SPECTRE: Defending Against Backdoor Attacks Using Robust Statistics

Jonathan Hayase [1] **Weihao Kong** [1] **Raghav Somani** [1] **Sewoong Oh** [1]

Abstract

Modern machine learning increasingly requires training on a large collection of data from multiple sources, not all of which can be trusted. A particularly concerning scenario is when a small fraction of poisoned data changes the behavior of the trained model when triggered by an attacker-specified watermark. Such a compromised model will be deployed unnoticed as the model is accurate otherwise. There have been promising attempts to use the intermediate representations of such a model to separate corrupted examples from clean ones. However, these defenses work only when a certain spectral signature of the poisoned examples is large enough for detection. There is a wide range of attacks that cannot be protected against by the existing defenses. We propose a novel defense algorithm using robust covariance estimation to amplify the spectral signature of corrupted data. This defense provides a clean model, completely removing the backdoor, even in regimes where previous methods have no hope of detecting the poisoned examples.[2]

1. Introduction

Large scale machine learning, such as federated learning (Kairouz et al., 2019), requires training data collected from multiple sources. As not all sources can be trusted and sanity checking the data is expensive, this opens an opportunity for an adversary to inject poisoned data into the training set. A particularly concerning scenario is the *backdoor attack*; the attacker attempts to embed a hidden backdoor in the trained model such that its prediction is maliciously changed when activated by samples with an attacker-defined trigger. As the model behavior on clean data is unchanged, such backdoored models may be deployed unnoticed.

Starting with the seminal work of (Gu et al., 2017), there has been an active line of work on designing backdoor attacks that use more stealth triggers (Chen et al., 2017; Liu et al., 2017; Li et al., 2019; Liu et al., 2020) or that can pass a human inspection (Turner et al., 2019; Zhao et al., 2020). Empirical evidence in these works suggest that a small fraction of poisoned data is sufficient to successfully create backdoors in trained neural networks. For example, CIFAR-10 data has 5,000 training examples for each of the ten classes. When the pixel attack (Gu et al., 2017) is launched with only 125 poisoned samples injected during training, the pixel attack succeeds in planting a backdoor in the trained model, achieving an attack accuracy of 63% (shown in Fig. 1 in blue triangles).

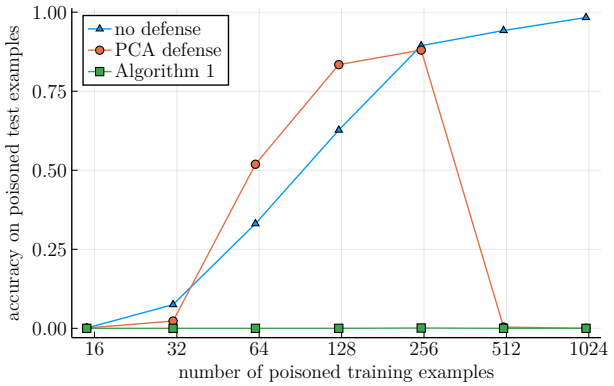


Figure 1: Under the pixel attack, the PCA defense fails to produce a clean model when the number of poisoned examples is between 64 and 256 (red circle). In fact, it removes clean data samples resulting in a model with higher accuracy on the poisoned test examples than when no defense was applied (blue triangle). SPECTRE produces clean models with the backdoor completely removed in all regimes (green square).

Recently, Tran et al. (2018) proposed, what we call, the PCA defense using the representations at the intermediate layers of neural networks trained on corrupted datasets. It is based on the observation that poisoned examples have special *spectral signatures* that can be used to filter them out. Concretely, given the intermediate representations $\{\boldsymbol{h}_i\}_{i=1}^n$ of the training data, each sample is assigned an outlier score $\tau_i = |\langle \boldsymbol{h}_i, \boldsymbol{v}_h \rangle|$, which is its magnitude in the top PCA direction $\boldsymbol{v}_h$ of the representations. Those with high scores

[1]Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, United States. Correspondence to: Jonathan Hayase <jhayase@cs.washington.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

[2]Code and pre-trained models are available at https://github.com/SewoongLab/spectre-defense.

are removed from the training data, and a fresh model is trained on the filtered data.

When there is a sufficient number of poisoned data (≥ 512) this PCA defense correctly detects poisoned samples and removes the backdoor completely; attack accuracy drops down to 0% when we retrain a model after removing samples detected as poisoned (shown in red circles). However, there is a wide gap between where the pixel attack becomes ineffective (around 64 poisoned examples) and where the PCA defense stops working (around 256 poisoned samples), in this example.

Contributions. We introduce SPECTRE (Spectral Poison ExCision Through Robust Estimation), a novel defense for general backdoor attacks. The key insight, illustrated in Fig. 2, is that we can significantly amplify the spectral signature of the poisoned data by (i) estimating the mean and the covariance of the clean data using robust statistical estimators and (ii) whitening the combined data with the estimated statistics. The resulting top PCA directions are well-aligned with the subspace that separates the poisoned samples from the clean ones (illustrated in Fig. 2). However, detecting those poisoned examples can still be challenging as the distribution of the (whitened) representations can vary widely depending on the types and strengths of the attacks. To adapt to such profile of the representations, we propose a variation of recently introduced QUantum Entropy (QUE) outlier scoring. We show in Section 4 that SPECTRE is able to eliminating the backdoor (e.g., shown in green squares in Fig. 1) under a broad range of attacks, significantly improving upon the state-of-the-art baselines. We show that every component of SPECTRE is crucial in achieving this performance gain with ablation study in Appendix C.

1.1. Related work

We focus on training-time attacks and refer readers to (Madry et al., 2017; Ilyas et al., 2019) for survey on inference-time attacks.

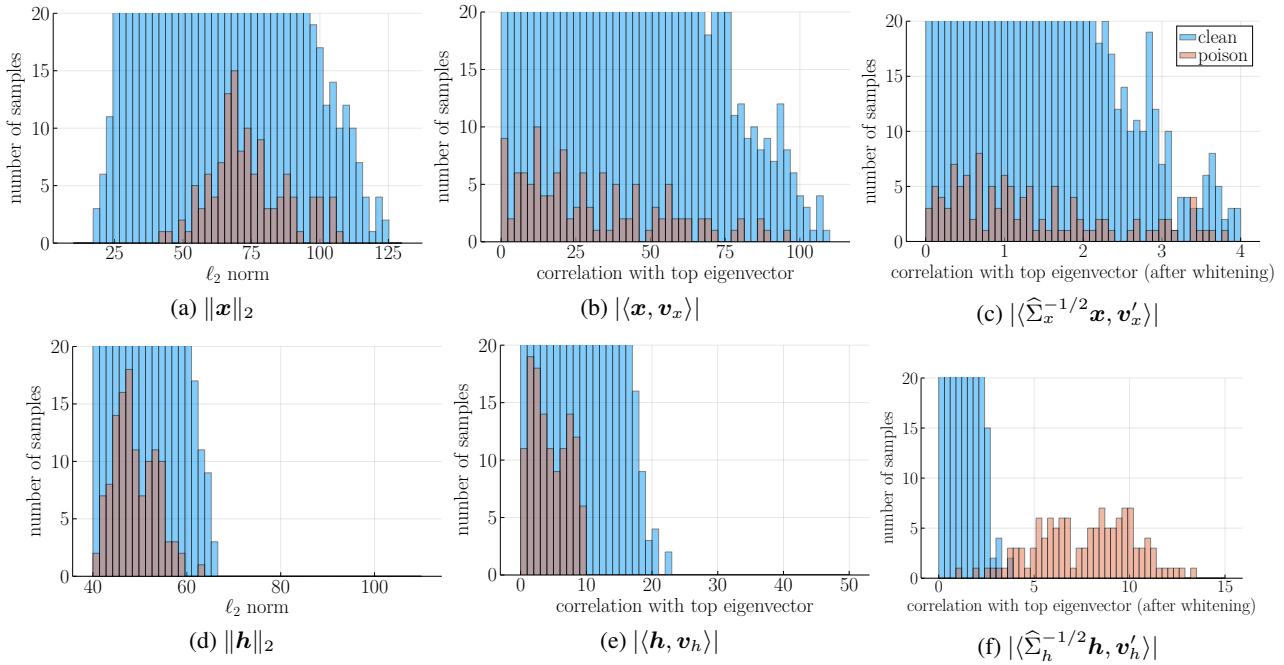
Data poisoning attacks and defenses. Data poisoning refers to attacks that insert poisoned examples into the training data. There are two types depending on the goal: reducing model quality or creating a backdoor. Model quality attacks have been studied in feature selection (Xiao et al., 2015), PCA (Rubinstein et al., 2009), neural networks (Yang et al., 2017), general models (Mozaffari-Kermani et al., 2014), and general function classes (Kearns & Li, 1993). These attacks have been successfully launched in deployed systems, as shown in (Newsome et al., 2006; Laskov, 2014; Biggio et al., 2014; Wang et al., 2020b).

Backdoor attacks. Backdoor attacks create backdoors in trained models that change the model's prediction to an attacker-specific target label, when the sample has a specific attacker-chosen trigger. The most common attack is to embed triggers in a subset of training samples from a source label and change the label to the target label. (Gu et al., 2017) first demonstrated that stamping an image with a small pattern can successfully create a backdoor. To design *triggers* that can pass a human inspection on the image $\boldsymbol{x}$, subsequent work mixed a pattern with the features (Chen et al., 2017), used periodic patterns to exploit convolutional layers (Zhong et al., 2020), used intermediate layers of a neural network (Liu et al., 2017), minimized ℓ_2 norm of the perturbation (Zhong et al., 2020), used perceptual similarity scores (Li et al., 2019), applied reflection to the image as the trigger (Liu et al., 2020), and leveraged downscaling pre-processing step common in image classification tasks (Quiring & Rieck, 2020). However, these approaches share a weakness that a human inspecting both the image $\boldsymbol{x}$ and the label y can easily detect a poisoned example, as it is perceived to be mislabelled as target y. Turner et al. (2019) and Zhao et al. (2020) propose embedding triggers in images that interpolate between the source and target labels. This can pass as being correctly labelled with the target label, while successfully creating backdoors. Saha et al. (2020) assumes a transfer learning scenario where a pretrained network is fine-tuned on a corrupted dataset and the attacker designs poisoned examples that can pass human inspection using the pretrained network. Shokri et al. (2020) proposed a backdoor attack designed to evade the defenses of (Tran et al., 2018) and (Chen et al., 2018a). However, this attack requires the attacker to control the training process of the network and does not fall within our threat model.

Defenses against backdoor attacks. As the defender is not assumed to have clean validation data, several approaches do not apply to our setting. Defenses using outlier detection require clean validation data (Liang et al., 2017; Lee et al., 2018; Steinhardt et al., 2017; Turner et al., 2019). (Liu et al., 2018) requires clean data to retrain a poisoned model to make it forget the backdoor. (Kolouri et al., 2020) requires a model trained on clean data to design a litmus test that detects poisoned models.

Some other defenses (Wang et al., 2019; Awasthi et al., 2020; Wang et al., 2020a; Weber et al., 2020; Chou et al., 2018) rely on triggers having a small norm, and are known to fail on attacks with large perturbations. Neural Cleanse (Wang et al., 2019) finds perturbations that change the label of a training sample. The smallest such perturbation is declared as the trigger. Randomized smoothing proposed in (Wang et al., 2020a; Weber et al., 2020) ensures that all bounded perturbations are consistently labelled, forcing clean image and its poisoned version to have the same label.

SentiNet (Chou et al., 2018) uses saliency maps to detect triggers corresponding to small connected regions of high



(a) $\|\boldsymbol{x}\|_2$ (b) $|\langle \boldsymbol{x}, \boldsymbol{v}_x \rangle|$ (c) $|\langle \widehat{\Sigma}_x^{-1/2} \boldsymbol{x}, \boldsymbol{v}'_x \rangle|$

(d) $\|\boldsymbol{h}\|_2$ (e) $|\langle \boldsymbol{h}, \boldsymbol{v}_h \rangle|$ (f) $|\langle \widehat{\Sigma}_h^{-1/2} \boldsymbol{h}, \boldsymbol{v}'_h \rangle|$

Figure 2: Plots of the 5,000 clean training examples and 125 poisoned examples bearing the target label under the 3-way pixel attack. Figs. 2a and 2d show the ℓ_2 norm of the images and representations respectively. Figs. 2b and 2e show the absolute inner product of the images and representations respectively with the top eigenvectors $\boldsymbol{v}_x$ and $\boldsymbol{v}_h$ of their covariances. Figs. 2c and 2f show the absolute inner product of the *robustly whitened* images and representations respectively with the top eigenvectors $\boldsymbol{v}'_x$ and $\boldsymbol{v}'_h$ of the covariances of the whitened data. Fig. 2f shows how robust whitening amplifies the spectral signature of the poisoned samples and separates them out along the direction of top principal components.

salience over multiple images. Other types of defenses protect against model quality attacks, including outlier detection without clean data (Sun et al., 2019; Steinhardt et al., 2017; Blanchard et al., 2017; Pillutla et al., 2019) and Byzantine-tolerant distributed learning approaches (Blanchard et al., 2017; Alistarh et al., 2018; Chen et al., 2018b).

Robust estimation. There has been significant progress in robust mean and covariance estimation under Gaussian samples in $\mathbb{R}^d$. (Chen et al., 2018c) gives the first exponential time algorithm that accurately estimates the covariance matrix with $\Omega(d)$ sample complexity under adversarial corruptions and prove a matching information theoretical lower bound. (Diakonikolas et al., 2019; Lai et al., 2016) give the first polynomial time algorithm with no (or very weak) dependency on the dimensionality in the estimation error (close to the one in (Chen et al., 2018c)), however at the cost of $\Omega(d^2)$ sample complexity. A *statistical query* (SQ) lower bound is later shown in (Diakonikolas et al., 2017b), indicating that a polynomial time algorithm with $\Omega(d^{1.99})$ sample complexity is unlikely. Recent work (Cheng et al., 2019; Li & Ye, 2020) improve the time complexity to match the matrix multiplication time, which nearly matches the time needed for the non-robust version of the problem.

2. Threat model and diversifying the attacks

2.1. Threat model

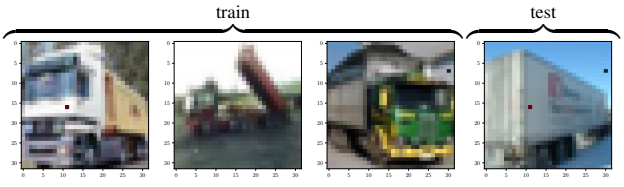
We assume the threat model of (Tran et al., 2018). The adversary has the training data and knows the user's neural architecture and training method. However, the adversary does not train the model herself. The user trains the model on training data that might be corrupted by the adversary, whose goal is to create a *backdoor* in the user's trained model. The goals of a backdoor are twofold: First, in order to avoid suspicion, the classification accuracy on the clean training data and clean test data should not decrease due to the presence of poisoned data (hence the name backdoor). Second, when a clean test data (whose label is not the target label) is corrupted by an attacker-defined trigger, the backdoor should be activated and the example should be classified as the attacker-defined target label.

To create a backdoor, the adversary injects poisoned data in the training set. We test our defense against the pixel attack, periodic attack, and clean label attack. We vary the fraction of injected poisoned examples denoted by

$$\varepsilon \triangleq \frac{\text{\# of poisoned examples injected}}{\text{\# of uncorrupted examples with target label}}.$$

2.2. Pixel attacks and m-way pixel attacks

One of the first successful demonstrations of a backdoor attack used a simple pixel attack (Gu et al., 2017). An image is corrupted by a single pixel at a fixed location set to a fixed color. At training, images from a label different from the target (e.g., "truck") are corrupted and injected to the dataset labelled as the target, e.g., "deer". On the CIFAR-10 dataset, each label has $5{,}000$ clean examples. The pixel attack only requires as few as 250 poisoned examples ($\varepsilon = 5\%$) to succeed in achieving 92% test accuracy on clean data and 89% test accuracy on poisoned data (see Fig. 1).



