Transferable Clean-Label Poisoning Attacks on Deep Neural Nets

Chen Zhu [*1] **W. Ronny Huang** [*1] **Ali Shafahi** [1] **Hengduo Li** [1] **Gavin Taylor** [2] **Christoph Studer** [3] **Tom Goldstein** [1]

Abstract

Clean-label poisoning attacks inject innocuous looking (and "correctly" labeled) poison images into training data, causing a model to misclassify a targeted image after being trained on this data. We consider *transferable* poisoning attacks that succeed without access to the victim network's outputs, architecture, or (in some cases) training data. To achieve this, we propose a new "polytope attack" in which poison images are designed to surround the targeted image in feature space. We also demonstrate that using Dropout during poison creation helps to enhance transferability of this attack. We achieve transferable attack success rates of over 50% while poisoning only 1% of the training set.

1. Introduction

Deep neural networks require large datasets for training and hyper-parameter tuning. As a result, many practitioners turn to the web as a source for data, where one can automatically scrape large datasets with little human oversight. Unfortunately, recent results have demonstrated that these data acquisition processes can lead to security vulnerabilities. In particular, retrieving data from untrusted sources makes models vulnerable to *data poisoning attacks* wherein an attacker injects maliciously crafted examples into the training set in order to hijack the model and control its behavior.

This paper is a call for attention to this security concern. We explore effective and transferable *clean-label poisoning attacks* on image classification problems. In general, data poisoning attacks aim to control the model's behavior during inference by modifying its training data (Shafahi et al., 2018; Suciu et al., 2018; Koh & Liang, 2017; Mahloujifar et al., 2017). In contrast to evasion attacks (Biggio et al., 2013; Szegedy et al., 2013; Goodfellow et al., 2015) and recently proposed backdoor attacks (Liu et al., 2017; Chen et al., 2017a; Turner et al., 2019), we study the case where the targeted samples are *not* modified during inference.

Clean-label poisoning attacks differ from other poisoning attacks (Biggio et al., 2012; Steinhardt et al., 2017) in a critical way: they do not require the user to have any control over the labeling process. Therefore, the poison images need to maintain their malicious properties even when labeled correctly by an expert. Such attacks open the door for a unique threat model in which the attacker poisons datasets simply by placing malicious images on the web, and waiting for them to be harvested by web scraping bots, social media platform operators, or other unsuspecting victims. Poisons are then properly categorized by human labelers and used during training. Furthermore, targeted clean label attacks do not indiscriminately degrade test accuracy but rather target misclassification of specific examples, rendering the presence of the attack undetectable by looking at overall model performance.

Clean-label poisoning attacks have been demonstrated only in the white-box setting where the attacker has complete knowledge of the victim model, and uses this knowledge in the course of crafting poison examples (Shafahi et al., 2018; Suciu et al., 2018). Black-box attacks of this type have not been explored; thus, we aim to craft clean-label poisons which transfer to unknown (black-box) deep image classifiers.

It has been demonstrated in evasion attacks that with only query access to the victim model, a substitute model can be trained to craft adversarial perturbations that fool the victim to classify the perturbed image into a specified class (Papernot et al., 2017). Compared to these attacks, transferable poisoning attacks remain challenging for two reasons. First, the victim's decision boundary trained on the poisoned dataset is more unpredictable than the unknown but fixed decision boundary of an evasion attack victim. Second, the attacker cannot depend on having direct access to the victim model (i.e. through queries) and must thus make the poisons model-agnostic. The latter also makes the attack more dangerous since the poisons can be administered in

[*]Equal contribution [1]University of Maryland, College Park [2]United States Naval Academy [3]Cornell University. Correspondence to: Chen Zhu <chenzhu@cs.umd.edu>, W. Ronny Huang <wronnyhuang@gmail.com>, Tom Goldstein <tomg@cs.umd.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

a distributed fashion (e.g. put on the web to be scraped), compromising more than just one particular victim model.

