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**
  $C_1, C_2 \leftarrow$ ActivationClustering(S, k) [Algorithm 6]
  $C \leftarrow$ 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		
$\ell_{\rm s}$	$\ell_{\rm t}$	m	εn	$\mathrm{acc_p}$	$\mathrm{acc_{p^*}}$	$\mathrm{p_{rm}}$	$\mathrm{acc'_p}$	$\mathrm{acc'_{p^*}}$	$\mathrm{p_{rm}}$	$\mathrm{acc'_p}$	$\mathrm{acc'_{p^*}}$	$\mathrm{p_{rm}}$	$\mathrm{acc'_p}$	$\mathrm{acc'_{p^*}}$
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 $\mathrm{p_{rm}}$ under m-way pixel attacks for various choices of the source label $\ell_{\rm s}$ and target label $\ell_{\rm t}$. 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	$\mathrm{acc}_{\mathrm{p}^*}$	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*
$$\left| \Pr_{X \in_u S}[L(X) \geq 0] - \Pr_{X \sim G}[L(X) \geq 0] \right| \leq \frac{\varepsilon}{T^2 \log\big(d\log(\frac{d}{\varepsilon\tau})\big)}$$
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 $28\,\mathrm{kW\,h}$ 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.

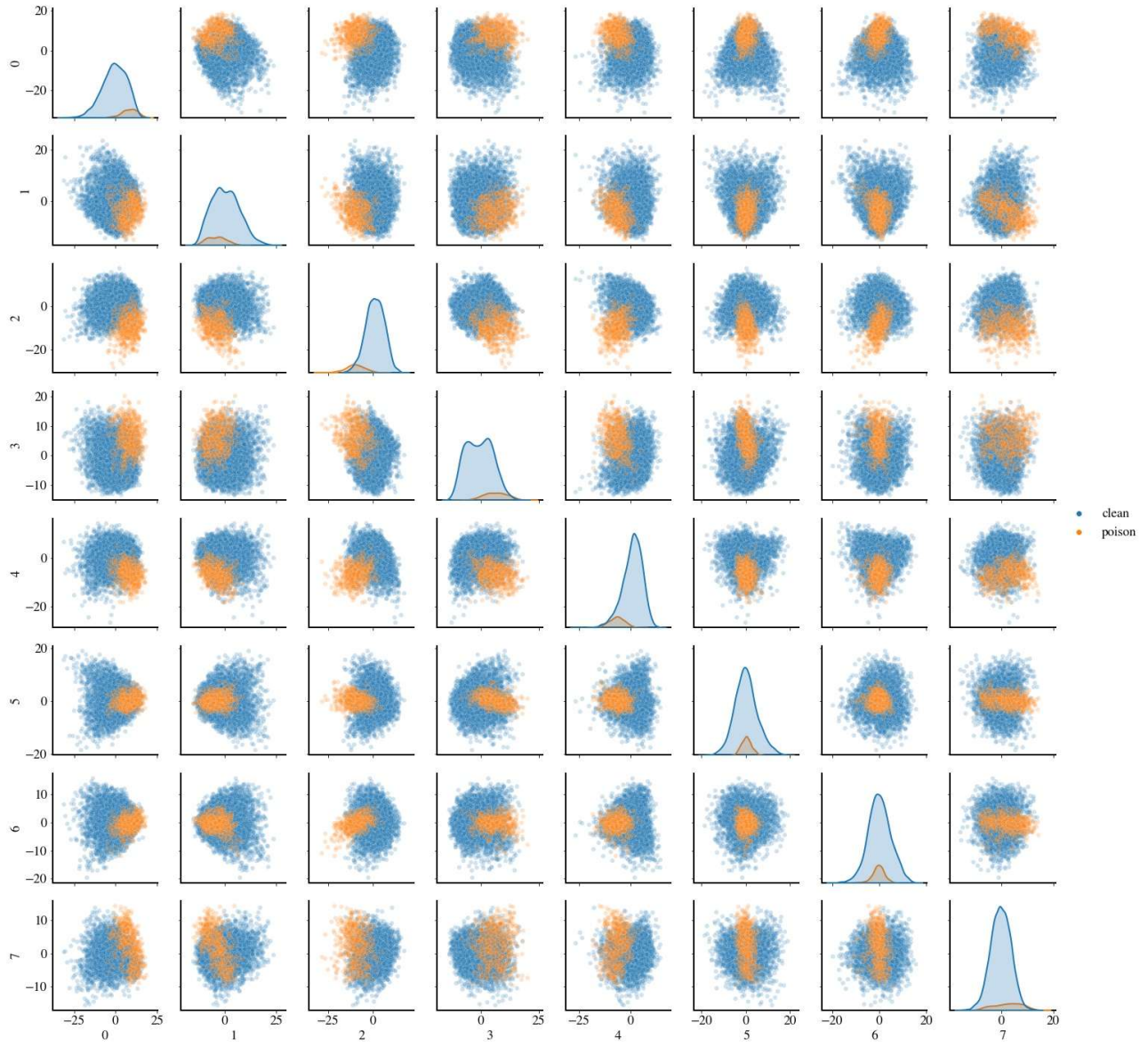


Figure 12: Scatter plots of the representations of the 3-way pixel attack with $\varepsilon = 0.1$ before any whitening. The whitened representations are projected onto their top eight PCA directions. Plots along the diagonal are Gaussian kernel density estimate plots after projecting onto that PCA direction (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.