(a) 1st pixel (b) 2nd pixel (c) 3rd pixel (d) all three

Figure 3: During training, the m-way pixel attack partitions the data and applies a group-specific pixel attack to each. At test time, all m pixels are applied to strengthen the trigger.

A downside of the pixel attack is that it leaves strong spectral signatures, such that it can be easily detected by the PCA defense of (Tran et al., 2018), which successfully removes 94% of poisoned data when $\varepsilon = 10\%$. However, as PCA defense relies on a *single* principal direction, an *m-way attack* introduced in (Xie et al., 2019) diversifies the watermark such that the spectral signature is hidden in the lower PCA subspaces. The corrupted training data is separated into m partitions and a group specific pixel attack is applied to each group. At $\varepsilon = 10\%$, most of the poisoned samples under 2-way pixel attack can evade detection by PCA defense, as shown in Table 6 in the appendix, while maintaining the poison accuracy of 91%. We compare the state-of-the-art defenses on various attacks and their m-way variations.

3. Algorithm

The pipeline of our approach is to train a model and extract representations from a hidden layer, then identify the target label with Algorithm 4, detect and remove the poisoned examples with Algorithm 1, and retrain (see Fig. 4). In this section, we assume that the representations have been extracted and the target label has been correctly identified and focus on the robust poison detector. We refer to Section 4.5 for the details on identifying the target label.

We propose the following three steps in SPECTRE (Algorithm 1). We first project the given representation data down to a k-dimensional space using its top k left singular vectors. We next apply robust estimation to get the approximate

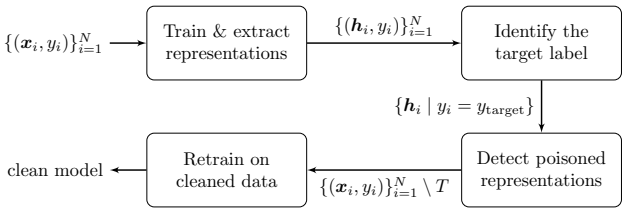


Figure 4: The defense pipeline. We first train a model on the poisoned data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ and extract the activation $\boldsymbol{h}_i \in \mathbb{R}^d$ of a hidden layer of the trained neural network as the representation of the data $\boldsymbol{x}_i$. Then, this representation is used in Algorithm 4 to identify the target label. Algorithm 1 uses the representations $\{\boldsymbol{h}_i \mid y_i = y_{\text{target}}\}$ of the target label to detect and remove suspicious examples T. Finally, we retrain a model with the cleaned data.

mean and covariance of the clean data. After whitening the data with the estimated mean and covariance, the spectral signature of the poisoned data is amplified such that it can be detected more effectively. Finally, we use QUantum Entropy (QUE) scores to find those with strong spectral signatures. Note that the sensitivity of the algorithm is tuned by the choice of removing $1.5\varepsilon n$ suspicious samples, following the same choice from (Tran et al., 2018). We show that the performance is not sensitive to this choice in Appendix H.

Algorithm 1: SPECTRE

Input: representation $S = \{\boldsymbol{h}_i \in \mathbb{R}^d\}_{i=1}^n$, dimension k, parameter α, poison fraction ε
$\boldsymbol{\mu}(S) \leftarrow \frac{1}{n}\sum_{i=1}^n \boldsymbol{h}_i$
Center the data: $S_1 \leftarrow \{\boldsymbol{h}_i - \boldsymbol{\mu}(S)\}_{\boldsymbol{h}_i \in S}$
$U, \Lambda, V \leftarrow \text{SVD}_k(S_1)$
$T_1 \leftarrow \{U^\top \boldsymbol{h}_i\}_{\boldsymbol{h}_i \in S}$
$\widehat{\Sigma}, \widehat{\boldsymbol{\mu}} \leftarrow \text{RobustEst}(T_1, \varepsilon)$ [Algorithm 13]
Whiten the data: $T_2 \leftarrow \{\widehat{\Sigma}^{-1/2}(\bar{\boldsymbol{h}}_i - \widehat{\boldsymbol{\mu}})\}_{\bar{\boldsymbol{h}}_i \in T_1}$
$\{\tau_i\} \leftarrow \text{QUEScore}(T_2, \alpha)$ [defined in Eq. (1)]
return $1.5\varepsilon n$ samples with greatest QUE-scores

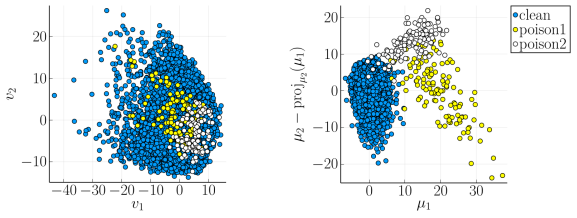
3.1. Step 1: Dimensionality reduction with SVD

A robust estimation of the mean and covariance in d-dimensions with ε fraction of poisoned data requires $\Omega(d^2/\varepsilon^2)$ samples, which we do not have in real data. On CIFAR-10 experiments, the representations are $4{,}096$ dimensional and the number of samples per label is $5{,}000$. We propose projecting the data down to a k-dimensional space using the top left singular vectors $U \in \mathbb{R}^{d \times k}$. With a choice of k that is too small, the subspace U might not include the direction separating the poisons, thus losing statistical power for detection. If we choose a k that is too large then the subspace U might contain directions where the clean data is not well-behaved and follows a heavy-tailed

distribution, thus misleading the robust covariance estimation due to the small sample size. Hence, we propose an algorithm to find an effective dimension k in Algorithm 3 and use it in all our experiments. This achieves a performance close to the best performance one can achieve by enumerating all k as we show in Section 4.4.

3.2. Step 2: Robust estimation

The PCA defense fails when the direction the algorithm checks (which is the top PCA direction of the combined data) is not aligned with the spectral signature of the poisoned examples (which is the direction that separates poisoned from clean data). This happens when the covariance of the clean data has a large condition number such that the variance along the spectral signature direction is much smaller than the variance along the top PCA direction, as shown in Figs. 2 and 5. In real data, the spectral signature often hides in these low-variance directions, causing PCA Defense to fail under most of the attacks we tested.



(a) 2-PCA (b) Poison mean subspace

Figure 5: When we project onto the top PCA directions of the representations $\{\boldsymbol{h}_i\}$ of the combined data on the left, the poisoned examples are indistinguishable from the clean ones. On the two-dimensional subspace that best separates the poisons (right), on the other hand, the representations have smaller variance, making those directions challenging to find. This example uses the 2-way pixel attack with $\varepsilon n = 250$.

If we know the true mean and covariance of the clean data, we can whiten the combined data to ensure that the clean data has the same variance along the spectral signature direction as any other directions, thus amplifying the hidden spectral signature. We propose using the recently introduced robust mean and covariance estimators of (Diakonikolas et al., 2017a), which are guaranteed to accurately estimate the true mean and covariance when we have enough samples from a Gaussian distribution.

Theorem 1 ((Diakonikolas et al., 2017a, Theorem 3.2 and Theorem 3.3)). *Let $G \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ be a Gaussian in d dimensions, and let $\varepsilon > 0$. Let S be an ε-corrupted set of samples from G of size $\Omega((d^2/\varepsilon^2)\operatorname{poly}\log(d/\varepsilon))$.* ROBUSTEST*$(S, \varepsilon)$, returns $\widehat{\Sigma}$ and $\widehat{\boldsymbol{\mu}}$, so that with probability at least $9/10$, it holds that $\|\boldsymbol{I} - \Sigma^{-1/2}\widehat{\Sigma}\Sigma^{-1/2}\|_F = O(\varepsilon \log(1/\varepsilon))$ and $\|\boldsymbol{\mu}' - \boldsymbol{\mu}\|_2 = O(\varepsilon\sqrt{\log(1/\varepsilon)})$.*

Under the assumption that the clean data is drawn from a Gaussian distribution, this provides the best known guarantee for joint mean and covariance estimation and also matches the known fundamental limit on the achievable accuracy up to a logarithmic factor. However, in practice, we do not have enough samples to do robust estimation of the $d = 4{,}096$ dimensional covariance in real data with CIFAR-10, where each label has $5{,}000$ samples. Therefore, it is critical to use an appropriate choice of k in reducing the dimensionality of the samples down to k in the pre-processing. In fact, a moderate choice of $k = 60$ can completely fail as we illustrate in Fig. 8. To this end, we propose Algorithm 3 to identify the dimensionality k. For completeness we also provide ROBUSTEST from (Diakonikolas et al., 2017a) in Algorithm 13 in Appendix D.

3.3. Step 3: Quantum entropy score poison detection

We want to assign an *outlier score* τ_i to each data point and remove those with high scores. Once we whiten and center the representation according to the approximate mean and covariance of the clean data (denoted by $\{\tilde{\boldsymbol{h}}_i \in \mathbb{R}^k\}$), the poisoned samples tend to be separated from the clean ones and are left with a *spectral signature*. Natural measures of this signature are the *squared norm* $\tau_i^{(0)} = (1/k)\|\tilde{\boldsymbol{h}}_i\|_2^2$ and the *squared projected norm* $\tau_i^{(\infty)} = (\boldsymbol{v}^\top \tilde{\boldsymbol{h}}_i)^2$ on the top principal direction $\boldsymbol{v}$ of the whitened representation $\{\tilde{\boldsymbol{h}}_i\}_{i=1}^n$ including both clean and poisoned samples. In practice, either choice can fail as shown in Table 1. To this end, we propose using a variation of QUantum Entropy (QUE) scoring from (Dong et al., 2019).

Algorithm 2: QUantum Entropy scoring (QUESCORE) based on (Dong et al., 2019, Algorithm 2)

Input: $T = \{\tilde{\boldsymbol{h}}_i \in \mathbb{R}^k\}_{i=1}^n$, parameter α

$$\tau_i^{(\alpha)} \leftarrow \frac{\tilde{\boldsymbol{h}}_i^\top Q_\alpha \tilde{\boldsymbol{h}}_i}{\operatorname{Tr}(Q_\alpha)}\ ,\ \forall i \in [n] \qquad (1)$$

where $Q_\alpha = \exp\left(\frac{\alpha(\tilde{\Sigma}-\boldsymbol{I})}{\|\tilde{\Sigma}\|_2 - 1}\right)$ and $\tilde{\Sigma} = \frac{1}{n}\sum_{i=1}^n \tilde{\boldsymbol{h}}_i \tilde{\boldsymbol{h}}_i^\top$

return $\{\tau_i^{(\alpha)}\}$

note on α: we use $\alpha = 4$ in all experiments

QUE score defined in (1) recovers $\tau_i^{(0)} = (1/k)\|\tilde{\boldsymbol{h}}_i\|^2$when $\alpha = 0$ and recovers $\tau_i^{(\infty)} = (v^\top \tilde{\boldsymbol{h}}_i)^2$ when $\alpha = \infty$. For intermediate α, this gracefully interpolates between these extremes, thus improving over both as shown below.

The name quantum entropy scoring comes from the fact that the matrix exponential $Q_\alpha / \operatorname{Tr}(Q_\alpha)$ is a solution of a particular linear maximization with a quantum entropy regularization. This matrix weighs the top and bottom principal directions differently, and the choice of α controls how ag-

Attacks \ Scores	$\tau_i^{(0)}$	$\tau_i^{(2)}$	$\tau_i^{(4)}$	$\tau_i^{(8)}$	$\tau_i^{(\infty)}$
1-way $\varepsilon = 0.1$	3	0	0	6	118
2-way $\varepsilon = 0.05$	69	40	30	49	97
3-way $\varepsilon = 0.0124$	22	8	5	5	5

Table 1: Number of remaining poisoned samples after removing $1.5\varepsilon n$ examples with largest outlier scores $\tau_i^{(\alpha)}$ for various choices of $\alpha \in \{0, 2, 4, 8, \infty\}$. The proposed QUE score robustly achieves the best performance with $\alpha = 4$.

gressively we want to emphasize the top principal directions. This allows the QUE score to naturally adapt to the effective dimensionality of the spectral signature in poisoned samples. The squared norm $\tau_i^{(0)}$ fails when this effective dimension is small, which happens when the signature is weak, i.e. large m and small ε. The squared projected norm $\tau_i^{(\infty)}$ fails when the effective dimension is large, which happens when the signature is strong, i.e., small m and large ε. The experiments support this intuition and we provide details in Appendix G. The performance of the score is not sensitive to the choice of α and we set it to 4 for all our experiments. QUE score plays critical roles also in identifying the target label (Algorithm 4) and also in selecting the dimensionality k (Algorithm 3).

3.4. Possible extensions to SPECTRE

In the dimensionality reduction step, we could have used robust *principal component analysis* (Kong et al., 2020; Jambulapati et al., 2020) to replace U with the estimated principal subspace of the clean data. Further, theoretically, we should partition the data into two groups $S_1 \cup S_2 = S$, and project the data from one group onto the subspace learned from the SVD of the other group. This ensures that the learned subspace does not overfit the data. In practice, these two variations did not give any improvement in performance.

4. Experiments

In our pipeline (Fig. 4) for removing poison and retraining, we replace our proposed SPECTRE with two competing state-of-the-art approaches and compare the resulting performances. Following (Tran et al., 2018), in all experiments, we set the sensitivity so that $1.5\varepsilon n$ data points are removed in total, we use "deer" as the target label, and use images of trucks to create poisoned samples (unless otherwise stated). In all experiments shown in this section, we use Algorithm 3 (explained in Section 4.4) to find the effective dimension k adaptively, and use Algorithm 4 (explained in Section 4.5) to identify the target label. Due to space constraints, we only report the attack accuracy on the backdoored test examples on the final re-trained model. The accuracy on the clean test examples is always between 92.5% and 93.5% unless otherwise stated, and is omitted from the results. Complete statistics of the poison removal process are provided in Appendix B.

We compare three defenses: the proposed Algorithm 1, the PCA defense of (Tran et al., 2018) and the Clustering defense of (Chen et al., 2018a). The Clustering defense uses standard 2-means on the representations and we allow access to the oracle to determine one cluster and randomly select $1.5\varepsilon n$ data points to remove from that cluster. Detailed descriptions are provided in Appendix A. We evaluate them on three popular families of backdoor attacks. Supplementary experimental results are given in Appendix B, including a section on defending against the hidden trigger attack of (Saha et al., 2020) in Appendix B.2.

4.1. m-way pixel attacks

We test the defenses on the m-way pixel attacks described in Section 2.2 with examples shown in Fig. 3. Following the experiments of (Tran et al., 2018), we use a poisoned CIFAR-10 dataset to train a 32-layer ResNet[3] model composed of three groups of residual blocks with 16, 32, and 64 filters respectively and 5 residual blocks per group. Details of the training are provided in Appendix E. A complete table of all the results including the number of poisoned training examples detected by each defense is provided in Table 6.

Attack			PCA	Clustering	SPECTRE
m	εn	acc_{p^*}	acc'_{p^*}	acc'_{p^*}	acc'_{p^*}
1	500	0.942	0.004	0.820	**0.000**
1	250	0.890	0.880	0.904	**0.001**
1	125	0.627	0.834	0.842	**0.000**
2	500	0.987	0.914	0.901	**0.000**
2	250	0.888	0.817	0.808	**0.002**
2	125	0.106	0.139	0.325	**0.000**
3	500	0.990	0.970	0.963	**0.000**
3	250	0.908	0.367	0.914	**0.000**
3	125	0.616	0.348	0.547	**0.000**

Table 2: m-way pixel attack test accuracy acc'_{p^*} on the backdoor examples of the model retrained with each defense. SPECTRE consistently eliminates the backdoor completely (achieving the poison accuracy near zero), in all regimes including those where existing methods fail. The attack accuracy acc_{p^*} on a model trained without any defense is shown as a reference.