Here, we demonstrate an approach to produce transferable clean-label targeted poisoning attacks. We assume the attacker has no access to the victim's outputs or parameters, but is able to collect a similar training set as that of the victim. The attacker trains substitute models on this training set, and optimizes a novel objective that forces the poisons to form a polytope in feature space that entraps the target inside its convex hull. A classifier that overfits to this poisoned data will classify the target into the same class as that of the poisons. This new objective has better success rate than feature collision (Shafahi et al., 2018) in the black-box setting, and it becomes even more powerful when enforced in multiple intermediate layers of the network, showing high success rates in both transfer learning and end-to-end training contexts. We also show that using Dropout when crafting the poisons improves transferability.

2. The Threat Model

Like the poisoning attack of (Shafahi et al., 2018), the attacker in our setting injects a small number of perturbed samples (whose labels are true to their class) into the training set of the victim. The attacker's goal is to cause the victim network, once trained, to classify a test image (not in the training set) as a specified class. We consider the case of image classification, where the attacker achieves its goal by adding adversarial perturbations $\boldsymbol{\delta}$ to the images. $\boldsymbol{\delta}$ is crafted so that the perturbed image $\boldsymbol{x}+\boldsymbol{\delta}$ shares the same class as the clean image $\boldsymbol{x}$ for a human labeler, while being able to change the decision boundary of the DNNs in a certain way.

Unlike (Shafahi et al., 2018) or (Papernot et al., 2017), which requires full or query access to the victim model, here we assume the victim model is not accessible to the attacker, which is a practical assumption in many systems such as autonomous vehicles and surveillance systems. Instead, we need the attacker to have knowledge about the victim's training distribution, such that a similar training set can be collected for training substitute models.

We consider two learning approaches that the victim may adopt. The first learning approach is *transfer learning*, in which a pre-trained but frozen feature extractor ϕ, e.g., the convolutional layers of a ResNet (He et al., 2016) trained on a reference dataset like CIFAR or ImageNet, is applied to images, and an application-specific linear classifier with parameters $\boldsymbol{W}, \boldsymbol{b}$ is fine-tuned on the features $\phi(\mathcal{X})$ of another dataset $\mathcal{X}$. Transfer learning of this type is common in industrial applications when large sets of labeled data are unavailable for training a good feature extractor. Poisoning attacks on transfer learning were first studied in the white-box settings where the feature extractor is known in (Koh & Liang, 2017; Shafahi et al., 2018), and similarly in (Mei & Zhu, 2015; Biggio et al., 2012), all of which target linear classifiers over deep features.

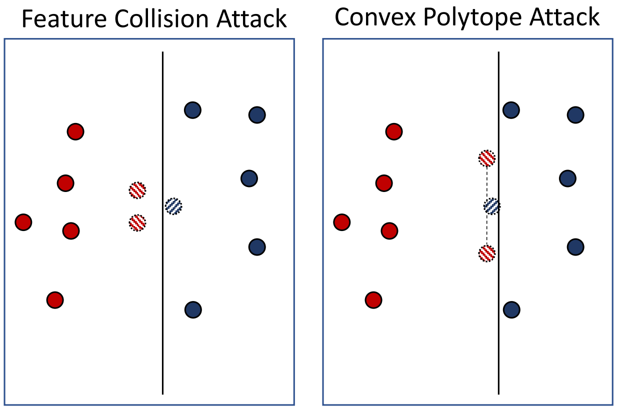


Figure 1. An illustrative toy example of a linear SVM trained on a two-dimensional space with training sets poisoned by Feature Collision Attack and Convex Polytope Attack respectively. The two striped red dots are the poisons injected to the training set, while the striped blue dot is the target, which is not in the training set. All other points are in the training set. Even when the poisons are the closest points to the target, the optimal linear SVM will classify the target correctly in the left figure. The Convex Polytope attack will enforce a small distance of the line segment formed by the two poisons to the target. When the line segment's distance to the target is minimized, the target's negative margin in the retrained model is also minimized if it overfits.