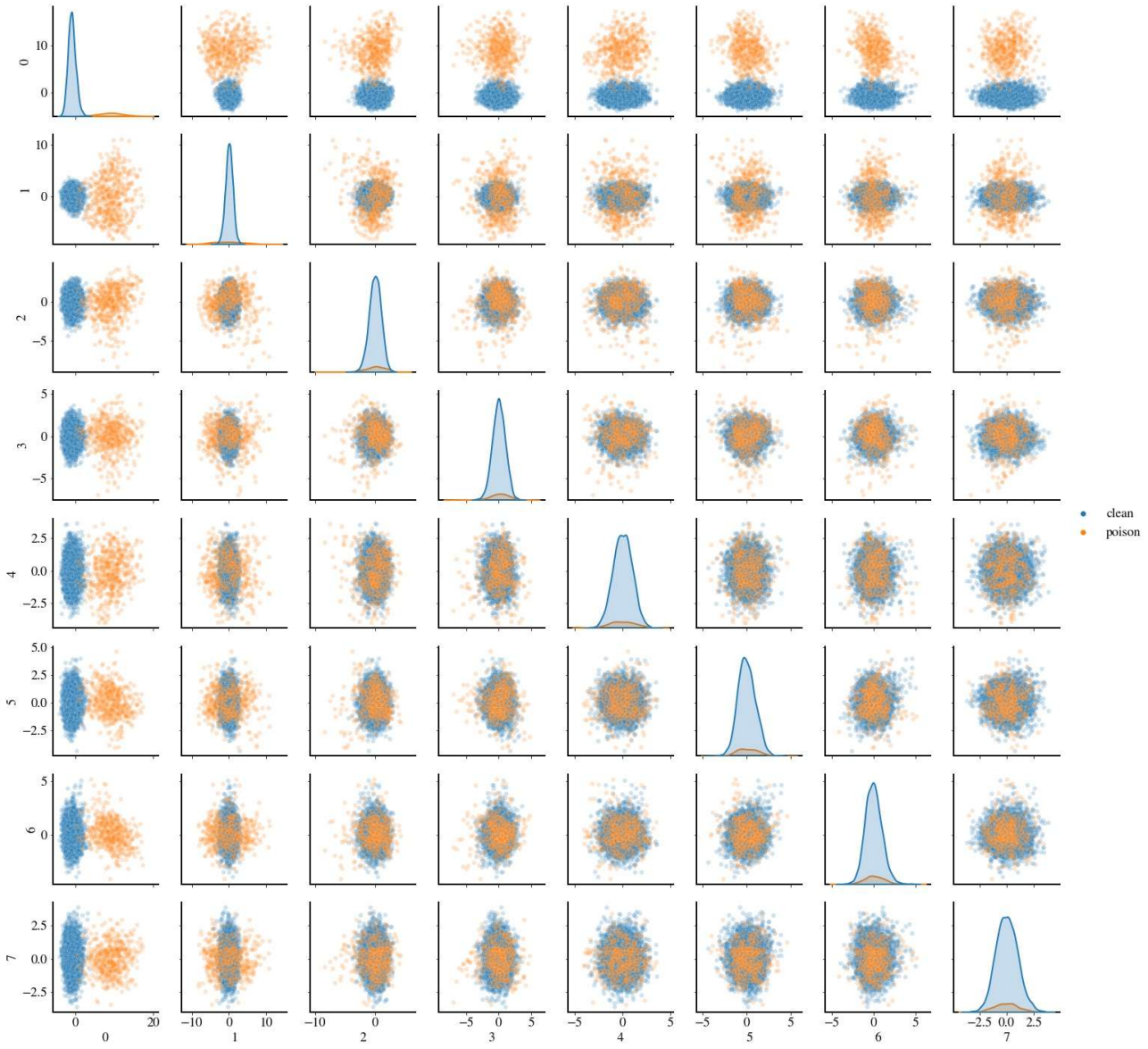


Figure 13: Scatter plots of the representations of the 3-way pixel attack with $\varepsilon = 0.1$ after whitening using the covariance of the clean samples. The whitened representations are projected onto their top eight PCA directions. Plots along the diagonal are Gaussian kernel density estimate plots 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.

