Deep *k*-NN Defense against Clean-label Data Poisoning Attacks

Neehar Peri [*1] **Neal Gupta** [*1] **W. Ronny Huang** [*1] **Liam Fowl** [1] **Chen Zhu** [1] **Soheil Feizi** [1] **Tom Goldstein** [1] **John P. Dickerson** [1]

Abstract

Targeted clean-label data poisoning is a type of adversarial attack on machine learning systems in which an adversary injects a few correctly-labeled, minimally-perturbed samples into the training data, causing a model to misclassify a particular test sample during inference. Although defenses have been proposed for general poisoning attacks, no reliable defense for clean-label attacks has been demonstrated, despite the attacks' effectiveness and realistic applications. In this work, we propose a simple, yet highly-effective Deep *k*-NN defense against both feature collision and convex polytope clean-label attacks on the CIFAR-10 dataset. We demonstrate that our proposed strategy is able to detect over 99% of poisoned examples in both attacks and remove them without compromising model performance. Additionally, through ablation studies, we discover simple guidelines for selecting the value of *k* as well as for implementing the Deep *k*-NN defense on real-world datasets with class imbalance. Our proposed defense shows that current clean-label poisoning attack strategies can be annulled, and serves as a strong yet simple-to-implement baseline defense to test future clean-label poisoning attacks. Our code is available on GitHub.

1. Introduction

Machine-learning-based systems are increasingly being deployed in settings with high societal impact, including hate speech detection on social networks (Rizoiu et al., 2019), autonomous driving (Chen et al., 2017a), biometric-based applications (Sun et al., 2014), and malware detection (Pascanu et al., 2015). In these applications, a system's robustness to not only noise, but also *adversarial manipulation* is paramount. With an increasing number of machine learning systems trained on data sourced from public and semi-public places such as social networks, collaboratively-edited forums, and multimedia posting services, adversaries can strategically inject training data to manipulate or degrade system performance.

Data poisoning attacks on neural networks occur at training time, wherein an adversary places specially-constructed *poisoned examples* into the training data with the intention of manipulating the behavior of the system at test time. Recent work on data poisoning has focused on either (i) an attacker generating a small fraction of training inputs to degrade overall model performance, or (ii) a defender aiming to detect or otherwise mitigate the impact of that attack. In this paper, we focus on *clean-label* data poisoning (Shafahi et al., 2018), where an attacker injects a small number of *correctly labeled*, minimally perturbed samples into the training data. In contrast with traditional data poisoning, these samples are crafted to cause a model to misclassify a particular *target* test sample during inference. These attacks are plausible in a wide range of applications, as they do not require the attacker to have control over the labeling function. Many large scale data sets are automatically scraped from the internet without direct supervision, so an adversary need only share their poisoned data online.

Our contribution: In this paper, we initiate the study of defending against *clean-label* poisoning attacks on neural networks by considering feature collision attacks (Shafahi et al., 2018) and convex polytope attacks (Zhu et al., 2019) on the CIFAR-10 dataset. Although poison examples are not easily detected by human annotators, we exploit the property that adversarial examples have different feature distributions than their clean counterparts in higher layers of the network, and that those features often lie near the distribution of the target class. This intuition lends itself to a defense based on k nearest neighbors in feature space, in which the poison examples are detected and removed *prior* to training. Further, the parameter k yields a natural lever for trading off between the number of undetected poisons and number of discarded clean images when filtering the training set.

Our contributions can be outlined as follows.

- We propose a novel Deep *k*-NN defense for clean-label

[*]Equal contribution [1]Center for Machine Learning, University of Maryland - College Park. Correspondence to: W. Ronny Huang <wrhuang@umd.edu>, John Dickerson <john@cs.umd.edu>.

poisoning attacks. We evaluate it against state-of-the-art clean-label data poisoning attacks, using a slate of architectures and show that our proposed strategy detects 99% of the poison instances without degrading overall performance.

- We reimplement a set of general data poisoning defenses (Koh et al., 2018), including L_2-Norm Outliers, One-Class SVMs, Random Point Eviction, and Adversarial Training as baselines and show that our proposed Deep k-NN defense is more robust at detection of poisons in precision, recall, F1 score, Matthews Correlation Coefficient metrics, as well as actual defense success rate in the trained victim models.

- From the insights of two ablation studies, we assemble guidelines for implementing Deep k-NN in practice. First we provide instructions for picking an appropriate value for k. Second, we provide a protocol for using the Deep k-NN defense when class imbalance exists in the training set.

2. Overview of Clean-label Data Poisoning

We briefly describe the how clean-label data poisoning works and the intuition behind a neighborhood conformity defense. Figure 1 shows the feature space representation, i.e. the representations in the penultimate layer of the network, for a targeted poisoning attack that causes a chosen target airplane image (feature representation shown as the dark gray triangle) to be misclassified as a frog during inference. To accomplish this, poison frog images (feature representation shown as dark orange circles) are perturbed to surround the target airplane in feature space. After training on this poisoned data set, the model changes its decision boundary between the two classes in order to accommodate the poison frogs, enveloping them onto the side of the frogs. Inadvertently, the nearby target airplane is also placed on the the side of the frogs, leading to misclassification.

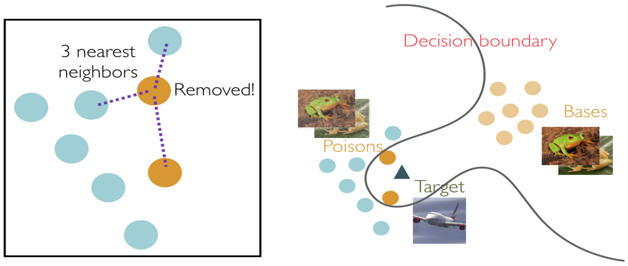


Figure 1. Proposed Deep k-NN defense ($k = 3$) correctly removing a poisoned example by comparing the class labels of poison with its k neighbors. Since a majority of the k points surrounding the poison do not share the same class label as the poison, it is removed.