The second learning approach is *end-to-end training*, where the feature extractor and the linear classifier are trained jointly. Obviously, such a setting has stricter requirements on the poisons than the transfer learning setting, since the injected poisons will affect the parameters of the feature extractor. (Shafahi et al., 2018) uses a watermarking strategy that superposes up to 30% of the target image onto about 40 poison images, only to achieve about 60% success rate in a 10-way classification setting.

3. Transferable Targeted Poisoning Attacks

3.1. Difficulties of Targeted Poisoning Attacks

Targeted poisoning attacks are more difficult to pull off than targeted evasion attacks. For image classification, targeted evasion attacks only need to make the victim model misclassify the perturbed image into a certain class. The model does not adjust to the perturbation, so the attacker only needs to find the shortest perturbation path to the decision boundary by solving a constrained optimization problem. For example, in the norm-bounded white-box setting, the attacker can directly optimize $\boldsymbol{\delta}$ to minimize the cross entropy loss L_{CE} on the target label y_t by solving

$\boldsymbol{\delta}_t = \arg\min_{||\boldsymbol{\delta}||_\infty \le \epsilon} L_{CE}(\boldsymbol{x}_t + \boldsymbol{\delta}, y_t)$ with Projected Gradient Descent (Madry et al., 2017). In the norm-bounded black-box setting, with query access, the attacker can also train a substitute model to simulate the behavior of the victim via distillation, and perform the same optimization w.r.t. the substitute model to find a transferable $\boldsymbol{\delta}$ (Papernot et al., 2017).

Targeted poisoning attacks, however, face a more challenging problem. The attacker needs to get the victim to classify the target sample $\boldsymbol{x}_t$ into the alternative target class $\tilde{y}_t$ after being trained on the modified data distribution. One simple approach is to select the poisons from class $\tilde{y}_t$, and make the poisons as close to $\boldsymbol{x}_t$ as possible in the feature space. A rational victim will usually overfit the training set, since it is observed in practice that generalization keeps improving even after training loss saturates (Zhang et al., 2016). As a result, when the poisons are close to the target in the feature space, a rational victim is likely to classify $\boldsymbol{x}_t$ into $\tilde{y}_t$ since the space near the poisons are classified as $\tilde{y}_t$. However, as shown in Figure 1, smaller distance to the target does not always lead to a successful attack. In fact, being close to the target might be too restrictive for successful attacks. Indeed, there exists conditions where the poisons can be farther away from the target, but the attack is more successful.

3.2. Feature Collision Attack

Feature collision attacks, as originally proposed in (Shafahi et al., 2018), are a reliable way of producing targeted clean-label poisons on white-box models. The attacker selects a base example $\boldsymbol{x}_b$ from the targeted class for crafting the poisons $\boldsymbol{x}_p$, and tries to make $\boldsymbol{x}_p$ become the same as the target $\boldsymbol{x}_t$ in the feature space by adding small adversarial perturbations to $\boldsymbol{x}_b$. Specifically, the attacker solves the following optimization problem (1) to craft the poisons:

$$\boldsymbol{x}_p = \arg\min_{\boldsymbol{x}} \|\boldsymbol{x} - \boldsymbol{x}_b\|^2 + \mu\|\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}_t)\|^2, \qquad (1)$$

where ϕ is a pre-trained neural feature extractor. The first term enforces the poison to lie near the base in input space, and therefore maintains the same label as $\boldsymbol{x}_b$ to a human labeler. The second term forces the feature representation of the poison to collide with the feature representation of the target. The hyperparameter $\mu > 0$ trades off the balance between these terms.

If the poison example $\boldsymbol{x}_p$ is correctly labeled as a member of the targeted class and placed in the victim's training dataset χ, then after training on χ, the victim classifier learns to classify the poison's feature representation into the targeted class. Then, if $\boldsymbol{x}_p$'s feature distance to $\boldsymbol{x}_t$ is smaller than the margin of $\boldsymbol{x}_p$, $\boldsymbol{x}_t$ will be classified into the same class as $\boldsymbol{x}_p$ and the attack is successful. Sometimes more than one poison image is used to increase the success rate.