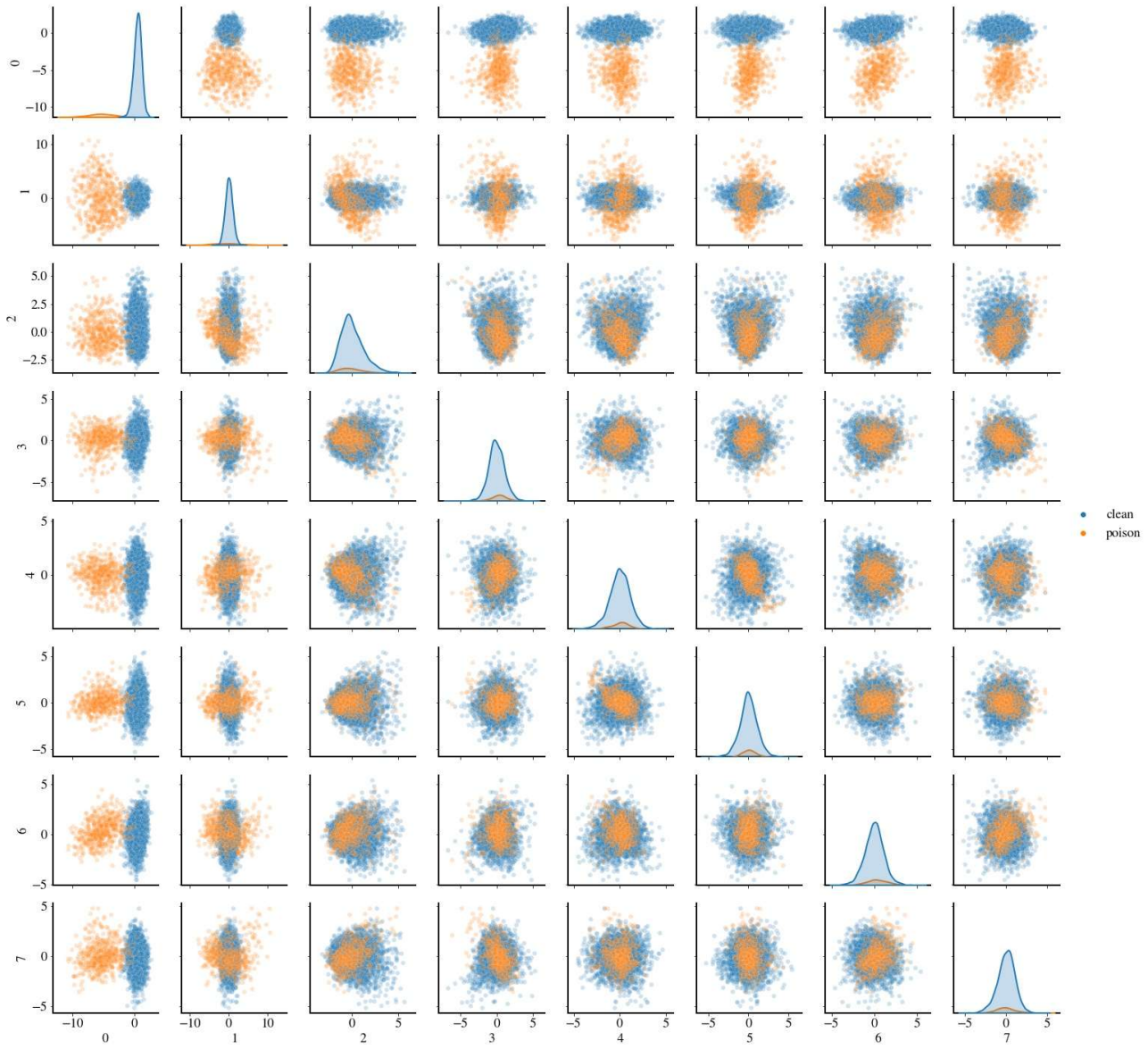


Figure 14: Scatter plots of the representations of the 3-way pixel attack with $\varepsilon = 0.1$ after whitening using the robustly estimated covariance. The whitened representations are projected onto their top eight PCA directions. Plots along the diagonal are Gaussian kernel density estimate plots 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.

