target level, visibility, selection, appearance, and type (digital versus physical).) In this work (AEGIS), we have focused on the injection and detection of both invisible and visible backdoor attacks in robust models. The aforementioned attacks were demonstrated for standard models, not for robust training. To the best of our knowledge, we are the first to demonstrate the susceptibility of models trained under robust optimisation conditions [22] to (both visible and invisible) backdoor attacks.

Backdoor Detection and Mitigation: Several approaches have been developed to detect and mitigate backdoor attacks on standard machine learning models. Li et al. [63] provides a comprehensive analysis of different defense mechanisms under different threat models.[7] Table I compares an excerpt of the main characteristics of these approaches. These approaches can be categorized into three main types, namely, backdoor detection via (1) outlier suppression, (2) input perturbation and (3) model anomalies [60].

Outlier suppression based defenses prevent backdoored inputs from being introduced into the model [7], [8]. The main idea of these approaches is to employ differential privacy mechanism to ensure that backdoored inputs are under-represented in the training set. Unlike these approaches, our approach is not a training-time defense, rather the focus of our approach is to detect models that are already poisoned with backdoored inputs.

Input perturbation methods detect backdoors by attempting to reverse engineer small input perturbations that trigger backdoor behavior in the model. Such approaches include Neural Cleanse (NC) [1], ABS [9], TABOR [12], STRIP [13], NEO [5], DeepCleanse [14], AD [11] and MESA [10]. In this paper, we focus on comparison to Neural Cleanse (NC) [1], we used NC as the representative backdoor defense. We compare our approach to NC (see **RQ3**), since NC is the state of the art and it has realistic defense assumptions (similar to AEGIS) (*see Table I*). In particular, NC relies on finding a fixed perturbation that mis-classifies a large set of inputs, but since robust models are designed to be resilient to exactly such perturbations, we show that NC is inapplicable for robust models.

Model anomaly defenses detect backdoors by identifying anomalies in the model behavior. Most of these techniques focus on identifying how the model behaves differently on benign and backdoored inputs, using model information such as logit layers, intermediate neuron values and spectral representations. These approaches include SentiNet [15], spectral signatures [4], fine-pruning [17], NeuronInspect [16], activation clustering [3], SCAn [18], NNoculation [19] and MNTD [20]. However, unlike our approach, none of these techniques detect backdoors in robust models. Additionally, SCAn [18], SentiNet [15], activation clustering [3] and spectral signatures [4] assume access to the poisoned dataset – an impractical assumption for backdoor defense (*see Table I*). Moreover, fine-pruning [17] is shown to be ineffective in existing work [1] and NNoculation [19] and MNTD [20] require training a shadow model for defense,

[7] https://github.com/THUYimingLi/backdoor-learning-resources

leading to a computationally inefficient process. In contrast, AEGIS is computationally efficient, it does not require access to the poisoned dataset and it accurately detects backdoor-infected robust models.

Unlike the aforementioned works, we rely on the *clustering of feature representations in robust models* to detect backdoor attacks. Like our approach, Chen et al. [3] employs feature clustering to detect backdoors in standard DNNs; it uses the feature representations of the training and poisoned data to detect the poisoned data. However, their approach relies on *the strong assumption that the user has access to the poisoned dataset*. Our approach requires access to only the model and the clean training dataset.

Adversarial Training and Backdoor Robustness: Several researchers have studied the relationship between adversarial inputs and poisoned models (including backdoor [73], [74]). Remarkably, Pang et al. [73] systematically studied the relationship between adversarial inputs and poisoned models in a unified manner by developing a new attack model that jointly optimizes both attacks. This work shows that there is a mutual reinforcement effect between the two attack vectors which can be easily exploited to optimize attacks with respect to multiple metrics. For instance, this work shows that leveraging one attack vector significantly amplifies the effectiveness of the other. Similar to our work (AEGIS), the paper encourages the need to study both attacks by designing countermeasures, albeit from multiple complementary perspectives (e.g., efficacy, fidelity and specificity) to account for the mutual reinforcement effects.