Unfortunately for the attacker, different feature extractors ϕ will lead to different feature spaces, and a small feature space distance $d_\phi(\boldsymbol{x}_p, \boldsymbol{x}_t) = \|\phi(\boldsymbol{x}_p) - \phi(\boldsymbol{x}_t)\|$ for one feature extractor does not guarantee a small distance for another.

Fortunately, the results in (Tramèr et al., 2017) have demonstrated that for each input sample, there exists an adversarial subspace for different models trained on the same dataset, such that a moderate perturbation can cause the models to misclassify the sample, which indicates it is possible to find a small perturbation to make the poisons $\boldsymbol{x}_p$ close to the target $\boldsymbol{x}_t$ in the feature space for different models, as long as the models are trained on the same dataset or similar data distributions.

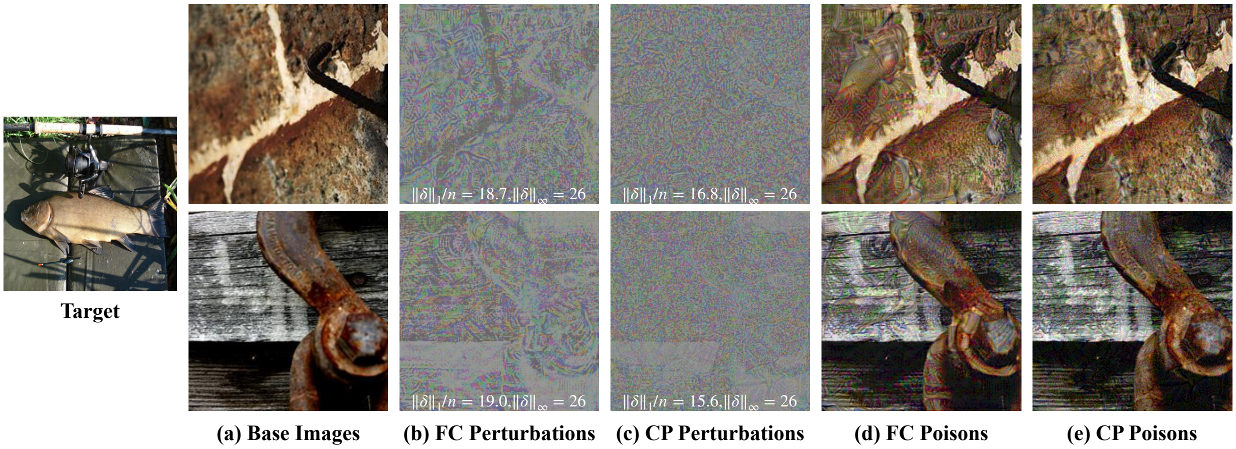
With such observation, the most obvious approach to forge a black-box attack is to optimize a set of poisons $\{\boldsymbol{x}_p^{(j)}\}_{j=1}^k$ to produce feature collisions for an *ensemble* of models $\{\phi^{(i)}\}_{i=1}^m$, where m is the number of models in the ensemble. Such a technique was also used in black-box evasion attacks (Liu et al., 2016). Because different extractors produce feature vectors with different dimensions and magnitudes, we use the following normalized feature distance to prevent any one network from dominating the objective due to such biases:

$$L_{FC} = \sum_{i=1}^{m} \sum_{j=1}^{k} \frac{\|\phi^{(i)}(\boldsymbol{x}_p^{(j)}) - \phi^{(i)}(\boldsymbol{x}_t)\|^2}{\|\phi^{(i)}(\boldsymbol{x}_t)\|^2}. \qquad (2)$$