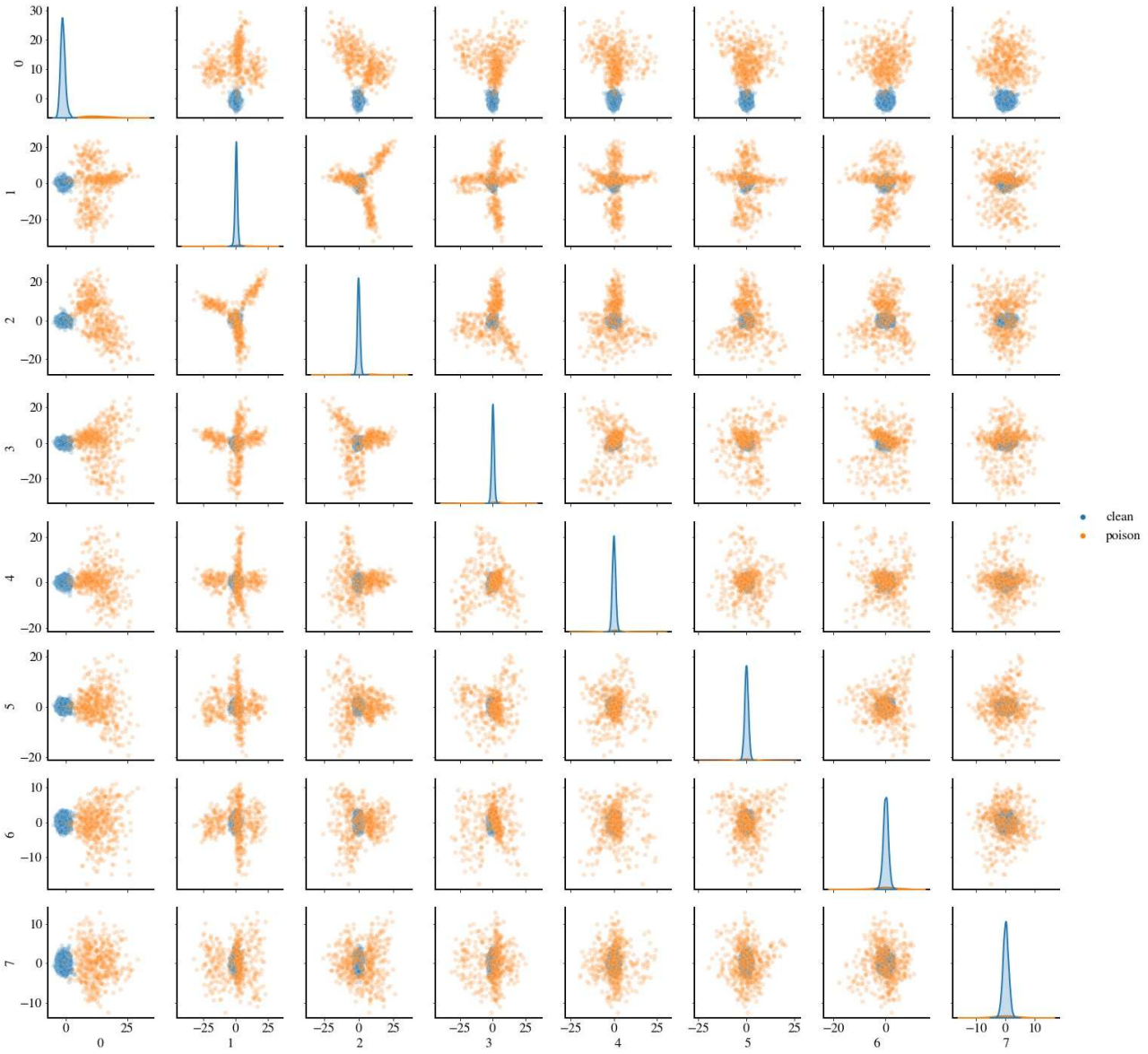


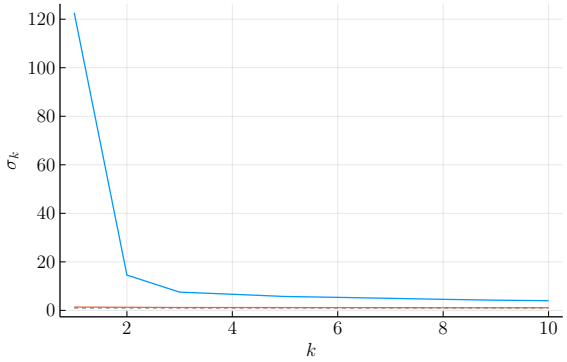
Figure 15: Scatter plots of the representations of the 2-way pixel attack with $\varepsilon = 0.1$ after whitening with the true covariance of the representation of the clean samples. The whitened representations are projected onto their top eight PCA directions. Plots along the diagonal are Gaussian 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

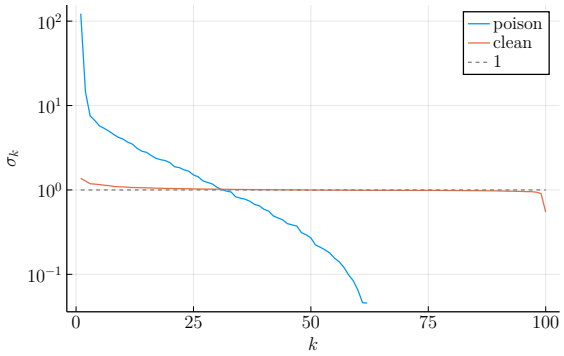
In Section 3.3, we showed that the squared norm scoring $\tau_i^{(0)} = \|\tilde{\boldsymbol{h}}_i\|^2$ and squared projected norm scoring $\tau_i^{(\infty)} = |\langle v, \tilde{\boldsymbol{h}}_i \rangle|^2$ can both fail under certain conditions. Here we will explain those conditions in greater detail.