Similarly, researchers have shown that there is a trade-off between adversarial robustness and backdoor attacks. Notably, Weng et al. [74] demonstrated that adversarial robustness is at odds with backdoor robustness. The authors found that increasing adversarial robustness via adversarial training makes a model more vulnerable to backdoor attacks. Consequently, this trade-off can influence the strength of both attacks and defenses against backdoor attacks. Weng et al. [74] shows that this trade-off can be leveraged to create more concealed backdoor attacks that evade existing backdoor defenses, and it can also be leveraged to further strengthen some defenses. Similarly, this work (AEGIS) shows that the inherent properties of adversarial training based robust optimization can aid the detection/defense against backdoor attacks in robust models. In the future, we plan to investigate the extent to which increasing or decreasing adversarial robustness may influence the success of backdoor attacks and the defense of AEGIS. In addition, we plan to investigate how to extend AEGIS to achieve joint defense against both adversarial and backdoor attacks.

VIII. Conclusion

In this paper, we demonstrate a new attack vector for PGD-trained robust DNN models, namely backdoor attacks. We show that such robust models are susceptible to several variants of backdoor attacks, including visible and invisible backdoors. Then, we leverage the inherent properties of these robust ML models to detect this attack. Our proposed detection technique (i.e., AEGIS) is based on clustering the feature representation of PGD-trained robust models to find anomalous clusters. In our evaluation, AEGIS accurately detects backdoor-infected PGD-trained robust models and identifies the poisoned class, without any access to the poisoned data, for all visible backdoor triggers. We also found that invisible backdoor triggers are more stealthy and slightly more difficult to detect for AEGIS. Overall, AEGIS detects a backdoor-infected model with 91.6% accuracy (i.e., 11 out of 12 backdoor-infected models), without any false positives. Furthermore, AEGIS detects the targeted class in the backdoor-infected model with a reasonably low (11.1%) false positive rate. Our work reveals that inherent properties of PGD-based robust optimization method allows to expose backdoors. Our code and experimental data are available for replication:

https://github.com/sakshiudeshi/Expose-Robust-Backdoors

Acknowledgment

This work was partially funded by OneConnect Financial grant number RGOCFT2001, Singapore Ministry of Education (MOE) President's Graduate Fellowship and the University of Luxembourg's Institute for Advanced Studies (IAS). Ezekiel Soremekun was funded by the IAS Audacity project titled "LAIWYERS: Law and AI: WaYs to Explore Robust Solutions".

References

[1] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *2019 IEEE Symposium on Security and Privacy, SP 2019, Proceedings, 20-22 May 2019, San Francisco, California, USA*, 2019.

[2] S. Santurkar, A. Ilyas, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Image synthesis with a single (robust) classifier," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, 2019, pp. 1260–1271.

[3] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, "Detecting backdoor attacks on deep neural networks by activation clustering," in *Workshop on Artificial Intelligence Safety 2019 co-located with the Thirty-Third AAAI Conference on Artificial Intelligence 2019 (AAAI-19), Honolulu, Hawaii, January 27, 2019*, 2019.

[4] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, 2018, pp. 8011–8021.

[5] S. Udeshi, S. Peng, G. Woo, L. Loh, L. Rawshan, and S. Chattopadhyay, "Model agnostic defence against backdoor attacks in machine learning," *CoRR*, vol. abs/1908.02203, 2019. [Online]. Available: http://arxiv.org/abs/1908.02203

[6] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *CoRR*, vol. abs/1708.06733, 2017. [Online]. Available: http://arxiv.org/abs/1708.06733

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

[8] S. Hong, V. Chandrasekaran, Y. Kaya, T. Dumitraş, and N. Papernot, "On the effectiveness of mitigating data poisoning attacks with gradient shaping," *arXiv preprint arXiv:2002.11497*, 2020.

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

[10] X. Qiao, Y. Yang, and H. Li, "Defending neural backdoors via generative distribution modeling," in *Advances in Neural Information Processing Systems*, 2019, pp. 14 004–14 013.

[11] Z. Xiang, D. J. Miller, and G. Kesidis, "Detection of backdoors in trained classifiers without access to the training set," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

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

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

[14] B. G. Doan, E. Abbasnejad, and D. Ranasinghe, "Deepcleanse: A black-box input sanitization framework against backdoor attacks on deepneural networks," *arXiv preprint arXiv:1908.03369*, 2019.

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

[16] X. Huang, M. Alzantot, and M. Srivastava, "Neuroninspect: Detecting backdoors in neural networks via output explanations," *arXiv preprint arXiv:1911.07399*, 2019.

[17] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *Research in Attacks, Intrusions, and Defenses - 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings*, 2018, pp. 273–294.