Under the *feature collision* attack (Shafahi et al., 2018), the perturbations are optimized so as to minimize the poison images' distance to the target image in feature space,

$$\boldsymbol{x}_p = \underset{\boldsymbol{x}}{\arg\min}\ |\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}_t)|_2^2 + |\boldsymbol{x} - \boldsymbol{x}_b|_2^2,$$

where $\boldsymbol{x}_p$, $\boldsymbol{x}_b$, $\boldsymbol{x}_t$ are the poison, base, and target images, respectively, and ϕ is a feature extractor that propagates input images to the penultimate layer of the network. Alternatively, under the *convex polytope* attack (Zhu et al., 2019), poisoned data points are optimized to form a convex hull of poisons around the target via a more sophisticated loss function. In both cases nonetheless, models fine-tuned on the poisoned dataset will have their decision boundaries adversarially warped and classify the targeted airplane image as a frog at inference time. Though the optimization causes a noticeable change in the feature representations of the images, the poison frogs are perturbed under some small ℓ_2 or ℓ_∞ constraint so that they still appear to be frogs to a human observer.

2.1. Intuition behind Deep *k*-NN Defense

As seen in Figure 1, poisons are surrounded by feature representations of the target class rather than of the base class. For instance, when $k = 3$ and $n_{poison} = 2$, each poison will almost always have a plurality of its neighbors as a non-poison in the target class. Since the plurality label of a poisons neighbors does not match the label of the poison itself, the poison can be removed from the dataset or simply not used for training. More generally, if $k > 2n_{poison}$, then we would expect the poisons to be outvoted by members of the target class and be filtered from the training set.

Note that by setting $k > 2n_{poison}$, the poisons' label cannot be the majority, but may still be the plurality, or mode, of the Deep k-NN set if the nearest neighbors of the current point are in multiple classes. Empirically, however, we do not observe this to be the case. Extracted features tend to be well-clustered by class; thus there are usually only 2 unique classes in the Deep k-NN neighborhood, base class and target class, with the target class being larger. Therefore to succeed the victim needs only to set a large value of k without needing to know exactly how many poisons there are *a-priori*. We further elucidate on the effect of k in Section 5.

3. Defenses against Clean-Label Poisoning

In this section, we formally introduce the Deep k-NN defense as well as a set of other baseline defenses against clean-label targeted poisoning attacks. We compare the effectiveness of each defense against both feature collision attacks and convex polytope attacks in Section 4.

3.1. Notation

We use $\boldsymbol{x}_t$ to denote the input space representation of the target image that an adversary tries to misclassify. The target has true label l_t but the attacker seeks to misclassify it as having label l_b. We use $\boldsymbol{x}_b$ to denote a base image having label l_b that is used to build a poison after optimization. We use $\boldsymbol{x}_w$ to denote a base image watermarked with a target image, that is $\gamma \cdot \boldsymbol{x}_t + (1-\gamma) \cdot \boldsymbol{x}_b$. To a human observer this image will retain the label l_b when γ is sufficiently low. We use $\phi(\boldsymbol{x})$ to denote the activations of the penultimate layer of a neural network. We will refer to this as the *feature layer* or *feature space* and $\phi(\boldsymbol{x})$ as *features* of $\boldsymbol{x}$.

3.2. Deep *k*-NN Defense

For each data point in the training set, the Deep *k*-NN defense takes the plurality vote amongst the labels of that point's k nearest neighbors in feature space. If the point's own label is not the mode amongst labels of its k nearest neighbors, the point is flagged as anomalous, and is not used when training the model. We use Euclidean distance to measure the distance between data points in feature space. See Algorithm 1.

Algorithm 1 Deep *k*-NN Defense

```
Result: Filtered training set $X^{train'}$
Let $S_k(x^{(i)})$ denote a set of $k$ points such that for all points
  $x^{(j)}$ inside the set and points $x^{(l)}$ outside the set, $|\phi(x^{(l)}) - \phi(x^{(i)})|_2 \geq |\phi(x^{(j)}) - \phi(x^{(i)})|_2$
  $X^{train'} \leftarrow \{\}$
  for Data points $\boldsymbol{x}^{(i)} \in X^{train}$ do
      Let l denote the label of $x^{(i)}$ and let $l(S_k(x^{(i)}))$ denote
        the labels of the points in $S_k(x^{(i)})$
        if $l \in mode(l(S_k(x^{(i)})))$ then
            $X^{train'} \leftarrow X^{train'} \cup \{x^{(i)}\}$;
      else
            Omit $x^{(i)}$ from $X^{train'}$;
      end
end
```

3.3. L2-Norm Outlier Defense

The L2 norm outlier defense removes an $\epsilon > 0$ fraction of points that are farthest in feature space from the centroids of their classes. For each class of label $l \in \mathcal{L}$, with size $s_l = |x^{(j)}$ s.t. $l(j) = l|$, we compute the centroid c_l as

$$c_l = \frac{1}{s_l} \sum_{x^{(j)} s.t. l(j)=l} \phi(x^{(j)})$$

and remove $\lfloor \epsilon s_l \rfloor$ points maximizing $|\phi(x^{(j)}) - c_l|_2$.

The L2 norm defense relies on the position of the centroid to filter outliers. However, the position of the centroid itself is prone to data poisoning if the per-class data size is small. This defense is adapted from traditional poison defenses not specific to neural networks (Koh et al., 2018).

3.4. One-Class SVM Defense

The one-class SVM defense examines the deep features of each class in isolation by applying the one-class SVM algorithm (Schölkopf et al., 2001) to identify outliers in feature space for each label in the training set. It utilizes a radial basis kernel and is calibrated to use a value $\nu = 0.01$ to identify anomalous points.

3.5. Random Point Eviction Defense

The random point eviction defense is a simple experimental control. It filters out a random subset of all training data. We remove 1% of our training data for the feature collision attack and 10% of our training data on the convex polytope attack. If the poisoning attack is sensitive to poisons being removed, the random defense may be successful, at the cost of losing a proportionate amount of the unpoisoned training data.