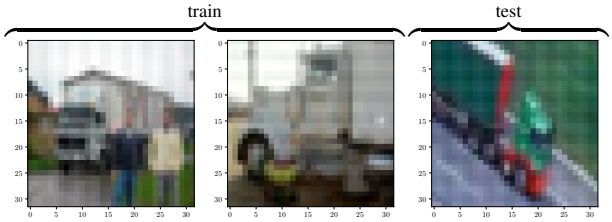
The PCA defense succeeds when the spectral signature is

[3] We modified the implementation at https://github.com/akamaster/pytorch_resnet_cifar10 to match that used in (Tran et al., 2018).

strong ($m = 1, \varepsilon = 500$) but fails when we diversify the attack, keeping the same number of poisons or reducing the number of poisons, because the spectral signature is weaker. Robust covariance estimation consistently amplifies these signatures, eliminating the backdoor in all cases. The clustering defense fails to separate poisons from clean ones.

4.2. m-way periodic attacks

Proposed in (Barni et al., 2019), the periodic attack adds a periodic signal to the image as a trigger, as shown in Fig. 6. We chose signals with amplitude 6 and frequency of 8. We design an m-way periodic attack by choosing m different (frequency, direction) pairs. Table 7 in the appendix provides all the experimental results. The same experimental setting was used as in Section 4.1. Algorithm 1 consistently removes the backdoors completely, whereas competing defenses fail.



(a) horizontal (b) vertical (c) both

Figure 6: At training, each poisoned sample is corrupted by a single periodic signal to better hide the spectral signature. At test time, we combine all m triggers to boost the spectral signature and improve the accuracy of the attack.

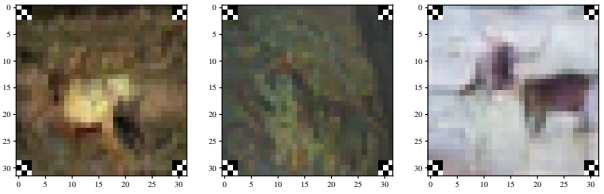
Attack m	εn	acc_{p^*}	PCA acc'_{p^*}	Clustering acc'_{p^*}	SPECTRE acc'_{p^*}
1	500	0.975	0.976	0.987	**0.004**
1	250	0.961	0.968	0.933	**0.001**
1	125	0.912	0.916	0.889	**0.000**
2	500	0.996	0.995	0.988	**0.001**
2	250	0.982	0.986	0.961	**0.000**
2	125	0.881	0.868	0.829	**0.000**

Table 3: m-way periodic attack results with notations from Table 2. The attack accuracy of SPECTRE shows that the backdoor has been completely eliminated.

4.3. Label consistent attacks

The obvious discrepancy between the image and the target label (e.g., a truck labelled as a deer) in previously presented attacks makes it trivial for a human to detect the poison. The label consistent attack, which was proposed in (Turner et al., 2019), designs images that are consistent with the target label, but can still create backdoors.

Concretely, three transforms are proposed to create images of the target label which are more difficult to classify: ℓ_2 and ℓ_∞ bounded adversarial perturbations and interpolation via the latent space of a Generative Adversarial Network (GAN). A watermark, which in our case is a 3x3 patch of black and white pixels on each corner, is then added to the transformed images, During training, the network may come to rely on the watermark to classify the poisoned examples, as classifying them without the watermark is difficult. At test time, the network outputs the target label whenever it detects the watermark. Examples are shown in Fig. 7.



(a) ℓ_2 perturbation (b) ℓ_∞ perturbation (c) GAN interpolation

Figure 7: Examples training samples for label consistent attacks, which are visually consistent with the target label "deer", while succeeding in creating backdoors that are triggered by the watermark in the corners.

We used the same experimental setup as (Turner et al., 2019). For our experiments, we ran the provided implementation.[4] Accuracy on clean data was between 91% and 92.5% in all experiments and are omitted in the table. More results are provided in Table 8 in the appendix.

Attack type	εn	acc_p	PCA p_{rm}	Clustering p_{rm}	SPECTRE p_{rm}
ℓ_2	250	0.932	**250**	140	**250**
ℓ_2	125	0.843	1	17	**125**
ℓ_2	62	0.856	0	5	**62**
ℓ_∞	250	0.894	**250**	245	**250**
ℓ_∞	125	0.744	0	24	**125**
ℓ_∞	62	0.472	0	5	**62**
GAN	250	0.584	47	78	**250**
GAN	125	0.680	28	20	**125**
GAN	62	0.261	0	2	**62**

Table 4: Under label consistent attacks each defense detects $1.5\varepsilon n$ candidates to remove, out of which p_{rm} are actual poisoned examples. This matches the total number of poisoned examples εn for SPECTRE.

Algorithm 1 removed all poisoned examples in every instance, guaranteeing that the backdoor was eliminated. How-

[4] https://github.com/MadryLab/label-consistent-backdoor-code

ever, in a wide regime, the PCA and Clustering defenses removed a small fraction of the poison or none at all.

4.4. Finding the effective dimension k

Algorithm 1 takes a parameter k, which is the number of dimensions to use for covariance estimation. Comparing Figs. 8 and 9, note that no fixed value of k works well for all experiments. A small choice of k fails when the spectral signature is not in the top k PCA directions, which happens when the attack is weak (Fig. 9). A large choice of k fails when the clean data is not well-behaved (resilience property fails) in the lower PCA subspaces causing robust covariance estimation to fail (Fig. 8).

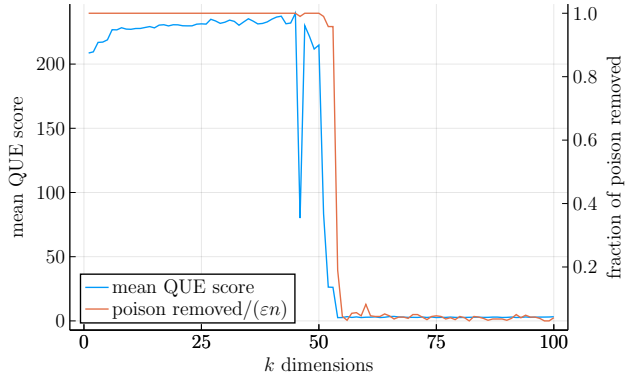


Figure 8: Under the GAN-based label consistent attack with $\varepsilon n = 500$, we want to choose $k \leq 55$ as we want the fraction of poisons removed by SPECTRE (in red) close to one. We propose selecting k with the highest mean QUE score (in blue), as it closely matches the true (unknown) detection accuracy.

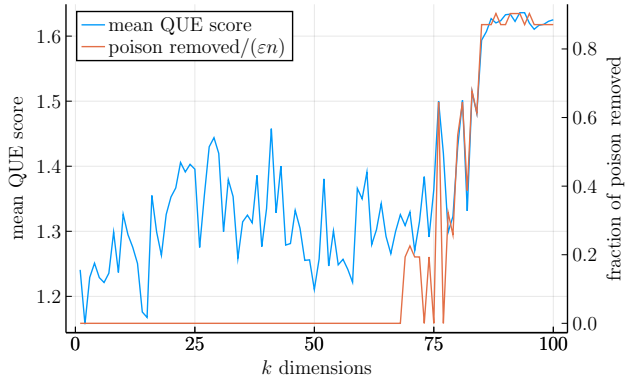


Figure 9: Under the 2-way pixel attack with $\varepsilon n = 31$, we want to select $k \geq 85$. We propose selecting k with the highest mean QUE score.

A major challenge in selecting the appropriate k is that we do not have oracle access to the performance of our SPECTRE (in red), as in practice we do not know which samples are poisoned. We therefore propose selecting k that maximizes the mean QUE score (in blue). Concretely, for each k we run SPECTRE to remove $1.5\varepsilon n$ data points. We use the covariance of the remaining cleaned examples (in the representation space) to whiten all the data, and compute the mean QUE score of all the data points after whitening. The idea is that if poisons were correctly identified, then the mean QUE score will be large as poisons have strong spectral signature. We write the algorithm explicitly in Algorithm 3. Table 5 shows that Algorithm 3 selects nearly optimal values of k.

Algorithm 3: k-IDENTIFIER

Input: representation $S = \{\boldsymbol{h}_i \in \mathbb{R}^d\}_{i=1}^n$, parameter α, poison fraction ε

$\boldsymbol{\mu}(S) \leftarrow \frac{1}{n}\sum_{i=1}^n \boldsymbol{h}_i$

Center the data: $S_1 \leftarrow \{\boldsymbol{h}_i - \boldsymbol{\mu}(S)\}_{\boldsymbol{h}_i \in S}$

$U, \Lambda, V \leftarrow \mathrm{SVD}_{k_{\max}}(S_1)$

for $k \in [k_{\max}]$ **do**

  $S_{\text{removed}} \leftarrow \mathrm{SPECTRE}(S, k, \alpha, \varepsilon)$ [Algorithm 1]

  $\Sigma' = \mathrm{Cov}(\{U^\top \boldsymbol{h} \mid \boldsymbol{h} \in S \setminus S_{\text{removed}}\})$

  $\{\tau_i\} \leftarrow \mathrm{QUESCORE}(\{\Sigma'^{-1/2} U^\top \boldsymbol{h} \mid \boldsymbol{h} \in S\})$ [Algorithm 2]

  $q \leftarrow \frac{1}{n}\sum_{i=1}^n \tau_i$

return k corresponding to the maximum q and the the maximum q

metric / choice of k	20	100	k_{oracle}	Alg. 3
mean $\mathrm{p}_{\text{rm}}/(\varepsilon n)$ (%)	76.5	86.8	98.6	98.2
min $\mathrm{p}_{\text{rm}}/(\varepsilon n)$ (%)	0.0	4.0	90.3	87.1

Table 5: Fixed choices of k results in failure in some examples, as shown by low min % of poisons removed ($\mathrm{p}_{\text{rm}}/(\varepsilon n)$). The minimum is over different attacks Algorithm 3 achieves a consistently reliable performance, close to the instance-wise optimal choice of k_{oracle}.

4.5. Identifying the target label

All three defenses require representations from the target label, which is not known. To identify which label is being targeted, we extend Algorithm 3, which identifies the effective dimension k, to identify both k and the target label l, giving Algorithm 4. Figs. 10 and 11 shows that the mean QUE scores obtained for the target label are clearly larger (for appropriate values of effective dimension k) than those obtained for untargeted labels. This follows from the same intuition as Algorithm 3, where higher mean QUE score indicates the presence of poisoned data samples. We run Algorithm 4 against all attacks with poison test accuracy $\mathrm{acc}_{\mathrm{p}}$ over 0.33; the correct target label was identified in all those experiments with 100% accuracy.

Algorithm 4: Target label identifier

Input: representations $S_l = \{\boldsymbol{h}_i \in \mathbb{R}^d\}_{i=1}^{n_l}$ for each label $l \in [L]$, parameter α, poison fraction ε

for $l \in [L]$ **do**

  $k, q \leftarrow k$-IDENTIFIER(S_l, ε) [Algorithm 3]

return l corresponding to the maximum q.

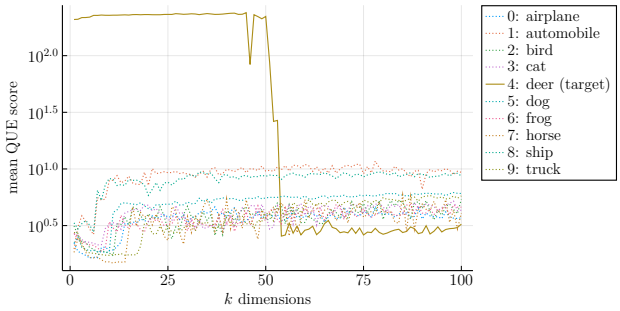


Figure 10: GAN-based label consistent attack with $\varepsilon n = 500$. We select (k, label) pair that maximizes the mean QUE score.

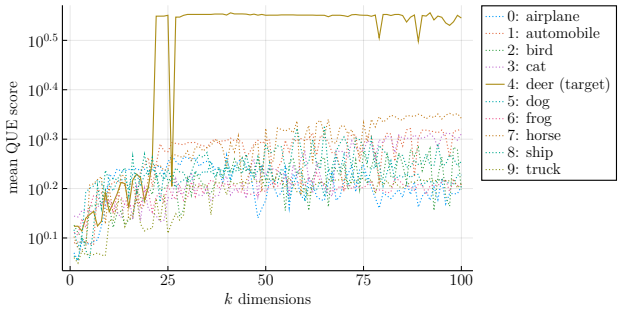


Figure 11: 3-way pixel attack with $\varepsilon n = 125$. We select (k, label) pair that maximizes the mean QUE score.

5. Conclusion

While existing backdoor attacks are powerful enough to corrupt the trained model with a small fraction of injected poisoned training data, existing defenses fail under a broad regime of backdoor attacks. The reason is that the spectral signatures that those methods build upon are challenging to detect for a wide range of attacks. We therefore introduce a novel defense algorithm, that we call SPECTRE, by combining the ideas from robust covariance estimation and quantum entropy outlier detection. Whitening with the robust covariance amplifies the spectral signature of the poisoned samples. The quantum entropy score can robustly detect that signature, adapting to the spectral profile of the poisoned examples. We demonstrate the superiority of our defense using several popular backdoor attacks, which suggest that the proposed defense is successful in all regimes we tested on, including those where the state-of-the-art baseline approaches fail. The empirical success of SPECTRE opens several new research directions, two of which we discuss in the following.

SPECTRE requires the trainer to have access to the corrupted training dataset. In some scenarios we might not have a direct access to the training data, for example due to privacy constraints. Identifying the statistical signatures in such settings is an interesting direction to make SPECTRE more widely applicable. A concrete direction is to design a decentralized and differentially private version of SPECTRE under the setting of federated learning (Pillutla et al., 2019). Recent advances in differentially private and robust estimators in (Liu et al., 2021) provide promising directions.

(Gao et al., 2019) proposes a different paradigm for defending against backdoor attacks. The defense, called STRIP, mixes each training sample with multiple other samples and measure the entropy of the resulting prediction. This leverages an aspect of common backdoor attacks that is different from spectral signatures. Understanding how these different types of defenses perform against different types of attacks, such as the hidden backdoor attacks from (Saha et al., 2020), is an important research question.

Acknowledgement

This work is supported by Google faculty research award and NSF grants CNS-2002664, IIS-1929955, and CCF-2019844 as a part of Institute for Foundations of Machine Learning.

References

Alistarh, D., Allen-Zhu, Z., and Li, J. Byzantine stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pp. 4613–4623, 2018.

Awasthi, P., Jain, H., Rawat, A. S., and Vijayaraghavan, A. Adversarial robustness via robust low rank representations. *Advances in Neural Information Processing Systems*, 33, 2020.

Barni, M., Kallas, K., and Tondi, B. A new backdoor attack in cnns by training set corruption without label poisoning. In *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 101–105. IEEE, 2019.

Biggio, B., Rieck, K., Ariu, D., Wressnegger, C., Corona, I., Giacinto, G., and Roli, F. Poisoning behavioral malware clustering. In *Proceedings of the 2014 workshop on artificial intelligent and security workshop*, pp. 27–36, 2014.

Blanchard, P., Guerraoui, R., and Stainer, J. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, pp. 119–129, 2017.

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

Chen, L., Wang, H., Charles, Z., and Papailiopoulos, D. Draco: Byzantine-resilient distributed training via redundant gradients. *arXiv preprint arXiv:1803.09877*, 2018b.

Chen, M., Gao, C., Ren, Z., et al. Robust covariance and scatter matrix estimation under huber's contamination model. *Annals of Statistics*, 46(5):1932–1960, 2018c.

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

Cheng, Y., Diakonikolas, I., Ge, R., and Woodruff, D. P. Faster algorithms for high-dimensional robust covariance estimation. In *Conference on Learning Theory*, pp. 727–757. PMLR, 2019.

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

Diakonikolas, I., Kamath, G., Kane, D. M., Li, J., Moitra, A., and Stewart, A. Being robust (in high dimensions) can be practical. In *International Conference on Machine Learning*, pp. 999–1008. PMLR, 2017a.

Diakonikolas, I., Kane, D. M., and Stewart, A. Statistical query lower bounds for robust estimation of high-dimensional gaussians and gaussian mixtures. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 73–84. IEEE, 2017b.