3.3. Convex Polytope Attack

One problem with the feature collision attack is the emergence of obvious patterns of the target in the crafted perturbations. Unlike prevalent objectives for evasion attacks which maximize single-entry losses like cross entropy, feature collision (Shafahi et al., 2018) enforces each entry of the poison's feature vector $\phi(\boldsymbol{x}_p)$ to be close to $\phi(\boldsymbol{x}_t)$, which usually results in hundreds to thousands of constraints on each poison image $\boldsymbol{x}_p$. What is worse, in the black-box setting as Eq. 2, the poisoning objective forces a collision over an ensemble of m networks, which further increases the number of constraints on the poison images. With such a large number of constraints, the optimizer often resorts to pushing the poison images in a direction where obvious patterns of the target will occur, therefore making $\boldsymbol{x}_p$ look like the target class. As a result, human workers will notice the difference and take actions. Figure 2 shows a qualitative example in the process of crafting poisons from images of *hook* to attack a *fish* image with Eq. 2. Elements of the target image that are evident in the poison images include the fish's tail in the top image and almost a whole fish in the bottom image in column 3.

Another problem with Feature Collision Attack is its lack of transferability. Feature Collision Attack tends to fail in the black-box setting because it is difficult to make the poisons


Target
$\|\delta\|_1/n = 18.7, \|\delta\|_\infty = 26$
$\|\delta\|_1/n = 16.8, \|\delta\|_\infty = 26$
$\|\delta\|_1/n = 19.0, \|\delta\|_\infty = 26$
$\|\delta\|_1/n = 15.6, \|\delta\|_\infty = 26$

(a) Base Images **(b) FC Perturbations** **(c) CP Perturbations** **(d) FC Poisons** **(e) CP Poisons**

Figure 2. A qualitative example of the difference in poison images generated by Feature Collision (FC) Attack and Convex Polytope (CP) Attack. Both attacks aim to make the model mis-classify the target *fish* image on the left into a *hook*. We show two of the five *hook* images that were used for the attack, along with their perturbations and the poison images here. Both attacks were successful, but unlike FC, which demonstrated strong regularity in the perturbations and obvious fish patterns in the poison images, CP tends to have no obvious pattern in its poisons. More details are provided in the supplementary.

close to $\boldsymbol{x}_t$ for every model in the feature space. The feature space of different feature extractors should be very different, since neural networks are using one linear classifier to separate the deep features $\phi(\boldsymbol{x})$, which has a unique solution if $\phi(\boldsymbol{x})$ is given, but different networks usually have different accuracies. Therefore, even if the poisons $\boldsymbol{x}_p$ collide with $\boldsymbol{x}_t$ in the feature space of the substitute models, they probably do not collide with $\boldsymbol{x}_t$ in the unknown target model, due to the generalization error. As demonstrated by Figure 1, the attack is likely to fail even when $\boldsymbol{x}_p$ has smaller distance to $\boldsymbol{x}_t$ than its intra-class samples. It is also impractical to ensemble too many substitute models to reduce such error. We provide experimental results with the ensemble Feature Collision Attack defined by Eq. 2 to show it can be ineffective.

We therefore seek a looser constraint on the poisons, so that the patterns of the target are not obvious in the poison images and the requirements on generalization are reduced. Noticing (Shafahi et al., 2018) usually use multiple poisons to attack one target, we start by deriving the necessary and sufficient conditions on the set of poison features $\{\phi(\boldsymbol{x}_p^{(j)})\}_{j=1}^k$ such that the target $\boldsymbol{x}_t$ will be classified into the poison's class.

Proposition 1. *The following statements are equivalent:*

1. *Every linear classifier that classifies* $\{\phi(\boldsymbol{x}_p^{(j)})\}_{j=1}^k$ *into label* ℓ_p *will classify* $\phi(\boldsymbol{x}_t)$ *into label* ℓ_p.
2. $\phi(\boldsymbol{x}_t)$ *is a convex combination of* $\{\phi(\boldsymbol{x}_p^{(j)})\}_{j=1}^k$, *i.e.,* $\phi(\boldsymbol{x}_t) = \sum_{j=1}^k c_j \phi(\boldsymbol{x}_p^{(j)})$, *where* $c_1, \dots, c_k \geq 0, \sum_{j=1}^k c_j = 1$.