3.6. Adversarial Training Defense

Thus far, we have only considered defenses which filter out examples prior to training. We consider here another defense strategy that does not involve filtering, but rather involves an alternative victim training procedure. Adversarial training, used often to harden networks against evasion attacks (Goodfellow et al., 2015; Madry et al., 2017), has been shown to produce neural network feature extractors which are less sensitive to weak features such as norm-bounded adversarial patterns (Ilyas et al., 2019). We explore here whether a victim's use of an adversarially trained feature extractor would yield features that are robust to clean-label data poisoning. Instead of the conventional loss over the training set, adversarial training aims to optimize

$$\min_{\theta} \mathcal{L}_\theta(X + \delta^*), \text{where } \delta^* = \underset{\delta < \epsilon}{\text{argmax}} \, \mathcal{L}_\theta(X + \delta),$$

where θ, X, and δ are the weights, training input, and adversarial perturbations, respectively, and $\mathcal{L}_\theta$ is some training loss, i.e., cross-entropy. In experiments, we perform adversarial training following the standard procedure in Madry et al. (2017), using an ℓ_∞ PGD adversary of 20 steps and $\epsilon = 8$.

4. Evaluation

In this section, we evaluate the effectiveness of our Deep *k*-NN defense and baseline defenses against the feature collision (Shafahi et al., 2018) and convex polytope (Zhu et al.,

2019) attacks on the CIFAR-10 dataset (Krizhevsky et al., 2009). All model architectures, data splits, and hyperparameters are taken directly from the evaluation setups used in Shafahi et al. (2018) and Zhu et al. (2019). We define the defense success rate as the number of times the poisoning attack fails to cause the target example to be misclassified, divided by the number of attempts. We only consider sets of poisons that lead to successful attacks in the undefended case so by definition the undefended defense success rate is 0%.

4.1. Defense against Feature Collision Attack

4.1.1. Attack Procedure

We randomly select 50 images in the base class. For each base image with input representation $\boldsymbol{x}_b$, we compute the watermark base $\boldsymbol{x}_w \leftarrow \gamma \cdot \boldsymbol{x}_t + (1-\gamma) \cdot \boldsymbol{x}_b$, then optimize p with initial value $\boldsymbol{w}$ using a forward-backward splitting procedure to solve

$$\boldsymbol{x}_p = \arg\min_{\boldsymbol{x}} |\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}_t)|_2^2 + \beta|\boldsymbol{x} - \boldsymbol{x}_w|_2^2$$

The hyperparameter β is fixed at 0.1. The resulting poisons $\boldsymbol{x}_p$ are both close to the target image $\boldsymbol{x}_t$ in feature space, and close to the watermarked input $\boldsymbol{x}_w$ in image space. To ensure statistical significance, we craft 16 of these collections of 50 poisons and evaluate each collection independently.

4.1.2. Defense Procedure

As in the original setup (Shafahi et al., 2018), we first train a modified AlexNet to convergence using only clean data. Next we apply our defenses on the set of clean data plus poisons to obtain a filtered dataset. That filtered dataset is then used to fine tune the pretrained model over 10 epoch with a batch size of 128. We evaluate the performance of all defenses described in Section 3 against collections of 50 poisons that successfully cause a targeted misclassification.

4.1.3. Results

The Deep k-NN defense with $k = 5000$, as seen in Table 1, successfully identifies all but one poison across multiple attacks, while filtering just 0.6% of the clean images from the training set. As a result, after victim training, models defended by Deep k-NN have defense success rates of 100%. In contrast, the $L2$-norm defense only identifies roughly half the feature collision poisons using $\epsilon = 0.01$. Both the One-Class SVM and the Random Point Eviction defenses are unable to detect a majority of the feature collision poisons.

4.2. Defense against Convex Polytope Attack

4.2.1. Attack Procedure

Following the procedure in Zhu et al. (2019), the CIFAR-10 dataset is split into 48000 images for pretraining, and 500 images for fine-tuning. The poison base images are taken from the remaining split of 1500 images. Since the attacker does not know the victim model parameters, they first pretrain their own model to convergence using the same subset of 48000 CIFAR-10 images used for pretraining. Next, an adversary uses this surrogate model to craft 5 poisons using the convex polytope method. To ensure statistical significance, 102 collections of 5 poisons are crafted.

When crafting convex polytope poisons, multiple surrogate models with different architectures are ensembled, so that the generated poisons generalize to victim architectures that the poisons were not crafted on. Our results are based on eight architectures: two of which are not used in crafting the poisons (black box setting), and six which use random initialization (grey box setting). The grey-box architectures are DPN92 (Chen et al., 2017b), GoogLeNet (Szegedy et al., 2015), MobileNetV2 (Sandler et al., 2018), ResNet50 (He et al., 2016), ResNeXT29-2x64d (Xie et al., 2017), and SENet18 (Hu et al., 2018), while the black-box architectures are DenseNet121 (Huang et al., 2017) and ResNet18 (He et al., 2016).

4.2.2. Defense Procedure

The victim model is first pretrained to convergence using a random initialization unknown to the attacker on the 48000 pretraining images from CIFAR-10[1]. Our defenses are applied to the 500 fine-tuning images plus poisons to obtain a filtered fine-tuning set[2]. Finally, this filtered dataset is used to fine-tune the victim model.

Again, the performance of all defenses is reported only on collections of poisons that lead to a successful attack in the undefended case. Since the attacker did not have access to the victim architecture or model parameters during crafting of the poisons, the defenses are evaluated independently for each individual victim architecture.

4.2.3. Overall Results

The aggregate results of each defense strategy on all 8 architectures are shown in Table 2. Both the Deep k-NN and $L2$-Norm defense filter out nearly all poisons, while incorrectly removing 4.3% and 9.1% of the clean training examples, respectively. Compared to feature collision poisons, convex polytope poisons trigger more false positive

[1] We use conventional training loss for all except the adversarial training defense.

[2] There is no filtering in adversarial training.

Table 1. Comparing the effectiveness of baseline defenses aggregated for all model architectures in Feature Collision Attack

Defense Strategy	Poisons Removed	Clean Images Removed (%)	Defense Success Rate (%)	CIFAR-10 Test Accuracy (%)
Deep k-NN $(k = 5000)$	**799/800**	**0.6**	**100.0**	**74.6**
$L2$-Norm Outliers	395/800	1.0	50.0	**74.6**
One-class SVM	168/800	1.0	37.5	74.5
Random Point Eviction	84/800	10.0	12.5	74.5

Table 2. Comparing the effectiveness of baseline defenses aggregated for all model architectures in Convex Polytope Attack

Defense Strategy	Poisons Removed	Clean Images Removed (%)	Defense Success Rate (%)	CIFAR-10 Test Accuracy (%)
Deep k-NN $(k = 50)$	**510/510**	**4.3**	**100.0**	**93.9**
$L2$-Norm Outliers	509/510	9.1	99.0	93.4
One-class SVM	114/510	7.1	29.9	91.7
Random Point Eviction	47/510	10.0	33.2	91.3
Adversarial Training	-	-	98.6	85.2

detections (i.e. clean images removed) across all defense methods, leading to fewer remaining clean examples and reduced test accuracy. Surprisingly, the $L2$-Norm defense is much better able to detect convex polytope poisons compared to feature collision poisons; it detects almost as many as Deep k-NN . However, it has a lower specificity because it removes more clean images, resulting in half-percent lower test accuracy.