Diakonikolas, I., Kamath, G., Kane, D., Li, J., Moitra, A., and Stewart, A. Robust estimators in high-dimensions without the computational intractability. *SIAM Journal on Computing*, 48(2):742–864, 2019.

Dong, Y., Hopkins, S. B., and Li, J. Quantum entropy scoring for fast robust mean estimation and improved outlier detection. *arXiv preprint arXiv:1906.11366*, 2019.

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

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

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, pp. 125–136, 2019.

Jambulapati, A., Li, J., and Tian, K. Robust sub-gaussian principal component analysis and width-independent schatten packing. *arXiv preprint arXiv:2006.06980*, 2020.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Kearns, M. and Li, M. Learning in the presence of malicious errors. *SIAM Journal on Computing*, 22(4):807–837, 1993.

Kolouri, S., Saha, A., Pirsiavash, H., and Hoffmann, H. Universal litmus patterns: Revealing backdoor attacks in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 301–310, 2020.

Kong, W., Somani, R., Kakade, S., and Oh, S. Robust meta-learning for mixed linear regression with small batches. *arXiv preprint arXiv:2006.09702*, 2020.

Lai, K. A., Rao, A. B., and Vempala, S. Agnostic estimation of mean and covariance. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 665–674. IEEE, 2016.

Laskov, P. Practical evasion of a learning-based classifier: A case study. In *2014 IEEE symposium on security and privacy*, pp. 197–211. IEEE, 2014.

Lee, K., Lee, K., Lee, H., and Shin, J. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, pp. 7167–7177, 2018.

Li, J. and Ye, G. Robust gaussian covariance estimation in nearly-matrix multiplication time. *arXiv preprint arXiv:2006.13312*, 2020.

Li, S., Xue, M., Zhao, B. Z. H., Zhu, H., and Zhang, X. Invisible backdoor attacks on deep neural networks via steganography and regularization. *arXiv preprint arXiv:1909.02742*, 2019.

Liang, S., Li, Y., and Srikant, R. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

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

Liu, X., Kong, W., Kakade, S., and Oh, S. Robust and differentially private mean estimation. *arXiv preprint arXiv:2102.09159*, 2021.

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

Liu, Y., Ma, X., Bailey, J., and Lu, F. Reflection backdoor: A natural backdoor attack on deep neural networks. In *European Conference on Computer Vision*, pp. 182–199. Springer, 2020.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Mozaffari-Kermani, M., Sur-Kolay, S., Raghunathan, A., and Jha, N. K. Systematic poisoning attacks on and defenses for machine learning in healthcare. *IEEE journal of biomedical and health informatics*, 19(6):1893–1905, 2014.

Newsome, J., Karp, B., and Song, D. Paragraph: Thwarting signature learning by training maliciously. In *International Workshop on Recent Advances in Intrusion Detection*, pp. 81–105. Springer, 2006.

Periša, L. *Recompression of Hadamard products of tensors in Tucker format*. PhD thesis, University of Zagreb. Faculty of Science. Department of Mathematics, 2017.

Pillutla, K., Kakade, S. M., and Harchaoui, Z. Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445*, 2019.

Quiring, E. and Rieck, K. Backdooring and poisoning neural networks with image-scaling attacks. *arXiv preprint arXiv:2003.08633*, 2020.

Rubinstein, B. I., Nelson, B., Huang, L., Joseph, A. D., Lau, S.-h., Rao, S., Taft, N., and Tygar, J. D. Antidote: understanding and defending against poisoning of anomaly detectors. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, pp. 1–14, 2009.

Saha, A., Subramanya, A., and Pirsiavash, H. Hidden trigger backdoor attacks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07): 11957–11965, Apr. 2020. doi: 10.1609/aaai.v34i07.6871. URL https://ojs.aaai.org/index.php/AAAI/article/view/6871.

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

Steinhardt, J., Koh, P. W. W., and Liang, P. S. Certified defenses for data poisoning attacks. In *Advances in neural information processing systems*, pp. 3517–3529, 2017.

Sun, Z., Kairouz, P., Suresh, A. T., and McMahan, H. B. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.

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

Turner, A., Tsipras, D., and Madry, A. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*, 2019.

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

Wang, B., Cao, X., and Gong, N. Z. On certifying robustness against backdoor attacks via randomized smoothing. *arXiv preprint arXiv:2002.11750*, 2020a.

Wang, H., Sreenivasan, K., Rajput, S., Vishwakarma, H., Agarwal, S., Sohn, J.-y., Lee, K., and Papailiopoulos, D. Attack of the tails: Yes, you really can backdoor federated learning. *Advances in Neural Information Processing Systems*, 33, 2020b.

Weber, M., Xu, X., Karlas, B., Zhang, C., and Li, B. Rab: Provable robustness against backdoor attacks. *arXiv preprint arXiv:2003.08904*, 2020.

Xiao, H., Biggio, B., Brown, G., Fumera, G., Eckert, C., and Roli, F. Is feature selection secure against training data poisoning? In *International Conference on Machine Learning*, pp. 1689–1698, 2015.

Xie, C., Huang, K., Chen, P.-Y., and Li, B. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2019.

Yang, C., Wu, Q., Li, H., and Chen, Y. Generative poisoning attack method against neural networks. *arXiv preprint arXiv:1703.01340*, 2017.

Zhao, S., Ma, X., Zheng, X., Bailey, J., Chen, J., and Jiang, Y.-G. Clean-label backdoor attacks on video recognition models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14443–14452, 2020.

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

Appendix

A. Previous approaches

For completeness, we write the algorithms we used for comparisons here.

A.1. Principal Component Defense

The principal component defense was proposed in (Tran et al., 2018). The representations produced by the network are analyzed by projecting them onto the top eigenvector of their covariance and then removing points that are far from the mean. This algorithm is shown in Algorithm 5.

Algorithm 5: PCA Defense (Tran et al., 2018)

Input: representation $S = \{\boldsymbol{h}_i \in \mathbb{R}^d\}_{i=1}^n$
$\boldsymbol{\mu}(S) \leftarrow \frac{1}{n}\sum_{i=1}^n \boldsymbol{h}_i$
Center the data: $S_1 \leftarrow \{\boldsymbol{h}_i - \boldsymbol{\mu}(S)\}_{\boldsymbol{h}_i \in S}$
$\boldsymbol{v}, \lambda, \boldsymbol{u} \leftarrow \mathrm{SVD}_1(S_1)$
return $1.5\varepsilon n$ samples with greatest $|\langle \boldsymbol{h}_i, \boldsymbol{v}\rangle|$

A.2. Clustering Defense

The clustering defense was proposed in (Chen et al., 2018a). The representations are analyzed by reducing their dimension using principal component analysis and running a clustering algorithm on the result. The exact algorithm is shown in Algorithm 6.

Algorithm 6: Activation Clustering (Chen et al., 2018a)

Input: representation $S = \{\boldsymbol{h}_i \in \mathbb{R}^d\}_{i=1}^n$, dimension k
$\boldsymbol{\mu}(S) \leftarrow \frac{1}{n}\sum_{i=1}^n \boldsymbol{h}_i$
Center the data: $S_1 \leftarrow \{\boldsymbol{h}_i - \boldsymbol{\mu}(S)\}_{\boldsymbol{h}_i \in S}$
$U, \Lambda, V \leftarrow \mathrm{SVD}_k(S_1)$
$C_1, C_2 \leftarrow 2\text{-means}(\{U^\top \boldsymbol{h} \mid \boldsymbol{h} \in S_1\})$
return clusters C_1, C_2

Chen et al. (2018a) propose several methods to determine which clusters, if any, contain poisoned representations. To avoid these complexities, we equip the algorithm with an oracle, ClusterOracle, which given two clusters returns the cluster with the greatest fraction of poisoned examples. The algorithm which returns the best cluster out of C_1, C_2 give by the oracle should perform at least as well as any heuristic to determine which clusters to return. There are two other concerns which make it difficult to compare this defense with Algorithm 1: first, there is no way to control how many examples are removed and second, the performance of the clustering varies with the initialization of k-means, which is random. Therefore, we use a second step which repeatedly runs Algorithm 6 and samples the cluster with the highest fraction of poison according to the oracle in order to build the set of samples to remove. The algorithm is shown in Algorithm 7.

Algorithm 7: Activation Clustering with Cluster Oracle

Input: representation $S = \{\boldsymbol{h}_i \in \mathbb{R}^d\}_{i=1}^n$, dimension k
$R \leftarrow \varnothing$
while $|R| < 1.5\varepsilon n$ **do**
  $C_1, C_2 \leftarrow \text{ActivationClustering}(S, k)$ [Algorithm 6]
  $C \leftarrow \text{ClusterOracle}(C_1, C_2)$
  Sample $\boldsymbol{h}$ uniformly from C
  Add $\boldsymbol{h}$ to R if $\boldsymbol{h} \notin R$
return samples corresponding to R

Algorithm 7 should perform well whenever the clustering is able to effectively separate the poisoned examples from clean ones and its performance should have relatively low variance as R is built using many independent clustering runs. Although this process is not guaranteed to terminate, we found that it did in all of our experiments.

B. Experimental results

B.1. Complete experimental results for pixel, periodic, and label consistent attacks

Complete experimental results for m-way pixel attacks, m-way periodic attacks, and label consistent attacks are shown in Tables 6 to 8 respectively.

m-Way Pixel Attack				PCA Defense			Clustering Defense			SPECTRE		
m	εn	$\mathrm{acc_p}$	$\mathrm{acc_{p^*}}$	$\mathrm{p_{rm}}$	$\mathrm{acc'_p}$	$\mathrm{acc'_{p^*}}$	$\mathrm{p_{rm}}$	$\mathrm{acc'_p}$	$\mathrm{acc'_{p^*}}$	$\mathrm{p_{rm}}$	$\mathrm{acc'_p}$	$\mathrm{acc'_{p^*}}$
1	500	0.942	0.942	471	0.004	0.004	375	0.820	0.820	500	0.000	0.000
1	250	0.894	0.890	103	0.880	0.880	54	0.904	0.904	249	0.001	0.001
1	125	0.627	0.627	0	0.834	0.834	11	0.842	0.842	122	0.000	0.000
1	62	0.331	0.331	0	0.519	0.519	2	0.297	0.297	59	0.000	0.000
1	31	0.075	0.075	0	0.023	0.023	0	0.010	0.010	30	0.000	0.000
1	15	0.001	0.001	0	0.001	0.001	1	0.002	0.002	0	0.000	0.000
2	500	0.830	0.987	172	0.675	0.914	186	0.631	0.901	495	0.000	0.000
2	250	0.588	0.888	9	0.503	0.817	35	0.518	0.808	237	0.002	0.002
2	125	0.058	0.106	0	0.058	0.139	6	0.148	0.325	118	0.000	0.000
2	62	0.009	0.017	0	0.007	0.011	1	0.002	0.007	59	0.000	0.000
2	31	0.002	0.002	0	0.000	0.000	0	0.000	0.000	25	0.000	0.000
2	15	0.000	0.000	0	0.001	0.000	0	0.000	0.000	0	0.000	0.000
3	500	0.742	0.990	147	0.665	0.970	204	0.606	0.963	486	0.001	0.000
3	250	0.503	0.908	0	0.367	0.367	35	0.482	0.914	241	0.001	0.000
3	125	0.225	0.616	0	0.083	0.348	4	0.186	0.547	122	0.000	0.000
3	62	0.003	0.010	0	0.002	0.008	0	0.013	0.025	57	0.000	0.001
3	31	0.001	0.001	0	0.000	0.002	0	0.000	0.001	0	0.000	0.000
3	15	0.000	0.000	0	0.001	0.000	0	0.001	0.000	0	0.002	0.002

Table 6: Under the m-way pixel attacks, the proposed robust poison detection in Algorithm 1 completely removes the backdoor for all $m \in \{1, 2, 3\}$ and all sizes of the poisoned data εn, achieving the retrained accuracy of near zero on backdoored test samples. On the other hand, the state-of-the-art PCA and clustering defenses fail to remove enough poisons on almost all cases. There are 5,000 clean training samples with the target label "deer". $\mathrm{acc_p}$ is the accuracy on poisoned test data with one pixel watermark and $\mathrm{acc_{p^*}}$ is the accuracy on poisoned test data with all m pixel watermarks simultaneously. $\mathrm{acc'_p}$ and $\mathrm{acc'_{p^*}}$ are the respective quantities after each defense has been applied and the network has been retrained. $\mathrm{p_{rm}}$ is the number of poisoned examples removed by the defense, out of $1.5\varepsilon n$ examples removed in total. Test accuracies on clean data were between 92.5% and 93.5% in all experiments and are omitted in the table.

m-Way Periodic Attack				PCA Defense			Clustering Defense			SPECTRE		
m	εn	$\mathrm{acc_p}$	$\mathrm{acc_{p^*}}$	$\mathrm{p_{rm}}$	$\mathrm{acc'_p}$	$\mathrm{acc'_{p^*}}$	$\mathrm{p_{rm}}$	$\mathrm{acc'_p}$	$\mathrm{acc'_{p^*}}$	$\mathrm{p_{rm}}$	$\mathrm{acc'_p}$	$\mathrm{acc'_{p^*}}$
1	500	0.975	0.975	19	0.976	0.976	151	0.987	0.987	493	0.004	0.004
1	250	0.961	0.961	2	0.968	0.968	40	0.933	0.933	249	0.001	0.001
1	125	0.912	0.912	0	0.916	0.916	16	0.889	0.889	123	0.000	0.000
1	62	0.744	0.744	0	0.764	0.764	4	0.722	0.722	62	0.001	0.001
1	31	0.318	0.318	0	0.329	0.329	0	0.440	0.440	28	0.003	0.003
1	15	0.003	0.003	0	0.005	0.005	0	0.002	0.002	0	0.007	0.007
2	500	0.896	0.996	176	0.873	0.995	172	0.824	0.988	499	0.001	0.001
2	250	0.813	0.982	10	0.817	0.986	63	0.666	0.961	248	0.000	0.000
2	125	0.501	0.881	0	0.460	0.868	10	0.416	0.829	124	0.000	0.000
2	62	0.118	0.359	0	0.070	0.280	1	0.058	0.209	61	0.002	0.003
2	31	0.012	0.057	0	0.001	0.010	0	0.015	0.067	0	0.004	0.021
2	15	0.001	0.004	0	0.001	0.005	0	0.004	0.001	0	0.004	0.008

Table 7: Under the m-way periodic attacks, the proposed robust poison detection in SPECTRE completely removes the backdoor for all $m \in \{1, 2\}$ and all sizes of the poisoned data εn, achieving the retrained accuracy of near zero on backdoored test samples. On the other hand, the state-of-the-art PCA and clustering defenses fail to remove enough poisons on almost all cases. There are 5,000 clean training samples with the target label "deer". Accuracies on clean data were between 92.5% and 93.5% in all experiments and are omitted in the table.

Attack type	εn	$\mathrm{acc_p}$	PCA Defense $\mathrm{p_{rm}}$	Clustering Defense $\mathrm{p_{rm}}$	SPECTRE $\mathrm{p_{rm}}$
ℓ_2	500	0.881	500	500	500
ℓ_2	250	0.932	250	140	250
ℓ_2	125	0.843	1	17	125
ℓ_2	62	0.856	0	5	62
ℓ_2	31	0.051	0	1	31
ℓ_2	15	0.018	0	0	0
ℓ_∞	500	0.798	500	500	500
ℓ_∞	250	0.894	250	245	250
ℓ_∞	125	0.744	0	24	125
ℓ_∞	62	0.472	0	5	62
ℓ_∞	31	0.024	0	0	31
ℓ_∞	15	0.017	0	0	0
GAN	500	0.633	500	500	500
GAN	250	0.584	47	78	250
GAN	125	0.680	28	20	125
GAN	62	0.261	0	2	62
GAN	31	0.022	0	0	0
GAN	15	0.010	0	0	0

Table 8: The number of removed poisoned examples $\mathrm{p_{rm}}$ under label consistent attacks. SPECTRE successfully removes all poisoned examples whenever the attack accuracy is larger than 10%. Accuracies on clean data were between 91% and 92.5% in all experiments and are omitted in the table.