[18] D. Tang, X. Wang, H. Tang, and K. Zhang, "Demon in the variant: Statistical analysis of dnns for robust backdoor contamination detection," *arXiv preprint arXiv:1908.00686*, 2019.

[19] A. K. Veldanda, K. Liu, B. Tan, P. Krishnamurthy, F. Khorrami, R. Karri, B. Dolan-Gavitt, and S. Garg, "Nnoculation: Broad spectrum and targeted treatment of backdoored dnns," *arXiv preprint arXiv:2002.08313*, 2020.

[20] X. Xu, Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li, "Detecting ai trojans using meta neural analysis," *arXiv preprint arXiv:1910.03137*, 2019.

[21] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017*, 2017, pp. 506–519.

[22] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[23] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *25nd Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-221, 2018*. The Internet Society, 2018.

[24] "Microsoft azure cognitive services," 2021. [Online]. Available: https://azure.microsoft.com/en-us/services/cognitive-services/

[25] "Google automl," 2021. [Online]. Available: https://cloud.google.com/automl

[26] M. Du, R. Jia, and D. Song, "Robust anomaly detection and backdoor attack detection via differential privacy," *CoRR*, vol. abs/1911.07116, 2019. [Online]. Available: http://arxiv.org/abs/1911.07116

[27] Z. Yi, H. Zhang, P. Tan, and M. Gong, "Dualgan: Unsupervised dual learning for image-to-image translation," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[28] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[29] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[30] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 700–708. [Online]. Available: http://papers.nips.cc/paper/6672-unsupervised-image-to-image-translation-networks.pdf

[31] S. Kaur, J. Cohen, and Z. C. Lipton, "Are perceptually-aligned gradients a general property of robust classifiers?" *CoRR*, vol. abs/1910.08640, 2019. [Online]. Available: http://arxiv.org/abs/1910.08640

[32] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[33] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[34] L. Engstrom, A. Ilyas, S. Santurkar, and D. Tsipras, "Robustness (python library)," 2019. [Online]. Available: https://github.com/MadryLab/robustness

[35] K. Fukunaga and L. D. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Trans. Information Theory*, vol. 21, no. 1, pp. 32–40, 1975.

[36] H. Zhong, C. Liao, A. C. Squicciarini, S. Zhu, and D. Miller, "Backdoor embedding in convolutional neural network models via invisible perturbation," in *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*, 2020, pp. 97–108.

[37] A. Maćkiewicz and W. Ratajczak, "Principal components analysis (pca)," *Computers & Geosciences*, vol. 19, no. 3, pp. 303–342, 1993.

[38] D. Dueck, *Affinity propagation: clustering data by passing messages*. Citeseer, 2009.

[39] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering." *J. Open Source Softw.*, vol. 2, no. 11, p. 205, 2017.

[40] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[41] Y. LeCun, C. Cortes, and C. J. Burges, "The mnist database of handwritten digits, 1998," *URL http://yann. lecun. com/exdb/mnist*, vol. 10, p. 34, 1998.

[42] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[44] E. Wong, F. Schmidt, J. H. Metzen, and J. Z. Kolter, "Scaling provable adversarial defenses," in *Advances in Neural Information Processing Systems*, 2018, pp. 8400–8409.

[45] E. Wong and Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *International Conference on Machine Learning*, 2018, pp. 5286–5295.

[46] A. Sinha, H. Namkoong, and J. Duchi, "Certifying some distributional robustness with principled adversarial training," *arXiv preprint arXiv:1710.10571*, 2017.

[47] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," *arXiv preprint arXiv:1801.09344*, 2018.

[48] L. McInnes, J. Healy, N. Saul, and L. Großberger, "Umap: Uniform manifold approximation and projection," *Journal of Open Source Software*, vol. 3, no. 29, p. 861, 2018.

[49] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern recognition*, vol. 36, no. 2, pp. 451–461, 2003.

[50] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 14, 2001.

[51] K. C. Gowda and G. Krishna, "Agglomerative clustering using the concept of mutual nearest neighbourhood," *Pattern recognition*, vol. 10, no. 2, pp. 105–112, 1978.

[52] Y. Meng, J. Su, J. O'Kane, and P. Jamshidi, "Athena: A framework based on diverse weak defenses for building adversarial defense," *arXiv preprint arXiv:2001.00308*, 2020.

[53] M. Abbasi and C. Gagné, "Robustness to adversarial examples through an ensemble of specialists," *arXiv preprint arXiv:1702.06856*, 2017.