The proof of Proposition 1 is given in the supplementary material. In words, a set of poisons from the same class is guaranteed to alter the class label of a target into theirs if that target's feature vector lies in the *convex polytope* of the poison feature vectors. We emphasize that this is a far more relaxed condition than enforcing a feature collision—it enables the poisons to lie much farther away from the target while altering the labels on a much larger region of space. As long as $\boldsymbol{x}_t$ lives inside this region in the unknown target model, and $\{\boldsymbol{x}_p^{(j)}\}$ are classified as expected, $\boldsymbol{x}_t$ will be classified as the same label as $\{\boldsymbol{x}_p^{(j)}\}$.

With this observation, we optimize the set of poisons towards forming a convex polytope in the feature space such that the target's feature vector will lie within or at least close to the convex polytope. Specifically, we solve the following optimization problem:

$$\begin{aligned}
\underset{\{\mathbf{c}^{(i)}\},\{\boldsymbol{x}_p^{(j)}\}}{\text{minimize}} \; & \frac{1}{2}\sum_{i=1}^{m} \frac{\|\phi^{(i)}(\boldsymbol{x}_t) - \sum_{j=1}^k c_j^{(i)} \phi^{(i)}(\boldsymbol{x}_p^{(j)})\|^2}{\|\phi^{(i)}(\boldsymbol{x}_t)\|^2} \\
\text{subject to} \; & \sum_{j=1}^{k} c_j^{(i)} = 1, c_j^{(i)} \geq 0, \forall i, j, \\
& \|\boldsymbol{x}_p^{(j)} - \boldsymbol{x}_b^{(j)}\|_\infty \leq \epsilon, \forall j,
\end{aligned} \tag{3}$$

where $\boldsymbol{x}_b^{(j)}$ is the clean image of the j-th poison, and ϵ is the maximum allowable perturbation such that the perturbations are not immediately perceptible.

Eq. 3 simultaneously finds a set of poisons $\{\boldsymbol{x}_p^{(j)}\}$, and a set of convex combination coefficients $\{c_j^{(i)}\}$ such that the target lies in or close to the convex polytope of the poisons in the feature space of the m models. Notice the coefficients $\{c_j^{(i)}\}$ are untied, i.e., they are allowed to vary across different models, which does not require $\phi^{(i)}(\boldsymbol{x}_t)$ to be close to any specific point in the polytope, including the vertices $\{\boldsymbol{x}_p^{(j)}\}$. Given the same amount of perturbation, such an objective is also more relaxed than Feature Collision Attack (Eq. 2) since Eq. 2 is a special case of Eq. 3 when we fix $c_j^{(i)} = 1/k$. As a result, the poisons demonstrate almost no patterns of the target, and the imperceptibility of the attack is enhanced compared with feature collision attack, as shown in Figure 2.

The most important benefit brought by the convex polytope objective is the improved transferability. For Convex Polytope Attack, $\boldsymbol{x}_t$ does not need to align with a specific point in the feature space of the unknown target model. It only needs to lie within the convex polytope formed by the poisons. In the case where this condition is not satisfied, Convex Polytope Attack still has advantages over Feature Collision Attack. Suppose for a given target model, a residual[1] smaller than ρ will guarantee a successful attack[2]. For Feature Collision Attack, the target's feature needs to lie within a ρ-ball centered at $\phi^{(t)}(\boldsymbol{x}_p^{(j^*)})$, where $j^* = \arg\min_j \|\phi^{(t)}(\boldsymbol{x}_p^{(j)}) - \phi^{(t)}(\boldsymbol{x}_t)\|$. For Convex Polytope Attack, the target's feature could lie within the ρ-expansion of the convex polytope formed by $\{\phi^{(t)}(\boldsymbol{x}_p^{(j)})\}_{j=1}^k$, which has a larger volume than the aforementioned ρ-ball, and thus tolerates larger generalization error.