B.2. Experimental results for hidden trigger attacks

We also ran experiments for the hidden trigger attack of (Saha et al., 2020). This attack applies in the transfer learning setting. To create poisoned images, a batch of images from the target label is selected. Next a batch of images from the source label is selected and watermarks are applied to them. Each corrupted source labelled image is paired with the target image that is closest in the representation space of the pretrained network. Then projected gradient descent is used to find small perturbations of the target images that bring their representations close to those of the watermarked images. This process is iterated until a suitably good set of perturbed images is found and these are added to the target label.

When the network is fine tuned on a dataset which has been poisoned this way, the perturbed images will behave similarly to the original watermarked images, constructing a backdoor which is triggered by the watermark. In production when given a watermarked image, the network will recognize it and activate the backdoor even though the watermark never appears in the training data. As in the label consistent attack, the perturbed images are visually similar to clean examples from the target label, so they are difficult to detect.

We tested the PCA defense, clustering defense, and our defense against the hidden trigger attack for $\varepsilon n \in \{400, 200, 100, 50\}$ where $n = 800$. The results are shown in Table 9.

Attack			PCA Defense	Clustering Defense	SPECTRE
ℓ_2 bound	εn	$\mathrm{acc_p}$	$\mathrm{p_{rm}}$	$\mathrm{p_{rm}}$	$\mathrm{p_{rm}}$
8	400	0.548	328	391	400
8	200	0.350	183	186	198
8	100	0.128	95	49	98
8	50	0.038	27	9	50
16	400	0.600	327	396	399
16	200	0.400	172	200	200
16	100	0.100	78	34	97
16	50	0.060	18	3	48
32	400	0.510	331	392	400
32	200	0.312	183	189	199
32	100	0.128	93	41	97
32	50	0.028	29	15	50

Table 9: The number of removed poisoned examples $\mathrm{p_{rm}}$ under hidden trigger attacks. The ℓ_2 bound is projected gradient descent perturbation norm limit. SPECTRE successfully removes nearly all poisoned examples in all cases. Accuracies on clean data were between 98% and 99.5% in all experiments and are omitted in the table.

More details regarding the experimental setup are given in Appendix E.4.

B.3. Experimental results for different source-target label pairs

In our previous experiments, we chose "deer" as the source label and "truck" as the target label following (Tran et al., 2018). We also ran the m-way pixel attack experiments for $m \in \{1, 3\}$ and $\varepsilon n \in \{500, 125\}$ for ten combinations of source and target labels. The labels are airplane (0), automobile (1), bird (2), cat (3), deer (4), dog (5), frog (6), horse (7), ship (8), and truck (9). The results are shown in Table 10. Overall the trend in performance is similar, although there are some cases where none of the defences work well (e.g., $(\ell_\mathrm{s}, \ell_\mathrm{t}, m, \varepsilon n) = (5, 3, 3, 125)$). We suspect that this is because the representations of the clean and poisoned samples are merged at an earlier point in the network, making them difficult to distinguish once they reach the penultimate residual block. We believe exploring this phenomenon presents an interesting research direction.

m-Way Pixel Attack						PCA Defense			Clustering Defense			SPECTRE		
ℓ_s	ℓ_t	m	εn	acc_p	acc_{p^*}	p_{rm}	acc'_p	acc'_{p^*}	p_{rm}	acc'_p	acc'_{p^*}	p_{rm}	acc'_p	acc'_{p^*}
0	9	1	500	0.978	0.978	397	0.655	0.655	254	0.979	0.970	496	0.002	0.002
0	9	1	125	0.913	0.913	3	0.865	0.865	11	0.845	0.845	124	0.009	0.009
0	9	3	500	0.834	0.995	15	0.823	0.997	79	0.814	0.996	374	0.223	0.576
0	9	3	125	0.464	0.868	0	0.475	0.890	3	0.158	0.474	47	0.013	0.025
1	7	1	500	0.963	0.963	195	0.933	0.933	237	0.905	0.905	500	0.001	0.001
1	7	1	125	0.758	0.758	0	0.665	0.665	17	0.750	0.750	125	0.000	0.000
1	7	3	500	0.765	0.986	15	0.714	0.979	138	0.687	0.969	498	0.000	0.000
1	7	3	125	0.2	0.598	0	0.127	0.441	5	0.313	0.746	122	0.001	0.001
2	5	1	500	0.963	0.963	417	0.682	0.682	259	0.985	0.985	493	0.026	0.026
2	5	1	125	0.758	0.758	94	0.020	0.020	13	0.956	0.956	119	0.024	0.024
2	5	3	500	0.765	0.986	17	0.781	0.995	66	0.789	0.991	375	0.042	0.099
2	5	3	125	0.2	0.598	1	0.306	0.754	4	0.043	0.187	27	0.055	0.196
3	8	1	500	0.993	0.993	491	0.004	0.004	355	0.966	0.966	500	0.003	0.003
3	8	1	125	0.94	0.940	0	0.941	0.941	26	0.935	0.935	125	0.003	0.003
3	8	3	500	0.825	0.997	1	0.819	0.998	152	0.601	0.947	482	0.006	0.004
3	8	3	125	0.131	0.448	0	0.102	0.340	5	0.021	0.074	113	0.002	0.005
4	1	1	500	0.951	0.951	283	0.994	0.994	252	0.986	0.986	500	0.001	0.001
4	1	1	125	0.951	0.951	0	0.956	0.956	8	0.944	0.944	125	0.001	0.001
4	1	3	500	0.89	0.996	0	0.851	0.998	107	0.782	0.994	461	0.003	0.007
4	1	3	125	0.159	0.536	0	0.226	0.657	4	0.376	0.822	0	0.074	0.346
5	3	1	500	0.99	0.990	423	0.357	0.357	355	0.911	0.911	495	0.072	0.072
5	3	1	125	0.944	0.944	10	0.878	0.878	4	0.905	0.905	118	0.075	0.075
5	3	3	500	0.815	0.998	159	0.619	0.940	74	0.745	0.995	400	0.107	0.146
5	3	3	125	0.22	0.533	6	0.206	0.516	2	0.263	0.655	1	0.286	0.668
6	2	1	500	0.99	0.990	262	0.981	0.981	179	0.980	0.980	497	0.014	0.014
6	2	1	125	0.962	0.962	15	0.948	0.948	6	0.954	0.954	122	0.021	0.021
6	2	3	500	0.712	0.984	93	0.678	0.975	78	0.672	0.989	300	0.028	0.048
6	2	3	125	0.066	0.208	0	0.082	0.267	3	0.104	0.313	0	0.065	0.211
7	0	1	500	0.998	0.998	459	0.044	0.044	292	0.964	0.964	500	0.009	0.009
7	0	1	125	0.923	0.923	1	0.882	0.882	17	0.915	0.915	125	0.010	0.010
7	0	3	500	0.882	1.000	14	0.790	0.997	168	0.635	0.974	489	0.009	0.018
7	0	3	125	0.178	0.574	0	0.281	0.689	3	0.223	0.611	108	0.005	0.014
8	6	1	500	0.964	0.964	491	0.001	0.001	245	0.957	0.957	500	0.000	0.000
8	6	1	125	0.902	0.902	0	0.894	0.894	14	0.888	0.888	123	0.000	0.000
8	6	3	500	0.739	0.992	3	0.751	0.994	138	0.712	0.987	428	0.005	0.006
8	6	3	125	0.447	0.918	0	0.493	0.939	9	0.526	0.954	119	0.002	0.002

Table 10: The number of removed poisoned examples p_{rm} under m-way pixel attacks for various choices of the source label ℓ_s and target label ℓ_t. Accuracy on clean data was between 91% and 92.5% in all experiments and are omitted in the table.

C. Ablation study

SPECTRE combines several steps to effectively detect poisoned examples.

1. Adaptive dimension reduction using Algorithm 3.
2. The covariance of the clean samples is estimated using Algorithm 10.
3. The samples are whitened using the estimated covariance.

4. We compute QUE scores using Algorithm 2 to determine which samples to discard.

Here we perform an ablation study to demonstrate that none of this steps can be omitted. We show that Step 1 is necessary in Section 4.4, where we show that no constant choice of k is sufficient to detect the majority of the poison across multiple experiments. Note that choosing $k = d$ is equivalent to performing no dimension reduction. In our experiments, we found that checking values of k which are substantially smaller than d sufficed. This also gave us a substantial computational speedup since the runtime of Algorithm 1 scales with k. We show that Step 4 is important in Section 3.3. In particular, in Table 1 we show that two other natural choices for outlier scoring can fail under certain conditions. For Steps 2 and 3, we provide Table 11, which shows the performance of Algorithm 1 on a variety of experiments where Step 3 has been omitted (removing the need for Step 2) and where Step 2 is omitted, and the whitening is done using the sample covariance. The results in Table 11 justify the use of Steps 2 and 3.

Attack type	m	εn	acc_{p^*}	1+4 p_{rm}	1+3+4 p_{rm}	1+2+3+4 p_{rm}
pixel	1	500	0.942	471	471	500
pixel	1	250	0.894	131	203	249
pixel	1	125	0.627	0	51	124
pixel	3	500	0.990	153	336	490
pixel	3	250	0.908	0	119	245
pixel	3	125	0.616	0	37	123
periodic	1	500	0.975	19	421	493
periodic	1	250	0.961	2	105	248
periodic	1	125	0.912	0	67	124
periodic	2	500	0.996	457	407	493
periodic	2	250	0.982	10	115	248
periodic	2	125	0.881	0	0	124
ℓ_2	1	500	0.881	500	500	500
ℓ_2	1	250	0.932	250	250	250
ℓ_2	1	125	0.843	1	125	125
GAN	1	500	0.633	500	500	500
GAN	1	250	0.584	246	239	250
GAN	1	125	0.680	79	124	125

Table 11: Performance for various combinations of: 1. adaptive dimension reduction, 2. robust covariance estimation, 3. whitening, 4. QUE scoring. Note: 1+2+3+4 is SPECTRE, which performs better than the other combinations.

D. Robust estimation

We reproduce details from (Diakonikolas et al., 2017a) which are relevant to the implementation and usage of Algorithm 1 here for completeness. First, we introduce some notations. Given two sets A and B, $\Delta(A, B)$ is the size of their symmetric difference $|(A \setminus B) \cup (B \setminus A)|$. Given a matrix $M \in \mathbb{R}^{d \times d}$, we write $M^\flat$ to denote the flattened vector $\boldsymbol{v} \in \mathbb{R}^{d^2}$ built by concatenating the columns of M. Similarly, given a vector $\boldsymbol{v} \in \mathbb{R}^{d^2}$, we write $v^\sharp$ to denote the matrix $M \in \mathbb{R}^{d \times d}$ with $\boldsymbol{v}_i$ as columns, where $\boldsymbol{v}$ is split into d contiguous vectors in $\mathbb{R}^d$.

D.1. Robust mean estimation

There exists a practical robust mean estimation algorithm RobustMean which is given explicitly in Algorithm 8.

Understanding Algorithm 10 requires the definition of an (ε, τ)-good set with respect to a Gaussian, which is given in Definition D.1. The key feature of (ε, τ)-goodness is that a set of independent samples from the Gaussian of sufficient size is (ε, τ)-good with high probability as stated in Lemma D.2.

Definition D.1. *(Diakonikolas et al., 2017a, Definition A.4) Let G be a sub-gaussian distribution in d dimensions with mean $\boldsymbol{\mu}(G)$ and covariance matrix I and let $\varepsilon, \tau > 0$. We say that a multiset S of elements in $\mathbb{R}^d$ is (ε, τ)-good with respect to G if the following conditions are satisfied:*

Algorithm 8: Robust mean estimation (ROBUSTMEAN) (Diakonikolas et al., 2017a)

Input: A multiset S' such that there exists an (ε, τ)-good set S with $\Delta(S, S') < 2\varepsilon$
Output: A vector $\boldsymbol{\mu}'$ such that $\|\boldsymbol{\mu}' - \boldsymbol{\mu}(G)\|_2 \leq O(\varepsilon\sqrt{\log(1/\varepsilon)})$
repeat
  $S' \leftarrow$ GAUSSIANMEANFILTER(S') [Algorithm 9]
until GAUSSIANMEANFILTER *returns* $\boldsymbol{\mu}'$
return $\boldsymbol{\mu}'$

1. *For all* $\boldsymbol{x} \in S$ *we have* $\|\boldsymbol{x} - \boldsymbol{\mu}(G)\|_2 \leq O(\sqrt{d\log(|S|/\tau)})$.
2. *For every affine function* $L : \mathbb{R}^d \to \mathbb{R}$ *such that* $L(\boldsymbol{x}) = \boldsymbol{v} \cdot (\boldsymbol{x} - \boldsymbol{\mu}(G)) - T$, $\|\boldsymbol{v}\|_2 = 1$, *we have that*
$$\left| \Pr_{X \in_u S}[L(X) \geq 0] - \Pr_{X \sim G}[L(X) \geq 0] \right| \leq \frac{\varepsilon}{T^2 \log\left(d\log(\frac{d}{\varepsilon\tau})\right)}$$
3. *We have that* $\|\boldsymbol{\mu}(S) - \boldsymbol{\mu}(G)\|_2 \leq \varepsilon$.
4. *We have that* $\|M_s - I\|_2 \leq \varepsilon$.

Lemma D.2. *(Diakonikolas et al., 2017a, Lemma A.6) Let* G *be a sub-gaussian distribution with parameter* $\nu = \Theta(1)$ *and identity covariance and let* $\varepsilon, \tau > 0$. *If the multiset* S *is obtained by taking* $\Omega((d/\varepsilon^2)\operatorname{poly}\log(d/\varepsilon\tau))$ *independent samples from* G, *it is* ε*-good with respect to* G *with probability at least* $1 - \tau$.

Now we give the definition of the filter used in Algorithm 8 in Algorithm 9, which shows that the sets S' in Algorithm 8 approach the ε-good set S with respect to the size of their symmetric difference.

Algorithm 9: Filter algorithm for a Gaussian with unknown mean. (Diakonikolas et al., 2017a, Algorithm 2)

Input: A multiset S' such that there exists an (ε, τ)-good set S with $\Delta(S, S') < 2\varepsilon$
Output: Either a set S'' with $\Delta(S, S'') \leq \Delta(S, S') - \varepsilon/\alpha$ where $\alpha \triangleq d\log(d/\varepsilon\tau)\log(d\log(d/\varepsilon\tau))$ or a vector $\boldsymbol{\mu}$ satisfying $\|\boldsymbol{\mu}' - \boldsymbol{\mu}(G)\|_2 \leq O(\varepsilon\sqrt{\log(1/\varepsilon)})$
Compute the sample mean $\boldsymbol{\mu}(S') = \mathbb{E}_{X \sim \mathrm{Unif}(S')}[X]$.
Compute the sample covariance matrix $\Sigma(S') = \mathbb{E}_{X \in \mathrm{Unif}(S')}[(X - \boldsymbol{\mu}(S'))(X - \mu(S'))^\top]$.
Compute an approximation of the largest absolute eigenvalue of $\Sigma - I$, $\lambda^* \approx \|\Sigma - I\|_2$ and an approximate associated eigenvector $\boldsymbol{v}^*$.
if $\lambda^* \leq O(\varepsilon\log(1/\varepsilon))$ **then**
  return $\boldsymbol{\mu}(S')$
Let $\delta = 3\sqrt{\varepsilon\lambda^*}$. Find a $T > 0$ such that
$$\Pr_{X \in \mathrm{Unif}(S')}(|\boldsymbol{v}^* \cdot (X - \boldsymbol{\mu}(S'))| > T + \delta) > 8\exp\left(-\frac{T^2}{2\nu}\right) + \frac{8\varepsilon}{T^2\log(d\log(\frac{d}{\varepsilon\tau}))}.$$
return $S'' = \{\boldsymbol{x} \in S'' : |\boldsymbol{v}^* \cdot (\boldsymbol{x} - \boldsymbol{\mu}(S'))| \leq T + \delta\}$

D.2. Robust covariance estimation

The structure of this subsection mirrors that of Appendix D.1. Theorem 1 states the existence of a practical robust covariance estimation algorithm ROBUSTCOV which is given explicitly in Algorithm 10.