These results are broken down for each victim architecture in Figure 2. The Deep k-NN attack is successful on all architectures with perfect defense success rate. $L2$-norm Outliers and Adversarial Training perform almost as well. Other strategies largely fail to be a viable defense.

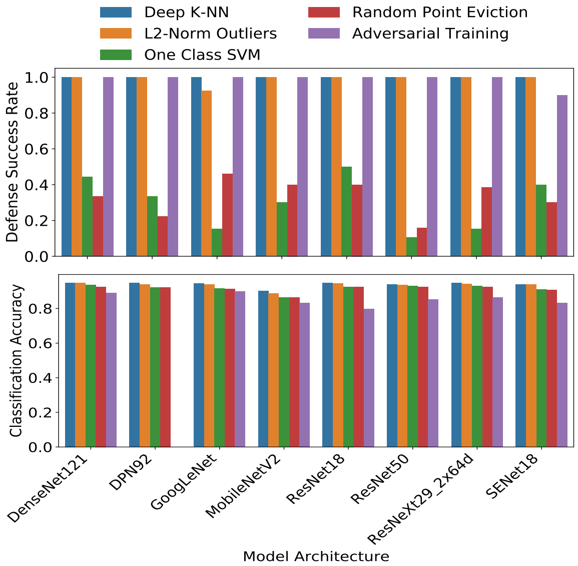


Figure 2. The Deep k-NN Defense is model-agnostic in its ability to defend against convex polytope attacks. It outperforms the all other defenses in terms of defense success rate while also being the best at keeping test classification accuracy high.

4.2.4. Results on Adversarial Training

We evaluate the effectiveness of adversarial training on the Convex Polytope-crafted poisons. In Table 2 and Figure 2, adversarially trained feature extractors—trained naively to provide resistance against only evasion attacks—do in fact help mitigate poisoning attacks as well. To our knowledge, this is the first time adversarial training has been shown to provide resistance against data poisoning, i.e. training time, attacks and is a direction for future work. The defense however significantly hurts test set accuracy (as is common for adversarially trained networks), which drops to 85% on average, compared with 94% on the same architectures without adversarial training. In scenarios when adversarial training for evasion attack robustness is not required, such as in situations when adversaries cannot control test time inputs, the Deep k-NN defense provides the poisoning resistance without the burden of decreased generalization performance.

4.2.5. Additional Metrics

We now take a closer look at how well Deep k-NN filters out the poison examples while keeping the clean examples. We display the metrics of precision, recall and F1 score in Table 3. Note here that precision and recall are with respect to the detection of poisons. In other words, precision is defined as the number of poisons detected correctly over the total number of points marked as anomalous. Deep k-NN performs better than the other baselines in all metrics. All defenses are weighted more toward high recall than high precision due to the severity of having a poisoned model compared to a slightly less performant model.

The use of traditional metrics such as precision, recall, and F1 score report the effectiveness of Deep k-NN at detecting poisons, but only indirectly reveals how well it performs at leaving alone the non-poison data points, and by extension preserving the test accuracy. For example, a low precision (i.e. high false positive rate) could result in a sizeable portion of the clean data being removed if the number of poisons takes up a sizeable fraction of the dataset (e.g. 10%). Alternatively, a low precision may not cause much harm if the number of poisons is a negligible portion of the dataset. Since it is important to measure how well Deep k-NN detects poisons *and* keeps clean data, we report the Matthew's Correlation Coefficient (MCC) as a balanced metric which takes both into account. In short, MCC accounts for both false positives and false negatives, but does not overemphasize one or the other when the two classes are of very different sizes, as is often the case when the classes are poison versus clean data. A perfect defense has an MCC score of 1 while the worst possible defense has a score of -1. A random point eviction defense has an average MCC of 0. Figure 3 shows that Deep k-NN achieves a correlation value of 0.53, with the errors contributed primarily from false positives (marking a clean data point as anomalous). We further explore the Deep k-NN detector performance with respect to the MCC in our ablation study (Section 5).

4.3. Feature Space Visualization

The favorable results of Deep k-NN defense also afford us an opportunity to understand anomaly detection in deep networks more generally via observing the effects in feature representations. A feature space visualization of the penultimate layer of the network is shown in Figure 3, with both filtered poisons and non-poisons displayed. Specifically, Figure 3 shows a projected visualization in the feature space of the fine tuning set in the target (blue) and base (green) classes. Following the projection scheme used in Shafahi et al. (2018), where the x-axis is the direction along the line connecting the centroids of the target and base class features and the y-axis is the component of the parameter vector (i.e. decision boundary) orthogonal to the between-centroids vector, the deep features of the DPN92 network are projected into a two-dimensional plane. The "x" markers denote poisons that are filtered out by the defense and would have otherwise almost formed a convex polytope around the target (blue triangle). The Deep k-NN acts with high specificity: all the poisons are filtered, while only 2 outlying clean points in the target class (not shown) are also filtered. No points in the base class are filtered. Additional visualizations for the adversarially trained feature extractors are shown in Section A.

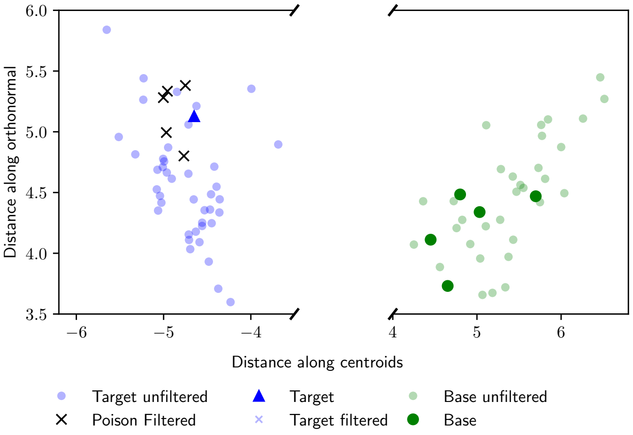


Figure 3. Feature space (i.e. penultimate layer) visualization of the Deep k-NN defense in action during a Convex Polytope Attack on the DPN92 architecture.