3.4. An Efficient Algorithm for Convex Polytope Attack

We optimize the non-convex and constrained problem (3) using an alternating method that side-steps the difficulties posed by the complexity of $\{\phi^{(i)}\}$ and the convex polytope constraints on $\{\boldsymbol{c}^{(i)}\}$. Given $\{\boldsymbol{x}_p^{(j)}\}_{j=1}^k$, we use forward-backward splitting (Goldstein et al., 2014) to find the the optimal sets of coefficients $\{\boldsymbol{c}^{(i)}\}$. This step takes much less computation than back-propagation through the neural network, since the dimension of $\boldsymbol{c}^{(i)}$ is usually small (in our case a typical value is 5). Then, given the optimal $\{\boldsymbol{c}^{(i)}\}$ with respect to $\{\boldsymbol{x}_p^{(j)}\}$, we take one gradient step to optimize $\{\boldsymbol{x}_p^{(j)}\}$, since back-propagation through the m networks is relatively expensive. Finally, we project the poison images to be within ϵ units of the clean base image so that the perturbation is not obvious, which is implemented as a clip operation. We repeat this process to find the optimal set of poisons and coefficients, as shown in Algorithm 1.

In our experiments, we find that after the first iteration, initializing $\{\boldsymbol{c}^{(i)}\}$ to the value from the last iteration accelerates its convergence. We also find the loss in the target network to bear high variance without momentum optimizers. Therefore, we choose Adam (Kingma & Ba, 2014) as the optimizer for the perturbations as it converges more reliably than SGD in our case. Although the constraint on the perturbation is ℓ_∞ norm, in contrast to (Dong et al., 2018) and the common practices for crafting adversarial perturbations such as FGSM (Kurakin et al., 2016), we do not take the sign of the gradient, which further reduces the variance caused by the flipping of signs when the update step is already small.

Algorithm 1 Convex Polytope Attack

```
Data: Clean base images$\{\boldsymbol{x}_b^{(j)}\}_{j=1}^k$, substitute networks
      $\{\phi^{(i)}\}_{i=1}^m$, and maximum perturbation $\epsilon$.
Result: A set of perturbed poison images $\{\boldsymbol{x}_p^{(j)}\}_{j=1}^k$.
Initialize $\boldsymbol{c}^{(i)} \leftarrow \frac{1}{k}\mathbf{1}$, $\boldsymbol{x}_p^{(j)} \leftarrow \boldsymbol{x}_b^{(j)}$
while not converged do
    for i=1,...,m do
        $A \leftarrow [\phi^{(i)}(\boldsymbol{x}_p^{(1)}), \dots, \phi^{(i)}(\boldsymbol{x}_p^{(k)})]$
        $\alpha \leftarrow 1/\|A^\top A\|_2$
        while not converged do
            $\boldsymbol{c}^{(i)} \leftarrow \boldsymbol{c}^{(i)} - \alpha A^\top(A\boldsymbol{c}^{(i)} - \phi^{(i)}(\boldsymbol{x}_t))$
            project $\boldsymbol{c}^{(i)}$ onto probability simplex
        end
    end
    Gradient step on $\boldsymbol{x}_p^{(j)}$ with Adam
    Clip $\boldsymbol{x}_p^{(j)}$ so that the infinity norm constraint is satisfied.
end
```

3.5. Multi-Layer Convex Polytope Attack

When the victim trains its feature extractor ϕ in addition to the classifier (last layer), enforcing Convex Polytope Attack only on the feature space of $\phi^{(i)}$ is not enough for a successful attack as we will show in experiments. In this setting, the change in feature space caused by a model trained on the poisoned data will also make the polytope more difficult to transfer.