Understanding Algorithm 10 requires the definition of an (ε-good set with respect to a Gaussian, which is given in Definition D.3. The key feature of ε-goodness is that a set of independent samples from the Gaussian of sufficient size is ε-good with high probability as stated in Proposition D.4.

Definition D.3. *(Diakonikolas et al., 2017a, Definition A.27) Let* G *be a Gaussian in* $\mathbb{R}^d$ *with mean* 0 *and covariance* Σ. *Let* $\varepsilon > 0$ *be sufficiently small. We say that a multiset* S *of points in* $\mathbb{R}^d$ *is* ε*-good with respect to* G *if the following hold:*

Algorithm 10: Robust covariance estimation (ROBUSTCOV) (Diakonikolas et al., 2017a)

Input: A multiset S' such that there exists an ε-good set S with $\Delta(S, S') < 2\varepsilon$
Output: A matrix Σ' such that $\|I - \Sigma^{-1/2}\Sigma'\Sigma^{-1/2}\|_F = O(\varepsilon \log(\frac{1}{\varepsilon}))$
repeat
  $S' \leftarrow$ GAUSSIANCOVARIANCEFILTER(S') [Algorithm 11]
until GAUSSIANCOVARIANCEFILTER *returns* Σ'
return Σ'

1. *For all* $\boldsymbol{x} \in S$, $\boldsymbol{x}^\top \Sigma^{-1} \boldsymbol{x} < d + O(\sqrt{d}\log(d/\varepsilon))$.
2. *We have that* $\|\Sigma^{-1/2}\operatorname{Cov}(S)\Sigma^{-1/2} - I\|_F = O(\varepsilon)$.
3. *For all even degree-2 polynomials* p, *we have that* $\operatorname{Var}(p(\boldsymbol{x})) = \operatorname{Var}(p(G))(1 + O(\varepsilon))$.
4. *For* p *an even degree-2 polynomial with* $\mathbb{E}[p(G)] = 0$ *and* $\operatorname{Var}(p(G)) = 1$, *and for any* $T > 10\log(1/\varepsilon)$ *we have that*

$$\Pr(|p(x)| > T) \leq \frac{\varepsilon}{T^2 \log^2(T)}.$$

Proposition D.4. *(Diakonikolas et al., 2017a, Proposition A.28) Let* N *be a sufficiently large constant multiple of* $(d^2/\varepsilon^2)\log^5(d/\varepsilon)$. *Then a set* S *of* N *independent samples from* G *is* ε*-good with respect to* G *with high probability.*

Now we give the definition of the filter used in Algorithm 10 in Algorithm 11, which shows that the sets S' in Algorithm 10 approach the (ε, τ)-good set S with respect to the size of their symmetric difference.

Algorithm 11: Filter algorithm for a Gaussian with unknown covariance matrix. (Diakonikolas et al., 2017a, Algorithm 4)

Input: A multiset S' such that there exists an ε-good set S with $\Delta(S, S') < 2\varepsilon$
Output: Either a set S'' with $\Delta(S, S'') < \Delta(S, S')$ or a matrix Σ' such that $\|I - \Sigma^{-1/2}\Sigma'\Sigma^{-1/2}\|_F = O(\varepsilon \log(\frac{1}{\varepsilon}))$
Let $C, C' > 0$ be sufficiently large universal constants.
$\Sigma' \leftarrow \mathbb{E}_{X \in S'}[XX^\top]$
$G' \leftarrow \mathcal{N}(0, \Sigma')$
if *there exists an* $\boldsymbol{x} \in S'$ *such that* $\boldsymbol{x}^\top \Sigma'^{-1} \boldsymbol{x} \geq Cd\log(10|S'|)$ **then**
  return $S'' = S' \setminus \{\boldsymbol{x} \in S' : \boldsymbol{x}^\top \Sigma'^{-1} \boldsymbol{x} > Cd\log(10|S'|)\}$
Let L be the space of even degree-2 polynomials $p : \mathbb{R}^k \to \mathbb{R}$ such that $\mathbb{E}_{X \sim G'}[p(X)] = 0$.
Define two quadratic forms on L:

(i) $Q_{G'}(p) = \mathbb{E}_{X \sim G'}[p^2(X)]$

(ii) $Q_{S'}(p) = \mathbb{E}_{X \sim \mathrm{Unif}(S')}[p^2(X)]$

Compute $\max_{p \in L \setminus \{0\}} Q_{S'}(p)/Q_{G'}(p)$ and the associated polynomial $p^*(x)$ normalized such that $Q_{G'}(p) = 1$ using Algorithm 12.
if $Q_{S'}(p^*) \leq (1 + C\varepsilon\log^2(1/\varepsilon))Q_{G'}(p^*)$ **then**
  return Σ'
$\mu \leftarrow$ the median value of $p^*(X)$ over $X \in S'$
Find a $T > C'$ such that

$$\Pr_{X \in T'}(|p^*(X) - \mu| \geq 3) \leq \mathrm{Tail}(T, d, \varepsilon),$$

where

$$\mathrm{Tail}(T, d, \varepsilon) = \begin{cases} 3\varepsilon/(T^2\log^2(T)) & \text{if } T \geq 10\ln(1/\varepsilon) \\ 1 & \text{otherwise} \end{cases}.$$

return $S'' = \{\boldsymbol{x} \in S'' : |p^*(U'^\top \boldsymbol{x}) - \mu| \leq T\}$

Algorithm 12: Algorithm to compute the polynomial with maximum variance relative to a Gaussian (Diakonikolas et al., 2017a, Algorithm 4)

Input: A multiset $S' = \{\boldsymbol{x}_i\}_{i=1}^n \subset \mathbb{R}^d$ and a Gaussian $G' = \mathcal{N}(0, \Sigma')$
Output: The even degree-2 polynomial $p^*(\boldsymbol{x})$ with $\mathbb{E}_{X\sim G'}[p(X)] \approx 0$ and $Q_{G'}(p^*) \approx 1$ that approximately maximizes $Q_{S'}(p^*)$ and this maximum is $\lambda^* = Q_{S'}(p^*)$
for $i \in [n]$ **do**
  $\boldsymbol{y}_i \leftarrow \Sigma_k'^{-1/2}\boldsymbol{x}_i$
  $\boldsymbol{z}_i \leftarrow (\boldsymbol{y}_i\boldsymbol{y}_i^\top)^\flat$
$T_{S'} \leftarrow -I^\flat I^{\flat\top} + \frac{1}{|S'|}\sum_{i=1}^n \boldsymbol{z}_i\boldsymbol{z}_i^\top$
Approximate the top eigenvalue λ^* and eigenvector $\boldsymbol{v}^*$ of $T_{S'}$
$p*(x) \leftarrow \frac{1}{\sqrt{2}}((\Sigma_k'^{-1/2}\boldsymbol{x})v^{*\sharp}(\Sigma_k'^{-1/2}\boldsymbol{x}) - \mathrm{Tr}(\boldsymbol{v}^{*\sharp}))$
return p^* and λ^*

Note that a naive implementation of Algorithm 12 requires $\Omega(nd^2)$ space to store the $\boldsymbol{y}_i$ and $\Omega(d^4)$ space to store $T_{S'}$. Additionally, the matrix multiplication performed by OpenBLAS to produce $T_{S'}$ requires $\Omega(nd^4)$ time. By representing the linear operator $T_{S'}$ implicitly, we can reduce these requirements substantially. First, the product $-I^\flat(I^{\flat\top}\boldsymbol{v})$ can be computed in $O(d^2)$ time and space. Next, if Y and Z are the matrices with columns $\boldsymbol{y}_i$ and $\boldsymbol{z}_i$ respectively, then Z is the Khatri-Rao product $Y \odot Y$. This means we can use the vec tricks for the Khatri-Rao and transpose Khatri-Rao vector products of (Periša, 2017) to calculate $ZZ^\top\boldsymbol{v}$ in $O(nd^2)$ time and $O(nd + d^2)$ space. We can then calculate the eigenvector $\boldsymbol{v}^*$ of the implicitly represented linear operator $T_{S'}$ using Krylov methods, requiring the evaluation of a small number of products $T_{S'}\boldsymbol{v}$. For our experiments, this provided a speedup of several orders of magnitude and a substantial reduction in the required amount of system memory versus the naive implementation.

D.3. Robust joint mean and covariance estimation

Note that Algorithm 8 requires the inputs to have identity covariance and Algorithm 10 requires the inputs to have zero mean. Here we show how to combine them to estimate both the mean and covariance of an arbitrary Gaussian, as described in (Diakonikolas et al., 2017a, Section 4.5). The key idea is to split the dataset into two halves, pair off samples from each half, and subtract them. The resulting vectors have zero mean and double the original covariance. This allows us to use Algorithm 10 to whiten the samples, which then allows us to use Algorithm 8. We reproduce the exact procedure in Algorithm 13.

Algorithm 13: Algorithm to robustly learn an arbitrary Gaussian (Diakonikolas et al., 2019, Algorithm 6)

Input: A multiset $S' = \{\boldsymbol{x}_i\}_{i=1}^n \subset \mathbb{R}^d$, corruption fraction ε
Output: A matrix Σ' such that $\|I - \Sigma^{-1/2}\Sigma'\Sigma^{-1/2}\|_F = O(\varepsilon\log(\frac{1}{\varepsilon}))$ and Aavector $\boldsymbol{\mu}'$ such that $\|\boldsymbol{\mu}' - \boldsymbol{\mu}(G)\|_2 \le O(\varepsilon\sqrt{\log(1/\varepsilon)})$
for $i \in [\lfloor n/2 \rfloor]$ **do**
  $\boldsymbol{x}_i' \leftarrow (\boldsymbol{x}_i - \boldsymbol{x}_{\lfloor n/2\rfloor+1})/\sqrt{2}$
$\widehat{\Sigma} \leftarrow$ RobustCov$(\{\boldsymbol{x}_i'\}, \varepsilon)$ [Algorithm 10]
for $i \in [n]$ **do**
  $\boldsymbol{x}_i'' \leftarrow \widehat{\Sigma}^{-1/2}\boldsymbol{x}_i$
$\widehat{\boldsymbol{\mu}} \leftarrow$ RobustMean$(\{\boldsymbol{x}_i''\}, \varepsilon)$ [Algorithm 8]
return $\widehat{\Sigma}$ and $\widehat{\Sigma}^{1/2}\widehat{\boldsymbol{\mu}}$

E. Experiment details

For each poisoned dataset, we performed one training run to produce each poisoned model. For the pixel and periodic attacks, we performed one retraining run for each defense. Training for our experiments was done on a server with a Xeon Gold 6230 CPU and eight Nvidia 2080 Ti GPUs. The training and retraining for our experiments took approximately 100

GPU hours. Running all defences for our experiments took approximately 200 CPU-core hours. Using the thermal design power of these components to estimate of our required power, we estimate that our experiments required a total of 28 kW h of energy.

E.1. m-way pixel attacks

For pixel attacks, we reproduce the experimental setup of (Tran et al., 2018). For our ResNet-32, we used a leaky ReLU with a negative slope of 0.1 for the nonlinearity and trained it using stochastic gradient descent with momentum for 200 epochs, dividing the learning rate by 10 every 75 epochs. Both data standardization and augmentation were used.

Although a fixed pixel is used for watermarking, data augmentation may ensure that the network is sensitive to pixels of the chosen color at multiple locations in the image. Using the standard random horizontal flip and random crop with 4 pixels of padding used for CIFAR-10, the pixel may end up in as many as $9 \times 9 \times 2 = 162$ distinct pixels in the transformed image, representing about 16% of the image's total area.

To implement an m-way pixel attack, m pairs of locations and colors are chosen. Only one of the m pixels is used for each poisoned training example, but all m are used simultaneously at test time. We ran experiments for $m \in \{1, 2, 3\}$. We used the same backdoor pixel Tran et al. used for their experiments, along with two more arbitrarily chosen. The exact locations and colors are shown in Table 12.

m	location	color
1	(11, 16)	#650019
2	(5, 27)	#657B79
3	(30, 7)	#002436

Table 12: Pixel watermarks used for the m-way pixel attacks. Location is a pixel coordinate in (x, y) format and color is a 24-bit hexadecimal color in HTML format.

E.2. m-way periodic attacks

For periodic attacks, we used the same network architecture and training environment used for pixel attacks. Although the phase of the signal is fixed for watermarking, the signal will be shifted by a random amount at training time due to the random flip and random crop and pad, in a manner similar to the pixel attack. Because our signals have a period of 4 pixels, which equals the maximum translation produced by the data augmentation, the backdoored network should be sensitive to signals with any phase.

E.3. Label consistent attacks

For label consistent attacks, we used the experimental setup of (Turner et al., 2019) which is provided at `https://github.com/MadryLab/label-consistent-backdoor-code`. The setup of (Turner et al., 2019) for CIFAR-10 appears to be very similar to that of (Tran et al., 2018). The same ResNet-32 architecture is used, albeit with a normal (i.e. not leaky) ReLU. Data standardization was enabled by default. Data augmentation was disabled by default, but we enabled it to ensure greater consistency with our previous experiments. We used an ℓ_2 perturbation bound of $\epsilon = 300$, an ℓ_∞ perturbation bound of $\epsilon = 8$, and a GAN perturbation bound of $\tau = 0.2$. We also enabled patch placement on all four corners to ensure the watermark would not be cropped out. For this family of attacks, we did not make any changes to the training system of (Turner et al., 2019), which does not provide retraining.

E.4. Hidden trigger attacks

For hidden trigger attacks, we used the experimental setup of (Saha et al., 2020) which is provided at `https://github.com/UMBCvision/Hidden-Trigger-Backdoor-Attacks`. The setup for hidden trigger attacks differs substantially from that of the other attacks in this work. The dataset used is a subset of ImageNet containing examples with label n04243546 or n03584254. (Saha et al., 2020) fine tunes Alexnet on the resulting binary classification task using SGD and constructs a backdoor using n04243546 as the source label and n03584254 as the target label. We used the default settings which enable data standardization but disable data augmentation and uses a patch size of 30x39 for the watermark.

We used the activations of the penultimate layer for our representations. We did not make any changes to the training system of (Saha et al., 2020), which does not provide retraining.

F. Analysis of poisoned representations

Here we include Figs. 12 to 15, which illustrate some relevant properties of the hidden layer activations of examples bearing the target layer under a successful backdoor poisoning attack.