5. Ablation Studies and Best Practices

We now turn to ablation studies to gain insight into best practices for using the Deep k-NN defense under realistic situations. All results are reported on the convex polytope attack for CIFAR-10 as described in Zhu et al. (2019) on all 8 architectures discussed previously. We specifically focus on the convex polytope attack method since it is shown to act as a stronger poison on black-box threat models, and study the transfer learning case to mimic the common practice of using pre-trained feature-extractors trained on large datasets.

We again closely mimick the setup in Zhu et al. (2019) using the first 4800 images in each class to train a model from scratch and then using the next 50 images of each class (making a fine-tuning set size of 500) to fine-tune the model. The Adam optimizer with a learning rate of 0.1 is used. In both studies, we assign frogs as the target class and ships as the base class. The first 5 ship images from the fine-tuning set are replaced with the 5 poisoned ships. Each set of 5 poisoned ships has an associated target frog image that is neither in the training nor fine-tune set. We use the standard CIFAR-10 test split to measure test accuracy.

5.1. Choosing a Value of k

In our first study, we vary the value of k used in the Deep k-NN defense. Since dataset sizes vary, as well as the number of classes, we normalize k against the number of data points *per class*. Specifically, we measure all metrics against a normalized-k ratio, such that normalized-$k = k/N$ where k is the number of nearest neighbors considered by the Deep k-NN and N is the maximum number of examples for any class in the fine-tune set.

As seen in Figure 4 (top left and middle left), the defense success rate and MCC begin to reach maximum levels at

Table 3. Deep k-NN defense outperforms other defense strategies on the Convex Polytope Attack on metrics of precision, recall, F1 Score, and Matthew's Correlation Coefficient

Defense Strategy	Precision	Recall	F1 Score	MCC
Deep k-NN ($k = 50$)	**0.34**	**1.00**	**0.44**	**0.53**
$L2$-Norm Outliers	0.10	0.99	0.18	0.30
One-class SVM	0.03	0.22	0.09	0.06
Random filtering	0.01	0.10	0.04	0

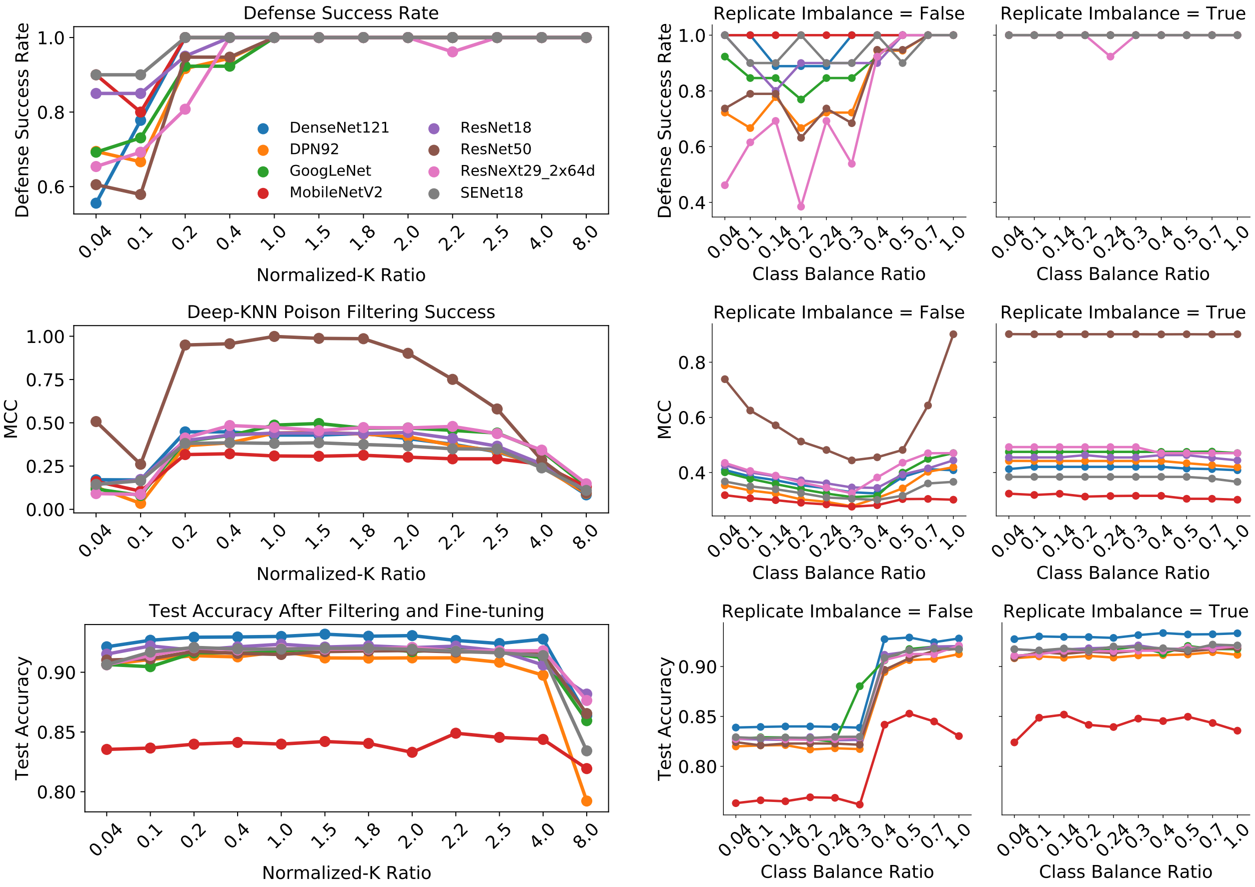


Figure 4. Ablation studies on the effect of k (left) and class imbalance (right). (Top Left) Defense success rate increases to 100% for all models as normalized-k ratio increases beyond 1.0 for all architectures. (Middle Left) Matthew's correlation coefficient is highest for all models when normalized-k ratio is between 0.4 and 2.0. (Bottom Left) Accuracy on the CIFAR-10 test split drops as normalized-k value increases beyond 4 times the number of examples per class. (Top Right) Defense and performance metrics under class imbalance. Defense success rate is stabilized when the target class training examples are first replicated to match the size of other classes. (Middle Right) Matthews correlation coefficient is also less dependent on the size of the target class when data replication is on. (Bottom Right) Test accuracy is highest when replicating the target examples to match the size of other classes.