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.

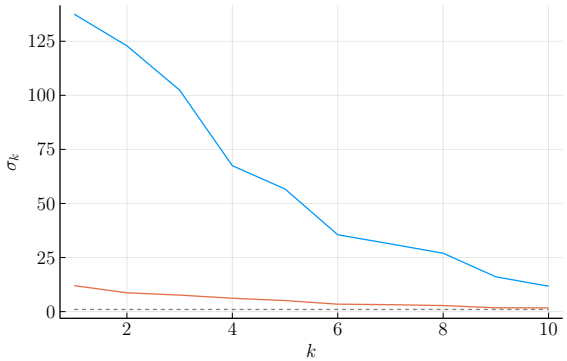
Squared projected norm scoring $\tau_i^{(\infty)} = |\langle v, \tilde{\boldsymbol{h}}_i \rangle|^2$ fails for the 1-way attack with $\varepsilon = 0.1$. Here the spectral signature of the poisoned examples is very strong. The poisoned examples have high variance along many directions, as seen in Fig. 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.



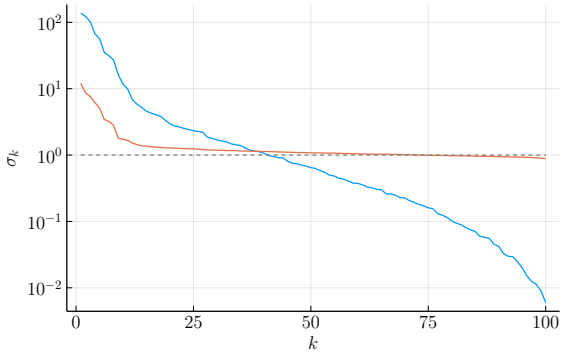
(a) 3-way attack with $\varepsilon = 0.0124$, top 10 singular values



(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.