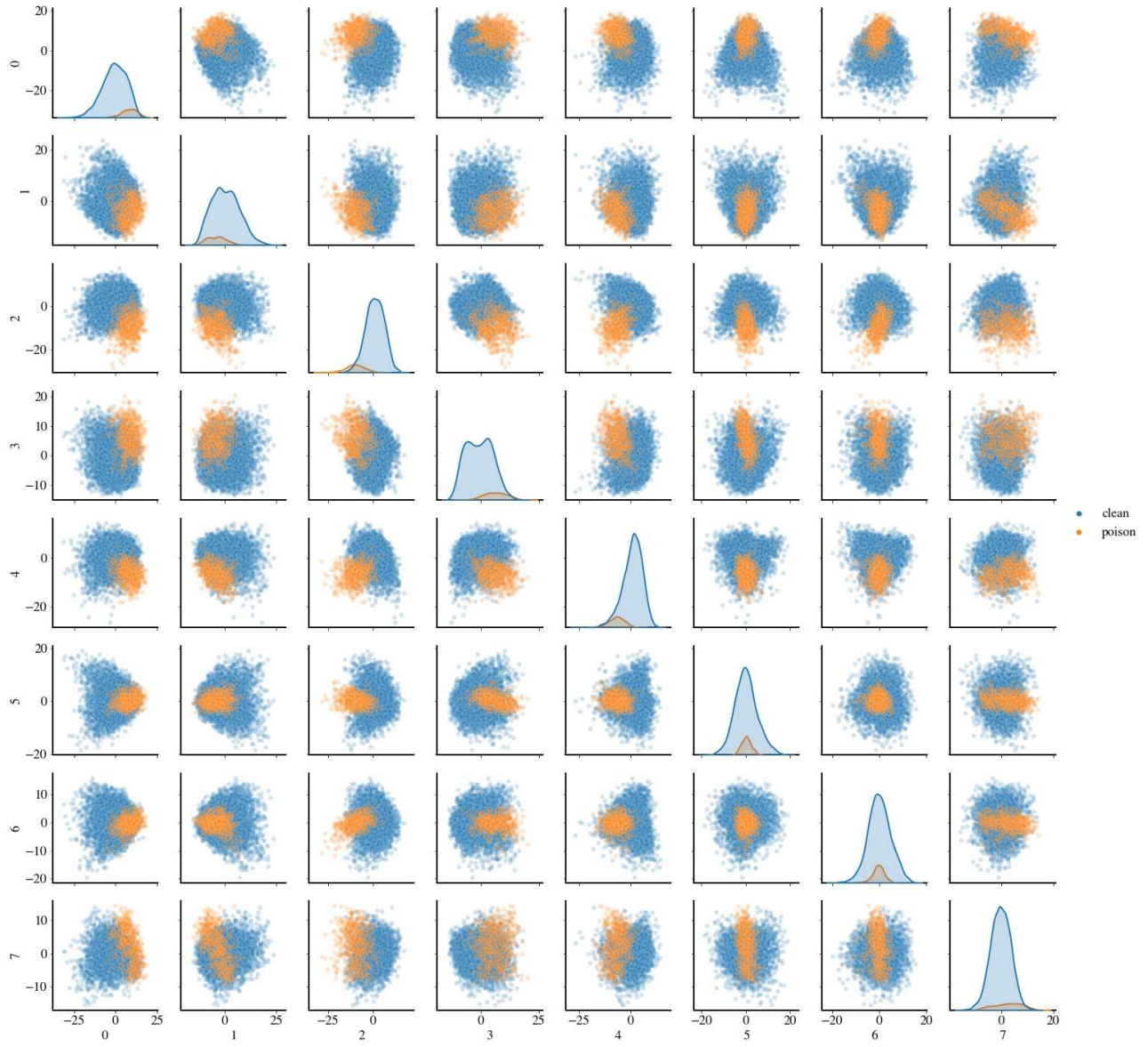


Figure 12: Scatter plots of the representations of the 3-way pixel attack with $\varepsilon = 0.1$ before any whitening. The whitened representations are projected onto their top eight PCA directions. Plots along the diagonal are Gaussian kernel density estimate plots after projecting onto that PCA direction (of the combined data including the representations of both the poisoned and the clean samples). Off-diagonal plots are scatter plots of the data projected onto the subspace spanned by the corresponding pair of PCA directions. This shows that the poisoned samples (in orange) are not separable from the clean ones (in blue), if we only focus on these top PCA directions; the spectral signature is hidden. We propose using robust covariance estimation to fine the approximate covariance of clean data and whiten the entire data with the estimated covariance. This enhances the spectral signature as we show in the next figure.

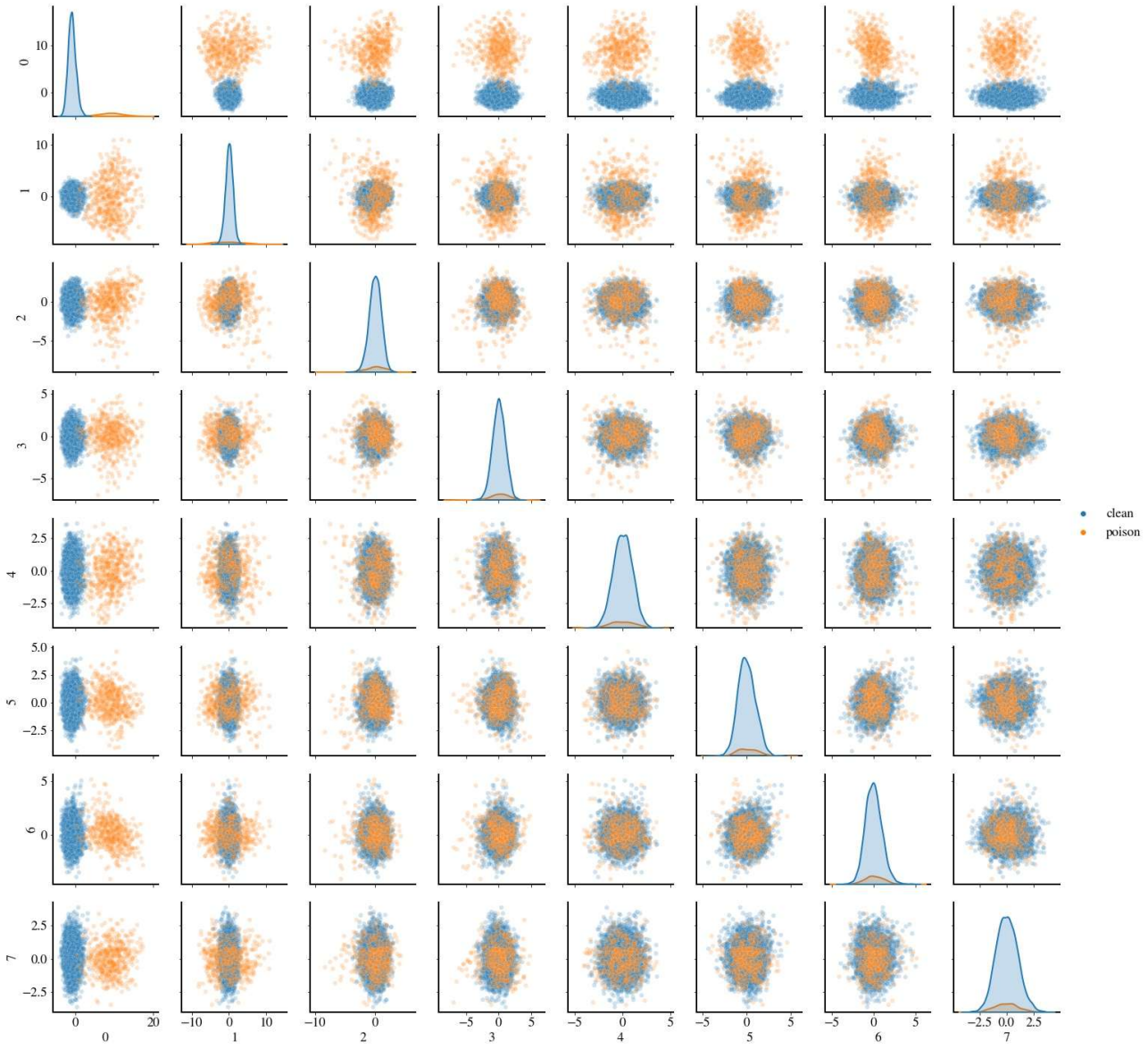


Figure 13: Scatter plots of the representations of the 3-way pixel attack with $\varepsilon = 0.1$ after whitening using the covariance of the clean samples. The whitened representations are projected onto their top eight PCA directions. Plots along the diagonal are Gaussian kernel density estimate plots after projecting onto that PCA direction (of the combined data including the representations of both the poisoned and the clean samples). Off-diagonal plots are scatter plots of the data projected onto the subspace spanned by the corresponding pair of PCA directions. This shows that the poisoned samples (in orange) are now separable from the clean ones (in blue) using the top PCA direction after whitening, for example.

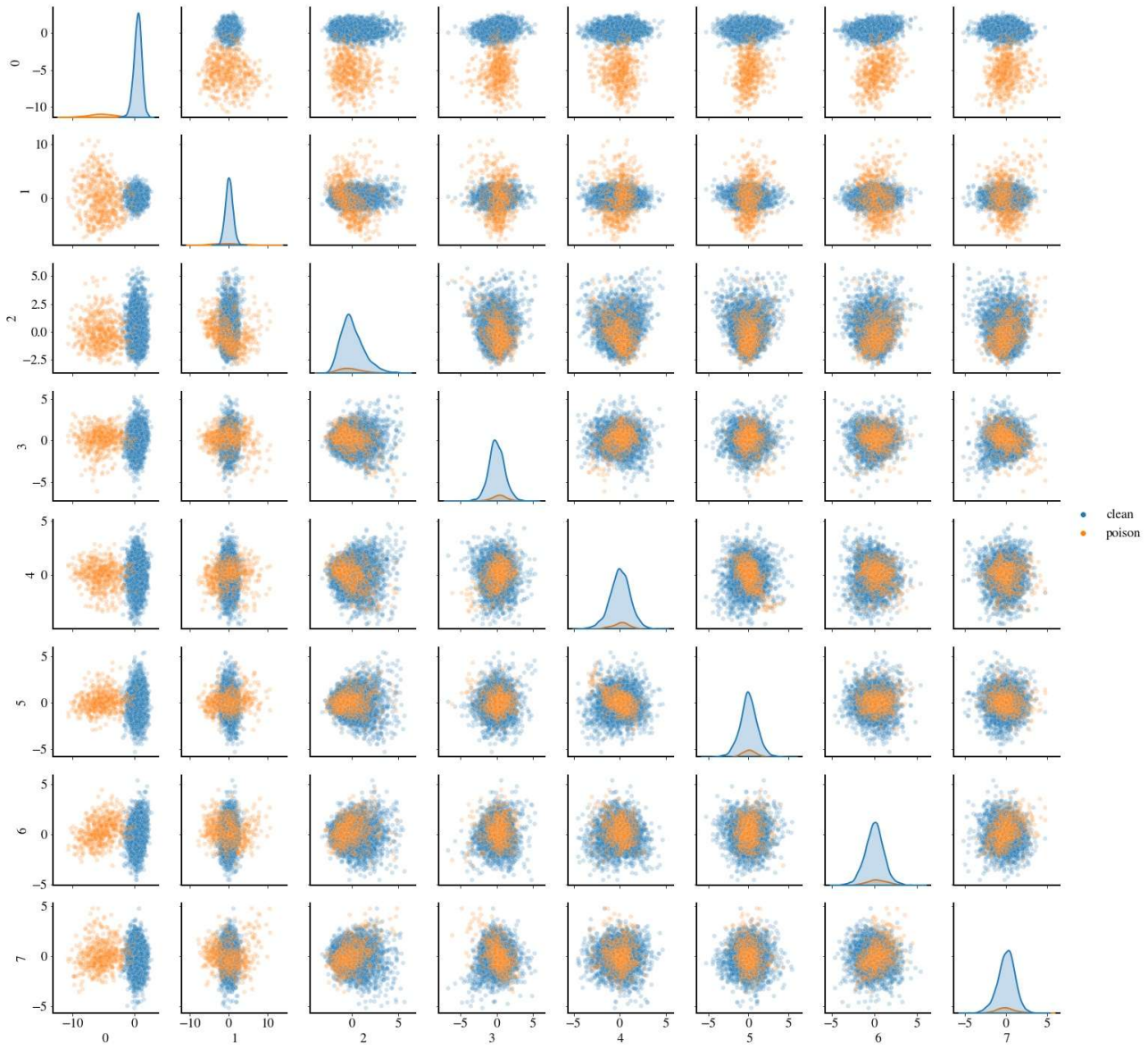


Figure 14: Scatter plots of the representations of the 3-way pixel attack with $\varepsilon = 0.1$ after whitening using the robustly estimated covariance. The whitened representations are projected onto their top eight PCA directions. Plots along the diagonal are Gaussian kernel density estimate plots after projecting onto that PCA direction (of the combined data including the representations of both the poisoned and the clean samples). Off-diagonal plots are scatter plots of the data projected onto the subspace spanned by the corresponding pair of PCA directions. This shows that the poisoned samples (in orange) remain separable from the clean ones (in blue) even when whitening using the estimated covariance instead of the true covariance of the clean samples.

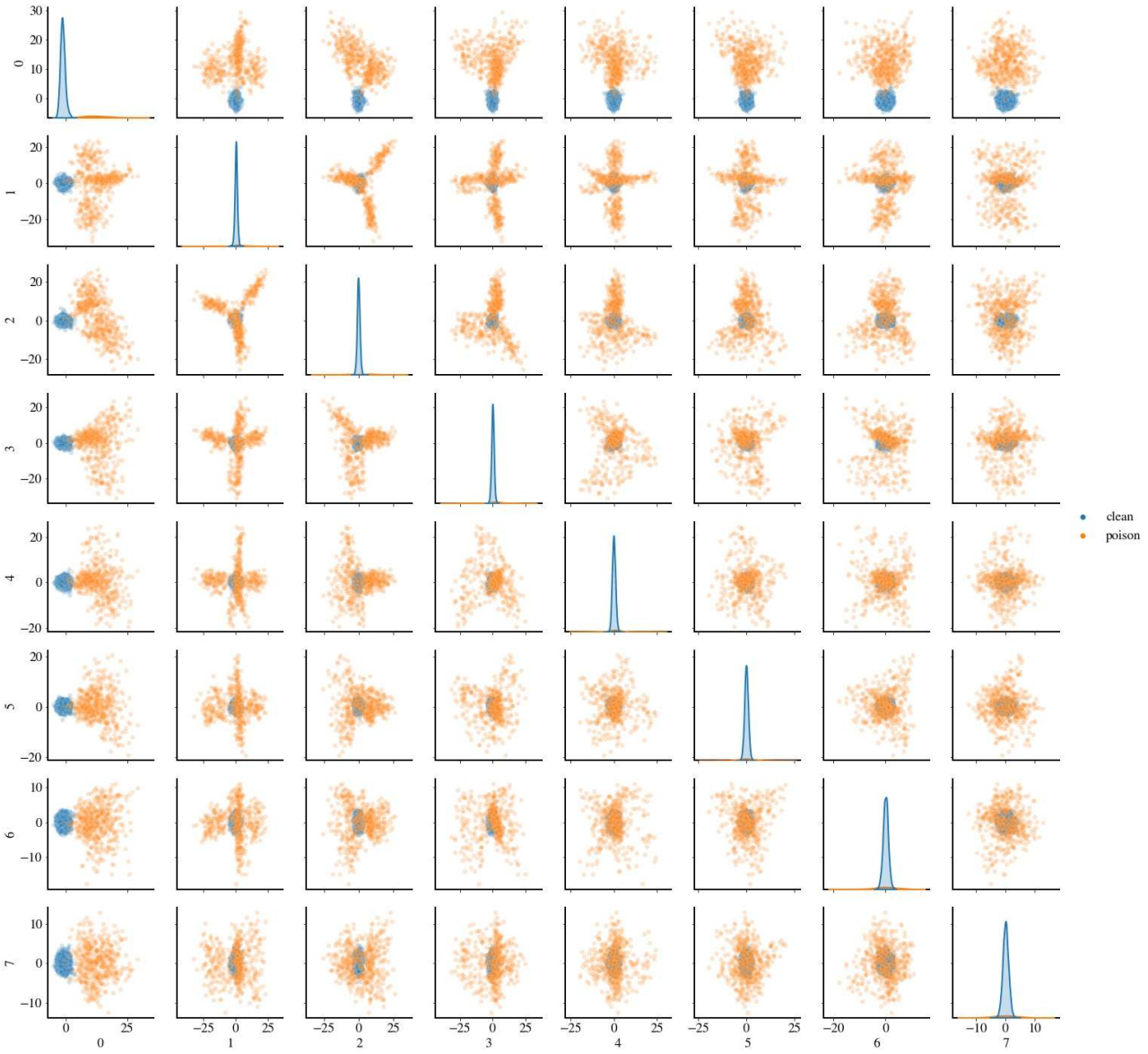


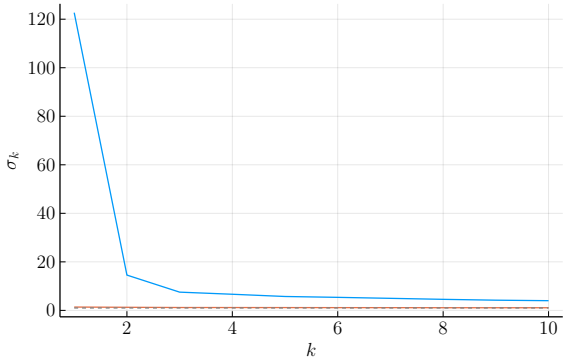
Figure 15: Scatter plots of the representations of the 2-way pixel attack with $\varepsilon = 0.1$ after whitening with the true covariance of the representation of the clean samples. The whitened representations are projected onto their top eight PCA directions. Plots along the diagonal are Gaussian kernel density estimate plots after projecting onto that PCA direction. Off-diagonal plots are scatter plots of the data projected onto the subspace spanned by the corresponding pair of PCA directions. This shows that the poisoned samples (in orange) have split into multiple distinct clusters, resulting in a weakened spectral signature. Nevertheless, whitening enhances the spectral signature and bring the direction of separation to the top principal components.