normalized-$k = 0.2$, corresponding to an (unnormalized) k of twice the number of poisons, $k = 10 = 2n_{poison}$. This confirms our intuition in Section 2.1: when $k > 2n_{poison}$, poisons will be marked anomalous since the poison class cannot be the majority label in the neighborhood and is unlikely to be the plurality because the neighborhood usually only contains two unique classes.

Of course, the victim must set a value of k without knowledge of the number of poisons employed by the attacker. Fortunately we observe that defense success rate remains at 100% as the normalized-k ratio increases beyond $k = 0.2$. Specifically, we see that after a normalized-k value greater than 1.0 ($k = 50$), i.e. the situation where Deep k-NN considers more neighbors than the per-class number of examples, the convex polytope attack is ineffective on all models. There are limits though. Despite successfully detecting all the poisons, too large of a k could lead to adverse effects on model test performance if too much clean data is removed (i.e. too many false positives).

To take both positives and negatives into account, we again invoke the MCC metric in Figure 4 (middle left) to measure the trade-off between detecting poisoned images and removing clean images. The maximum correlation coefficient for all models occurs for normalized-*k* values in the range of 1 and 2. This makes intuitive sense. On one hand, for *k* smaller than the class size, Deep *k*-NN could fail to look within a large enough neighborhood around a data point to properly judge its conformity. For example, a poison point may lie within a small, yet very tight cluster of other poison points of the same class and be improperly marked as benign even though the poison cluster itself may lie within a much larger cluster of clean target points. On the other hand, for *k* larger than 2 times the class size, the neighborhood may be too large and contain too many data points from a competing class. For example, the current target point may lie in a cluster of other target points, but since the neighborhood is so large that it contains all the target points as well as all the points in the nearby poison class cluster, the current target point will be improperly marked as anomalous.

This upper threshold of normalized-$k = 2$ is confirmed by looking at test accuracy performance in Figure 4 (bottom left). We note that performance is highest in the normalized-k region from 0.2 to 2. It slightly decreases after a normalized-*k* ratio of 2 and sharply decreases after 4. This shows that a model's ability to generalize suffers when too many legitimate data points are removed under sufficiently large values of *k*.

Based on these experiments, we recommend using a normalized-*k* value between 1 and 2 for optimal success in defending against poisoning attacks while minimizing unnecessary removal of clean examples.

5.2. Dealing with Class Imbalance

In our second study, we consider the effectiveness of our defense on datasets with an imbalanced number of examples per class. Given an imbalanced dataset, the target class could be either the majority class or a minority class. The easiest case for the defender is when the target is the majority class. In this case, so long as *k* is set sufficiently large, there will be more than enough target training examples to cause the poisons in their midst (in feature space) to be marked as anomalous after running Deep *k*-NN . In this section, we will consider the worst case, wherein the target class is the smallest minority class in the dataset. Without applying any protocol to balance out the classes, there may not be enough target class neighbors when running Deep *k*-NN to know that the poisons clustered in their midst are anomalous.

A typical way to deal with imbalanced classes is to upweight the loss from examples in the minority classes or, equivalently, sample examples from minority classes at a higher rate that is inversely proportional to the fraction of the dataset that their class occupies. We consider a simple and equivalent modification of the latter protocol: given an imbalanced-class dataset, the examples in each class are *replicated* by a factor of N/n, where n is the number of examples in that class and N is the maximum number of examples in any class. After this operation, the dataset will be larger, but once again balanced.

We study the effect of this data replication protocol on imbalanced classes. Specifically, we set the number of examples in the target class (frog) to $n < N$ while leaving the number of examples in all other classes as N. We then replicate the frog examples by a factor of N/n such that its size match the size of the other classes. Finally, we plot the defense success rate against the class imbalance ratio n/N in Figure 4 (top right). The value of normalized-*k* is fixed at 2 ($k = 100$) for this experiment.

Figure 4 (top right, left panel) shows the defense success rate when no protocol is applied prior to running Deep *k*-NN : the success rate suffers for class balance ratios below 0.7. When our data replication protocol is applied before Deep *k*-NN , the defense success rate is near perfect regardless of the class balance ratio. These results show that our minority class replication protocol, combined with the Deep *k*-NN defense, is very effective at removing poisons in an imbalanced class dataset. Our replication-based balancing protocol normalizes the number of examples considered by the Deep *k*-NN defense in feature space.

Next, we observe the MCC as a function of class imbalance in the absence of any protocol in Figure 4 (middle right, left panel). When the ratio is small, then the only thing that can hurt MCC is the misdetection of the targets as being anomalous. On the other hand, when the ratio is large, there is no class imbalance. MCC performs worst when there is a modest underrepresentation of the target class. That is where both the targets and the poisons can cause false negatives and false positives.

When the replication protocol is applied in Figure 4 (middle right), the MCC experiences an improvement, although the relative improvement is small. More interesting, however, is that the replication protocol stabilizes the MCC against class imbalance; the MCC is essentially a flat curve in Figure 4 (middle right).

All models experience better test accuracy on the CIFAR-10 test set when replicating target examples as shown in Figure 4 (bottom right). Despite only having n *unique* points in feature space, replicating them boosts model performance to be similar to the control experiment with a class balance ratio of 1.0. At lower class balance values, replicating data in unbalanced classes improves test accuracy by 8%.

Based on these experiments, we recommend the protocol of replicating images of underrepresented classes to match the maximum number of examples in any particular class prior to running Deep k-NN . Defense success rate and model generalizability are both improved and stabilized by this protocol.