[54] R. Li, H. Zhang, P. Yang, C.-C. Huang, A. Zhou, B. Xue, and L. Zhang, "Ensemble defense with data diversity: Weak correlation implies strong robustness," *arXiv preprint arXiv:2106.02867*, 2021.

[55] T. Pang, K. Xu, C. Du, N. Chen, and J. Zhu, "Improving adversarial robustness via promoting ensemble diversity," in *International Conference on Machine Learning*. PMLR, 2019, pp. 4970–4979.

[56] S. Kariyappa and M. K. Qureshi, "Improving adversarial robustness of ensembles with diversity training," *arXiv preprint arXiv:1901.09981*, 2019.

[57] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *ICML*, 2020.

[58] ——, "Mind the box: l_1-apgd for sparse adversarial attacks on image classifiers," in *ICML*, 2021.

[59] F. Croce, M. Andriushchenko, V. Sehwag, E. Debenedetti, N. Flammarion, M. Chiang, P. Mittal, and M. Hein, "Robustbench: a standardized adversarial robustness benchmark," *arXiv preprint arXiv:2010.09670*, 2020.

[60] E. Bagdasaryan and V. Shmatikov, "Blind backdoors in deep learning models," *arXiv preprint arXiv:2005.03823*, 2020.

[61] C. Guo, R. Wu, and K. Q. Weinberger, "Trojannet: Embedding hidden trojan horse models in neural networks," *arXiv preprint arXiv:2002.10078*, 2020.

[62] M. Zou, Y. Shi, C. Wang, F. Li, W. Song, and Y. Wang, "Potrojan: powerful neural-level trojan designs in deep learning models," *arXiv preprint arXiv:1802.03043*, 2018.

[63] Y. Li, B. Wu, Y. Jiang, Z. Li, and S.-T. Xia, "Backdoor learning: A survey," *arXiv preprint arXiv:2007.08745*, 2020.

[64] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[65] N. Papernot, P. D. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, 2016, pp. 582–597.

[66] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, 2016, pp. 372–387.

[67] W. He, J. Wei, X. Chen, N. Carlini, and D. Song, "Adversarial example defense: Ensembles of weak defenses are not strong," in *11th USENIX Workshop on Offensive Technologies, WOOT 2017, Vancouver, BC, Canada, August 14-15, 2017*, 2017.

[68] E. Wong and J. Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, 2018, pp. 5283–5292. [Online]. Available: http://proceedings.mlr.press/v80/wong18a.html

[69] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018. [Online]. Available: https://openreview.net/forum?id=Bys4ob-Rb

[70] A. Sinha, H. Namkoong, and J. C. Duchi, "Certifying some distributional robustness with principled adversarial training," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018. [Online]. Available: https://openreview.net/forum?id=Hk6kPgZA-

[71] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," *Advances in neural information processing systems*, vol. 32, 2019.

[72] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, "Latent backdoor attacks on deep neural networks," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, 2019, pp. 2041–2055. [Online]. Available: https://doi.org/10.1145/3319535.3354209

[73] R. Pang, H. Shen, X. Zhang, S. Ji, Y. Vorobeychik, X. Luo, A. Liu, and T. Wang, "A tale of evil twins: Adversarial inputs versus poisoned models," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 85–99.

[74] C.-H. Weng, Y.-T. Lee, and S.-H. B. Wu, "On the trade-off between adversarial and backdoor robustness," *Advances in Neural Information Processing Systems*, vol. 33, pp. 11 973–11 983, 2020.

APPENDIX

TABLE XIV
STANDARD HYPERPARAMETERS USED FOR MODEL TRAINING.

Dataset	Epochs	LR	Batch Size	LR Schedule
CIFAR-10	110	0.1	128	Drop by 10 at epochs $\in [50, 100]$
MNIST	100	0.1	128	Drop by 10 at epochs $\in [50, 100]$
Fashion-MNIST	100	0.1	128	Drop by 10 at epochs $\in [50, 100]$

TABLE XV
BACKDOOR DETECTION PARAMETERS

Detection Parameters	All Models		
	MNIST	Fashion-MNIST	CIFAR-10
Epsilon (ϵ)	100	100	500
t-SNE Perplexity	30	30	30
Mean shift Bandwidth	35	28	21

TABLE XVI
DETECTION EFFICACY: NUMBER OF FEATURE CLUSTERS FOR EACH CLASS FOR CLEAN MODEL AND VISIBLE TRIGGER INFECTED BACKDOOR MODELS