G. Analysis of QUE scores

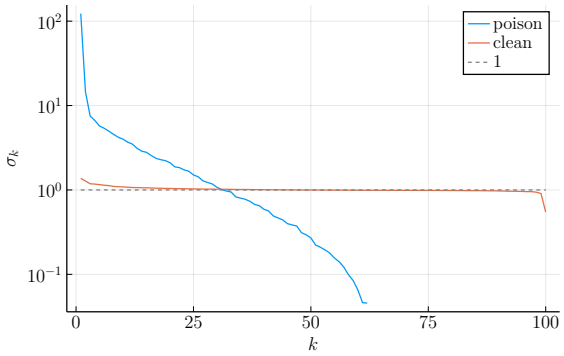
In Section 3.3, we showed that the squared norm scoring $\tau_i^{(0)} = \|\tilde{\boldsymbol{h}}_i\|^2$ and squared projected norm scoring $\tau_i^{(\infty)} = |\langle v, \tilde{\boldsymbol{h}}_i \rangle|^2$ can both fail under certain conditions. Here we will explain those conditions in greater detail.

In our experiments, squared norm scoring $\tau_i^{(0)} = \|\tilde{\boldsymbol{h}}_i\|^2$ fails for the 3-way attack with $\varepsilon = 0.0124$. For this attack, the poisoned representations have high variance along a single direction, and relatively low variance along all other directions, as seen in Fig. 16a. Because there are few poisoned examples relative to clean ones, the resulting spectral signature of the poisoned examples is weak. The directions where the variance of the clean data was amplified, as seen in Fig. 16b, dominate all but one of the directions where the poison had high variance. This can be seen in Fig. 17, where only the top PCA direction, which corresponds to projected norm scoring, is suitable for removing the poisoned examples. Using the squared norm scoring $\tau_i^{(0)} = \|\tilde{\boldsymbol{h}}_i\|^2$ here causes the top PCA direction to be mixed with the less useful directions, diluting its utility as a metric for removing the poison.

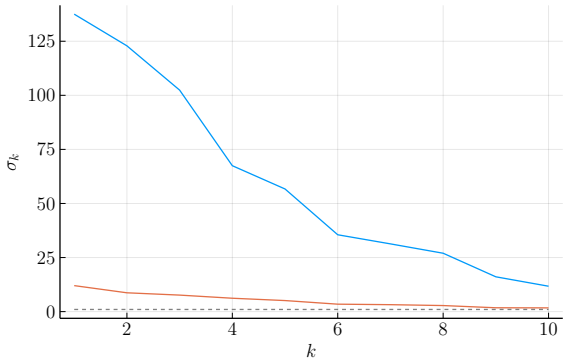
Squared projected norm scoring $\tau_i^{(\infty)} = |\langle v, \tilde{\boldsymbol{h}}_i \rangle|^2$ fails for the 1-way attack with $\varepsilon = 0.1$. Here the spectral signature of the poisoned examples is very strong. The poisoned examples have high variance along many directions, as seen in Fig. 16c. The resulting top PCA direction $\boldsymbol{v}$ is not well aligned with the direction of the separation $\boldsymbol{\mu}(S_{\text{poison}}) - \boldsymbol{\mu}(S_{\text{clean}})$. In fact, the angle between them is $\cos^{-1}(\langle \boldsymbol{v}, \boldsymbol{\mu}(S_{\text{poison}}) - \boldsymbol{\mu}(S_{\text{clean}}) \rangle / \|\boldsymbol{\mu}(S_{\text{poison}}) - \boldsymbol{\mu}(S_{\text{clean}})\|) = 35.7°$. The consequence of this misalignment can be seen in Fig. 18, where it is clear that $\boldsymbol{v}$ does not separate the poisoned examples from the clean ones. On the other hand, squared norm scoring works well here because the poisoned examples have large variance along many directions, which is apparent in Fig. 18.



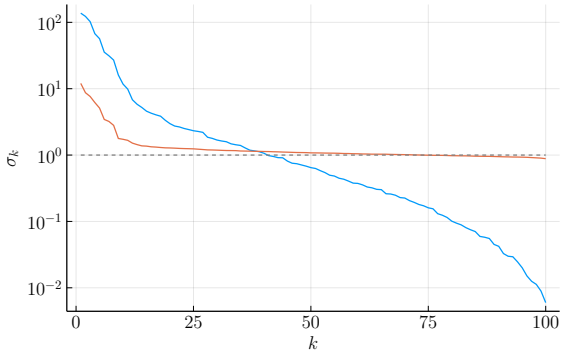
(a) 3-way attack with $\varepsilon = 0.0124$, top 10 singular values



(b) 3-way attack with $\varepsilon = 0.0124$, top 100 singular values



(c) 1-way attack with $\varepsilon = 0.1$, top 10 singular values



(d) 1-way attack with $\varepsilon = 0.1$, top 100 singular values

Figure 16: Plots of the top 10/100 singular values of the covariances of the poison and clean representations after whitening with the robustly estimated covariance in order of decreasing magnitude.

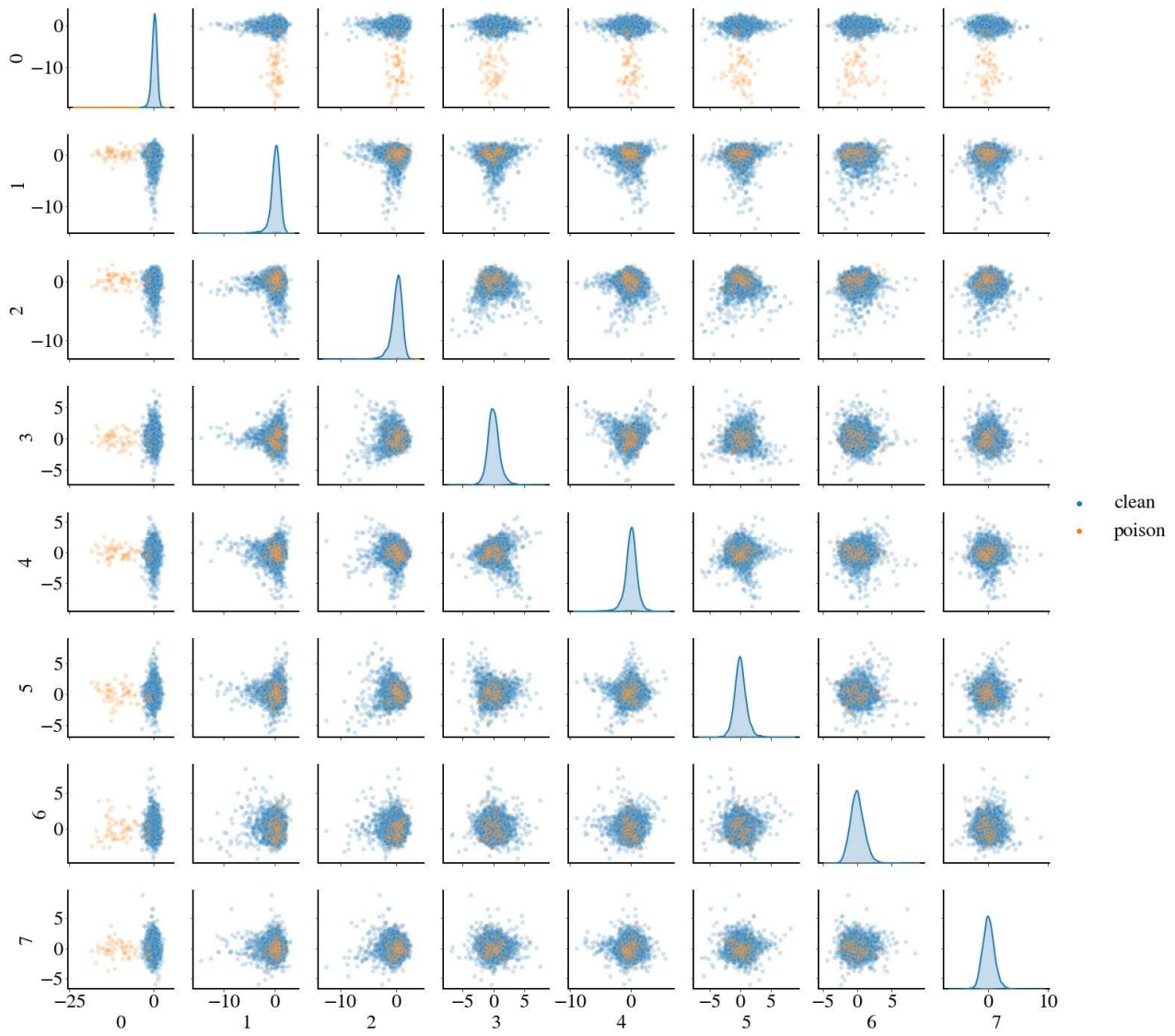


Figure 17: Scatter plots of the representations of the 3-way pixel attack with $\varepsilon = 0.0124$ after robust whitening; whitening the representations of the data with the estimated covariance of the clean samples. The whitened representations are projected onto their top eight PCA directions. Plots along the diagonal are Gaussian kernel density estimate plots after projecting onto that PCA direction. Off-diagonal plots are scatter plots of the data projected onto the subspace spanned by the corresponding pair of PCA directions. This clearly shows that the top PCA direction is aligned with the direction of separation between the poisoned samples (in orange) and clean samples (in blue), hence the squared projected norm scoring works. However, the variance of the poisoned examples are generally smaller, making it hard to distinguish using the squared norm scoring.

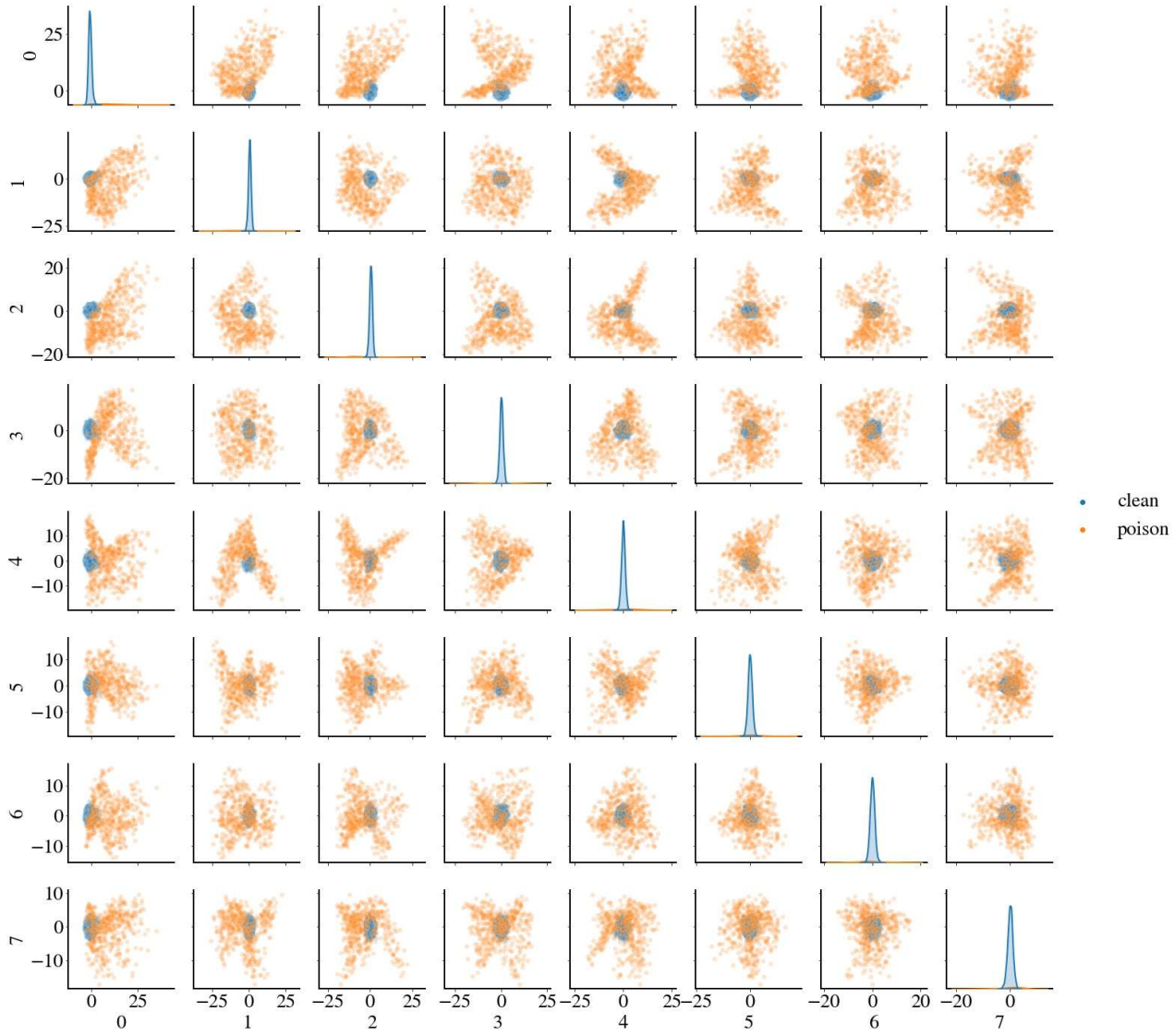


Figure 18: Scatter plots of the representations of the 1-way pixel attack with $\varepsilon = 0.1$ after robust whitening; whitening the representations of the data with the estimated covariance of the clean samples. The whitened representations are projected onto their top eight PCA directions. Plots along the diagonal are Gaussian kernel density estimate plots after projecting onto that PCA direction. Off-diagonal plots are scatter plots of the data projected onto the subspace spanned by the pair of PCA directions. This clearly shows that the top PCA direction is not aligned with the direction of separation between the poisoned samples (in orange) and clean samples (in blue), hence the squared projected norm scoring does not works. However, the variance of the poisoned examples are generally larger, making it easy to distinguish using the squared norm scoring.

H. Sensitivity to number of removed examples

Following (Tran et al., 2018), we choose to remove the $1.5\varepsilon n$ samples with the highest QUE scores from the $(1+\varepsilon)n$ total samples bearing the target label. We show in Fig. 19 that our defence performance is not overly sensitive to this choice. In particular, the fraction of poisoned samples removed does not vary substantially with the total number of removed samples after the first εn samples are removed.

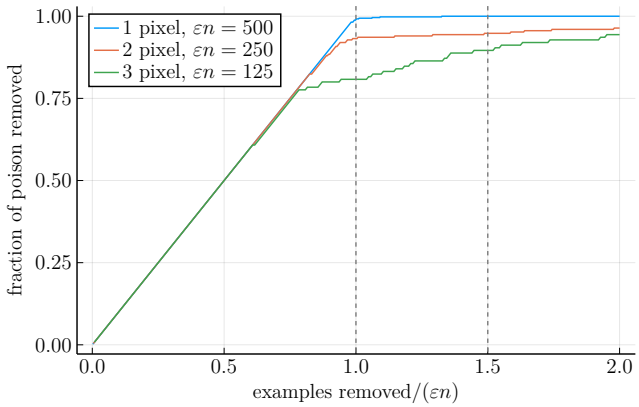


Figure 19: Fraction of all poisoned samples removed vs. the total number of samples removed by SPECTRE, for three pixel attacks featuring spectral signatures of varying strength.