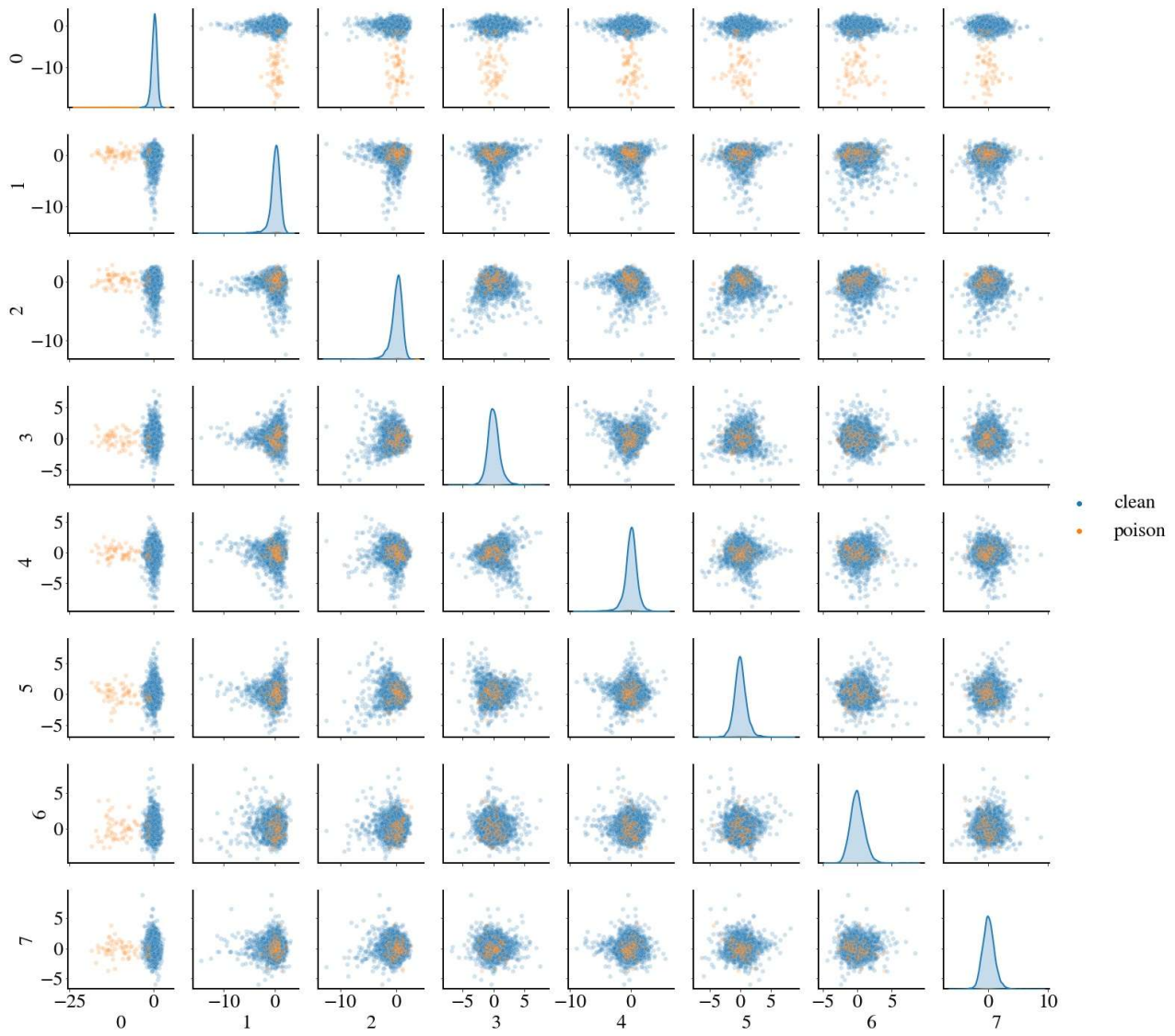


Figure 17: Scatter plots of the representations of the 3-way pixel attack with $\varepsilon = 0.0124$ after robust whitening; whitening the representations of the data with the estimated covariance of the clean samples. The whitened representations are projected onto their top eight PCA directions. Plots 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.