6. Related Work

We briefly overview related work in the space of defenses to adversarial attacks (Biggio et al., 2013; Goodfellow et al., 2014), which are categorized into two groups: inference time evasion attacks and train time data poisoning attacks. Most adversarial defenses have focused on mitigating evasion attacks, where inference-time inputs are manipulated to cause misclassification. In neural networks, evasion adversarial examples are perturbed such that that the loss on the victim network increases. The search for an optimal perturbation is facilitated by use of the local gradient $\nabla_{\mathbf{x}}\mathcal{L}$ obtained via backpropagation on either a white box network or a surrogate network if the victim network is unknown (Liu et al., 2016). Many defenses against evasion attacks leverage the attacker's reliance on gradient information by finding ways to obfuscate gradients, using non-differentiable layers or reshaping the loss surface such that the gradients are highly uncorrelated. Athalye et al. (2018) showed that obfuscated gradient defenses are insufficient for defending against evasion attacks. Using various strategies to circumvent loss of gradient information, such as replacing non-differentiable layers with differentiable approximations during the backward pass, Athalye et al. (2018) demonstrates that stronger attacks can reduce inference accuracy to near zero on most gradient-based defenses. Defense strategies that withstand strong attacks are characterized by loss surfaces that are "smooth" with respect to a particular input everywhere in the data manifold. Variants of adversarial training (Madry et al., 2017; Xie et al., 2019; Shafahi et al., 2019) and linearity or curvature regularizers (Qin et al., 2019; Moosavi-Dezfooli et al., 2019) have maintained modest accuracy despite strong multi-iteration PGD attacks (Madry et al., 2017).

In evasion attacks, Deep k-NN based methods have been used across multiple layers of a neural network to generate confidence estimates of network predictions as a way to detect adversarial examples (Papernot & McDaniel, 2018). Similarly, (Sitawarin & Wagner, 2019) proposes a white box threat model where an adversary has full access to the training set, and uses prior knowledge of model hyperparameters, including the value of k used in the Deep k-NN defense when constructing poisons for general attacks. Our Deep k-NN based defense differs in that it identifies and filters poisoned data at training time rather than at test time, using ground truth labels. Furthermore, a soft nearest neighbor regularizer has been used during training time to improve robustness to evasion examples (Frosst et al., 2019), but its resistance to clean-label poisoning examples has yet to be explored.

Backdoor attacks have recently received attention from the research community as a realistic threat to machine learning models. Backdooring, proposed by (Gu et al., 2017), can be seen as a subset of data poisoning. In their simplest form, backdoor attacks modify a small number of training examples with a specific *trigger* pattern that is accompanied by a *target* label. These attacks exploit a neural network's ability to over fit to the training set data, and use the trigger at inference time to misclassify an example into the target class. The trigger need not change the ground truth label of the training example, making such attacks clean-label attacks (Turner et al., 2019). However, these attacks rely upon the attacker being able to modify data at inference time, an assumption that may not always hold true, and one we do not make in this paper.

A number of defenses to backdoor attacks have been proposed. Many defenses seek to sanitize training data by detecting and removing poisons. Often, these defenses rely upon the heuristic that backdoor attacks create "shortcuts" in a neural network to induce target misclassification. Steinhardt et al. (2017) employed two variants of an L_2 centroid defense, which we adapt in this paper. In one case, data is anomalous if it falls outside of an acceptable radius in feature space. Alternatively, data is first projected onto a line connecting class centroids in feature space and is removed based on its position on this line.

Chen et al. (2018) proposed using feature clustering for data sanitation. This defense assumes that naive backdoor triggers will cause poison samples to cluster in feature space. The success for this defense diminishes drastically when exposed to stronger poisoning methods which do not use uniform triggers. Convex polytope attacks (Zhu et al., 2019) create much stronger poisons by surrounding a target image in feature space with a convex hull of poisons. Such attacks will not always result in easily identifiable clusters of poisons. Tran et al. (2018) examines spectral signatures as a method for detecting backdoor attacks, stating that all attacks share a set of underlying properties. Spectral signatures are boosted in learned representations, and can be used to identify poisoned images through SVD.

Lastly, we consider defenses that seek to identify and reconstruct triggers that causes misclassification. Wang et al. (2019) removes input data points if its activation is similar to the activations induced by the reconstructed trigger. This defense is able to detect certain uniform ℓ_0 triggers inserted in training data using methods such as neuron activation clustering. However, this tactic does work well on recent poisoning attacks that use variable, learned perturbations to cause misclassification via feature collisions (Shafahi et al., 2018; Zhu et al., 2019). Qiao et al. (2019) addresses the problem of reconstructing a single trigger by using sample free generation to create a distribution of potential triggers.

7. Conclusion

In summary, we have demonstrated that the simple Deep k-NN approach provides an effective defense against clean-label poisoning attacks with minimal degradation in model performance. With an appropriately selected value of k, the Deep k-NN defense identifies virtually all poisons from two state-of-the-art clean-label data poisoning attacks, while only filtering a small percentage of clean images. The Deep k-NN defense outperforms other data poisoning baselines and provides a strong benchmark on which to measure the efficacy of future defenses.

References

Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., and Roli, F. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pp. 387–402. Springer, 2013.

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*, 2018.

Chen, X., Ma, H., Wan, J., Li, B., and Xia, T. Multi-view 3d object detection network for autonomous driving. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1907–1915, 2017a.

Chen, Y., Li, J., Xiao, H., Jin, X., Yan, S., and Feng, J. Dual path networks. In *Advances in Neural Information Processing Systems*, pp. 4467–4475, 2017b.

Frosst, N., Papernot, N., and Hinton, G. Analyzing and improving representations with the soft nearest neighbor loss. *arXiv preprint arXiv:1902.01889*, 2019.

Goodfellow, I., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL `http://arxiv.org/abs/1412.6572`.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

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

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

Hu, J., Shen, L., and Sun, G. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141, 2018.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269. IEEE, 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.

Koh, P. W., Steinhardt, J., and Liang, P. Stronger data poisoning attacks break data sanitization defenses. *arXiv preprint arXiv:1811.00741*, 2018.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Liu, Y., Chen, X., Liu, C., and Song, D. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

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.

Moosavi-Dezfooli, S.-M., Fawzi, A., Uesato, J., and Frossard, P. Robustness via curvature regularization, and vice versa. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9078–9086, 2019.

Papernot, N. and McDaniel, P. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*, 2018.