Class Type	Class Labels	MNIST Models			Fashion-MNIST Models			CIFAR-10 Models		
		Backdoor-Infected		Clean	Backdoor-Infected		Clean	Backdoor-Infected		Clean
		Local	Distributed		Local	Distributed		Local	Distributed	
Targeted	$\{7\}$	**3**	**3**	2	**4**	**3**	2	**3**	**4**	2
Untargeted	$\{0-6, 8, 9\}$	2	2	2	2	2	2	2	2	2

TABLE XVII
DETECTION EFFICACY: NUMBER OF FEATURE CLUSTERS FOR EACH CLASS FOR INVISIBLE BACKDOORS

Class Type	Class Labels	MNIST Models Backdoor-Infected		Fashion-MNIST Models Backdoor-Infected		CIFAR-10 Models Backdoor-Infected	
		Static	Adversarial	Static	Adversarial	Static	Adversarial
Targeted	$\{7\}$	**2**	**2**	**3**	**3**	**3**	**4**
Untargeted	$\{0\}$	1	2	3	2	2	2
	$\{1\}$	2	2	3	2	2	3
	$\{2\}$	2	2	2	2	2	2
	$\{3\}$	2	**3**	2	2	2	2
	$\{4\}$	2	2	2	3	2	2
	$\{5\}$	2	2	2	2	2	2
	$\{6\}$	2	2	2	2	2	2
	$\{8\}$	2	2	3	2	2	2
	$\{9\}$	2	2	2	2	2	2

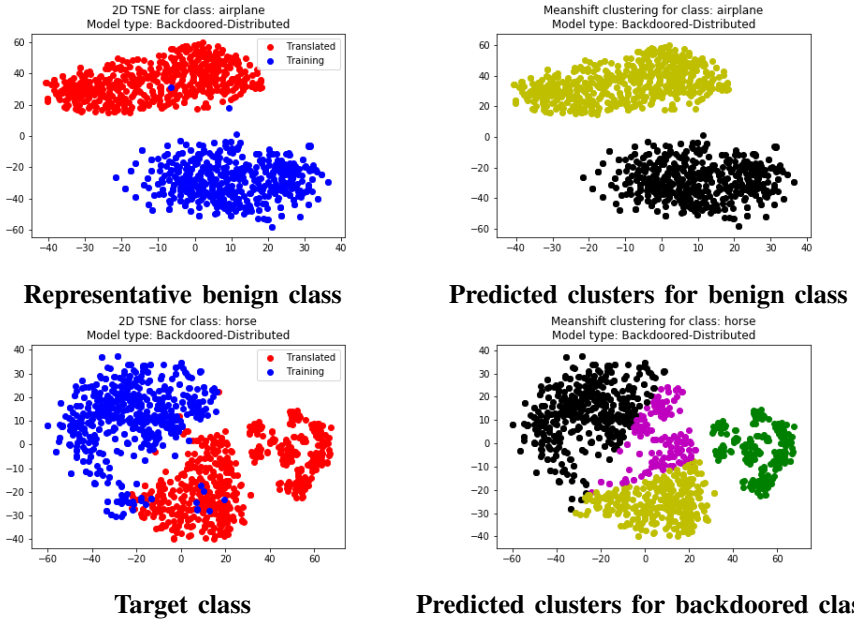


Fig. 14. Feature representation clusters for backdoored CIFAR models (Distributed) with target class *Horse* (7). This figure shows class *0* and *7*. The left column shows the feature representations of the translated and the training images, whereas the right column shows the result of the Mean shift clustering on the corresponding points where different colours represent different classes.