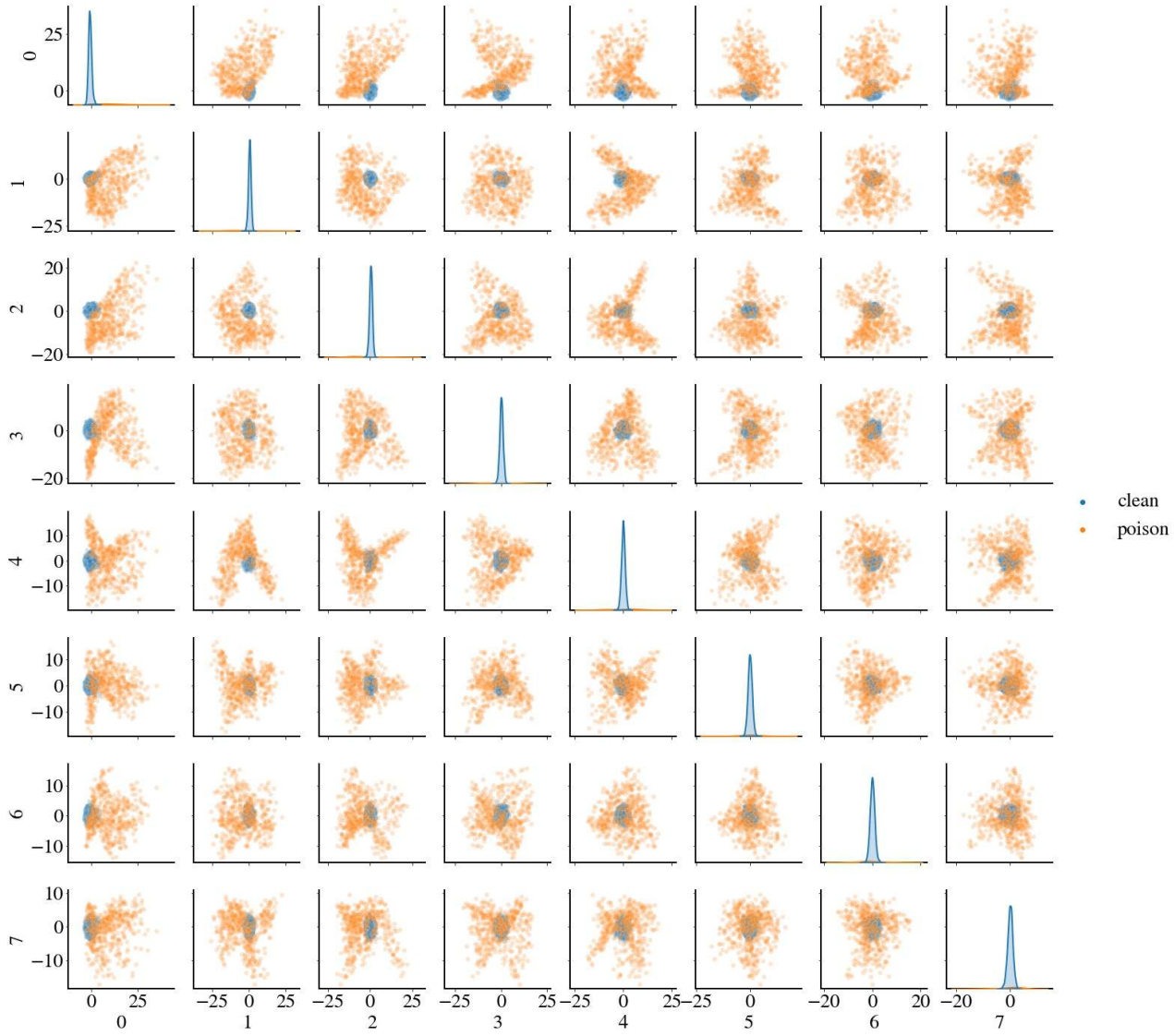


Figure 18: Scatter plots of the representations of the 1-way pixel attack with $\varepsilon = 0.1$ after robust whitening; whitening the representations of the data with the estimated covariance of the clean samples. The whitened representations are projected onto their top eight PCA directions. Plots 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.

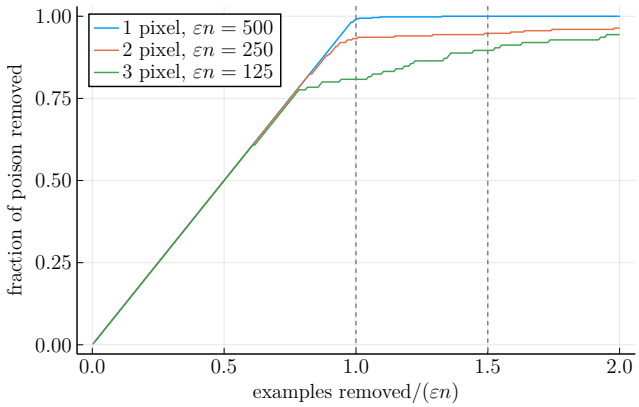


Figure 19: Fraction of all poisoned samples removed vs. the total number of samples removed by SPECTRE, for three pixel attacks featuring spectral signatures of varying strength.