Pascanu, R., Stokes, J. W., Sanossian, H., Marinescu, M., and Thomas, A. Malware classification with recurrent networks. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1916–1920. IEEE, 2015.

Qiao, X., Yang, Y., and Li, H. Defending neural backdoors via generative distribution modeling. 2019.

Qin, C., Martens, J., Gowal, S., Krishnan, D., Fawzi, A., De, S., Stanforth, R., Kohli, P., et al. Adversarial robustness through local linearization. *arXiv preprint arXiv:1907.02610*, 2019.

Rizoiu, M.-A., Wang, T., Ferraro, G., and Suominen, H. Transfer learning for hate speech detection in social media. In *Conference on Artificial Intelligence (AAAI)*, 2019.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. *arXiv preprint arXiv:1801.04381*, 2018.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., and Williamson, R. C. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7): 1443–1471, 2001.

Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*, pp. 6103–6113, 2018.

Shafahi, A., Najibi, M., Ghiasi, A., Xu, Z., Dickerson, J. P., Studer, C., Davis, L. S., Taylor, G., and Goldstein, T. Adversarial training for free! In *Advances in Neural Information Processing Systems*, 2019.

Sitawarin, C. and Wagner, D. On the robustness of keep k-nearest neighbors. 2019.

Steinhardt, J., Koh, P. W., and Liang, P. Certified defenses for data poisoning attacks. *CoRR*, abs/1706.03691, 2017. URL http://arxiv.org/abs/1706.03691.

Sun, Y., Wang, X., and Tang, X. Deep learning face representation from predicting 10,000 classes. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1891–1898, 2014.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

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. Clean-label backdoor attacks, 2019. URL https://people.csail.mit.edu/madry/lab/cleanlabel.pdf.

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. *Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks*, pp. 0, 2019.

Xie, C., Wu, Y., Maaten, L. v. d., Yuille, A. L., and He, K. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 501–509, 2019.

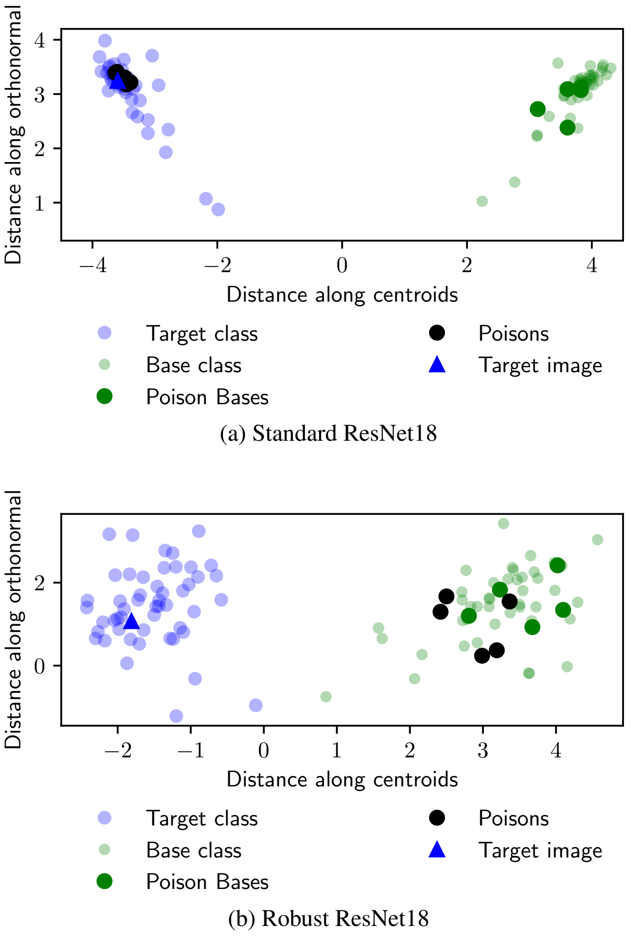
Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pp. 5987–5995. IEEE, 2017.

Zhu, C., Huang, W. R., Li, H., Taylor, G., Studer, C., and Goldstein, T. Transferable clean-label poisoning attacks on deep neural nets. In *International Conference on Machine Learning*, pp. 7614–7623, 2019.

A. Adversarial Training as a Defense against Poisoning Attacks: Visualizations

In Section 4.2.1, we explored if an adversarially trained feature extractor enables the network that is fine-tuned on the poisons to be resistant to data poisoning. In order to be successful, the adversarial trained feature extractor must prevent the feature representations of the poisons from colliding with or forming a convex polytope around the target in feature space. We visualize this intuition using a ResNet18 in Figure 5. Indeed, when the feature extractor is not adversarially trained, the poisons encroach the target class's cluster and surround the target representation (Figure 5a). When the feature extractor is robust and adversarially trained, however, the poisons remain inside their own class's cluster (Figure 5b). Robust feature extractors prevent the feature representations of poisons from becoming non-conformant anomalies in the first place.

For a collision or convex polytope attack to be successful, the poisons must surround the target, an effect which could be measured by the average poison-to-target distance. In Figure 6, we show the average poison-to-target distance at multiple layers of the ResNet18 network. We see that the distance is relatively large for the earlier layers of the network regardless of whether the model is robustly trained. In the last two layers, however, the non-robust model collides the poisons with the target while the robust model keeps their distances far apart.



(a) Standard ResNet18

(b) Robust ResNet18

Figure 5. Visualization of a 5-poison Convex Polytope attack in the feature space of a (top) non-robust and (bottom) robust ResNet18 model.

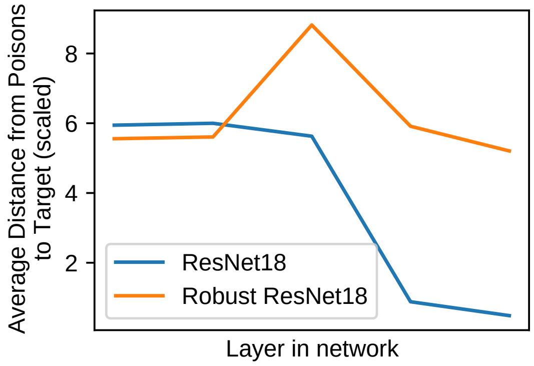


Figure 6. Average distance from poisons to target with adversarial training