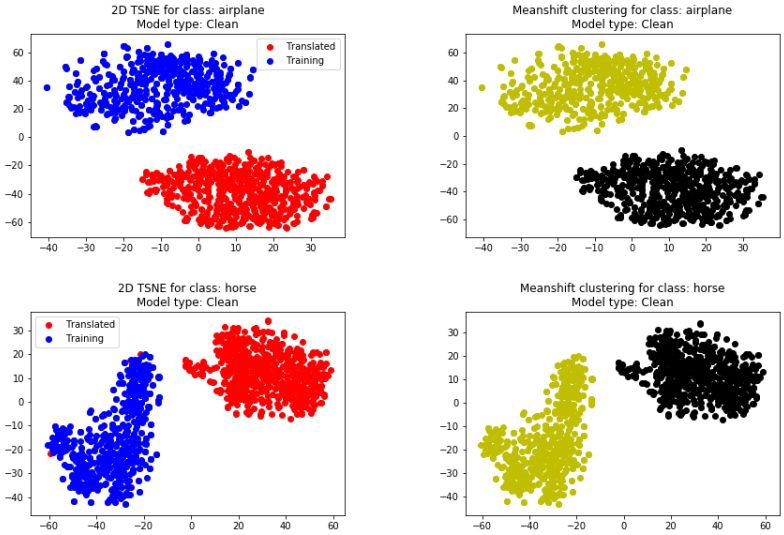


Fig. 15. Feature representation clusters for clean CIFAR10 models. This figure shows class *0* and *7*. The left column shows the feature representations of the translated and the training images, whereas the right column shows the result of the Mean shift clustering on the corresponding points where different colours represent different classes.

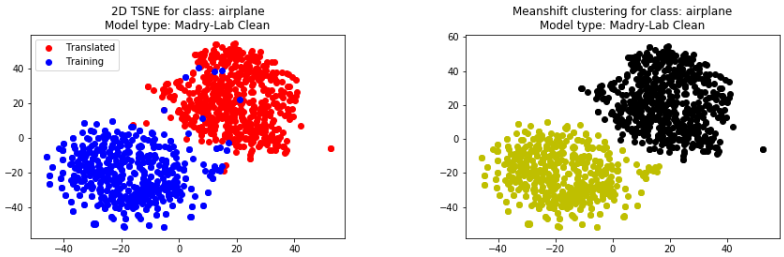


Fig. 16. Feature representation clusters for clean CIFAR10 models from Madry-Lab. This figure shows class *0*. The left column shows the feature representations of the translated and the training images, whereas the right column shows the result of the Mean shift clustering on the corresponding points where different colours represent different classes. It is important to note that the translated images and training set images form separate clusters.

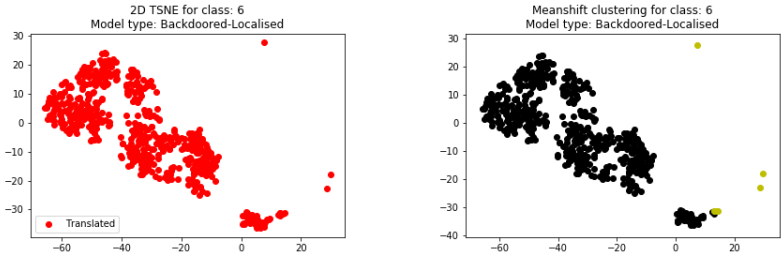
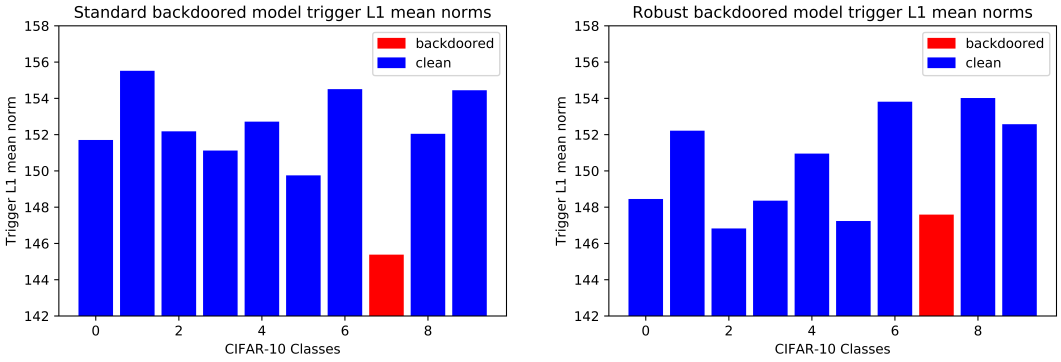


Fig. 17. Representative false positives. These kinds of false positives occur when AEGIS only considers the translated images in the detection for backdoors. This figure shows class *6* of a robust MNIST model poisoned with a localised backdoor. The left column shows the feature representations of the translated and the training images, whereas the right column shows the result of the Mean shift clustering on the corresponding points where different colours represent different classes.



(a) Standard model **(b) Robust model**

Fig. 18. L1 norms (mean) of the reverse engineered triggers for backdoor-infected standard and robust models. The L1 norms for the reverse engineered triggers are in line with the sizes of the reverse engineered triggers seen in [1].