Unlike linear classifiers, deep neural networks have much better generalization on image datasets. Since the poisons all come from one class, the whole network can probably generalize well enough to discriminate the distributions of $\boldsymbol{x}_t$ and $\boldsymbol{x}_p$, such that after trained with the poisoned dataset, it will still classify $\boldsymbol{x}_t$ correctly. As shown in Figure 7, when CP attack is applied to the last layer's features, the lower capacity models like SENet18 and ResNet18 are more susceptible to the poisons than other larger capacity models. However, we know there probably exist poisons with small

[1] For FC, it is $\min_j \|\phi^{(t)}(\boldsymbol{x}_p^{(j)}) - \phi^{(t)}(\boldsymbol{x}_t)\|$; for CP, it is $\|\sum_j c_j^{(t)} \phi^{(t)}(\boldsymbol{x}_p^{(j)}) - \phi^{(t)}(\boldsymbol{x}_t)\|$

[2] When the residual is small enough, $\phi^{(t)}(\boldsymbol{x}_t)$ will not cross the decision boundary if poisons are classified as expected.

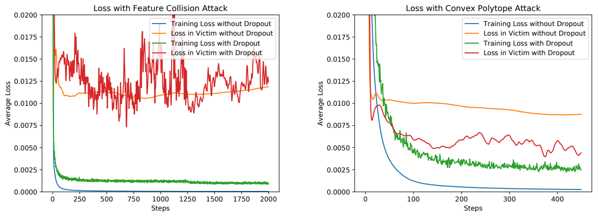


Figure 3. Loss curves of Feature Collision and Convex Polytope Attack on the substitute models and the victim models, tested using the target with index 2. Dropout improved the minimum achievable test loss for the FC attack, and improved the test loss of the CP attack significantly.

perturbations that are transferable to networks trained on the poison distribution, as empirical evidence from (Tramèr et al., 2017) have shown there exist a common adversarial subspace for different models trained on the same dataset, and naturally trained networks usually have large enough Lipschitz to cause mis-classification (Szegedy et al., 2013), which hopefully will also be capable of shifting the polytope into such a subspace to lie close to $\boldsymbol{x}_t$.

One strategy to increase transferability to models trained end-to-end on the poisoned dataset is to jointly apply Convex Polytope Attack to multiple layers of the network. The deep network is broken into shallow networks $\phi_1, ..., \phi_n$ by depth, and the objective now becomes

$$\underset{\{\boldsymbol{c}_l^{(i)}\},\{\boldsymbol{x}_p^{(j)}\}}{\text{minimize}} \sum_{l=1}^{n}\sum_{i=1}^{m} \frac{\|\phi_{1:l}^{(i)}(\boldsymbol{x}_t) - \sum_{j=1}^{k} c_{l,j}^{(i)} \phi_{1:l}^{(i)}(\boldsymbol{x}_p^{(j)})\|^2}{\|\phi_{1:l}^{(i)}(\boldsymbol{x}_t)\|^2}, \tag{4}$$

where $\phi_{1:l}^{(i)}$ is the concatenation from $\phi_1^{(i)}$ to $\phi_l^{(i)}$. Networks similar to ResNet are broken into blocks separated by pooling layers, and we let $\phi_l^{(i)}$ be the l-th layer of such blocks. The optimal linear classifier trained with the features up to $\phi_{1:l}$ ($l < n$) will have worse generalization than the optimal linear classifier trained with features of ϕ, and therefore the feature of $\boldsymbol{x}_t$ should have higher chance to deviate from the features of the same class after training, which is a necessary condition for a successful attack. Meanwhile, with such an objective, the perturbation is optimized towards fooling models with different depths, which further increases the variety of the substitute models and adds to the transferability.

3.6. Improved Transferability via Network Randomization

Even when trained on the same dataset, different models have different accuracy and therefore different distributions of the samples in the feature space. Ideally, if we craft the poisons with arbitrarily many networks from the function class of the target network then we should be able effectively minimize Eq. 3 in the target network. It is, however, impractical to ensemble a large number of networks due to memory constraints.

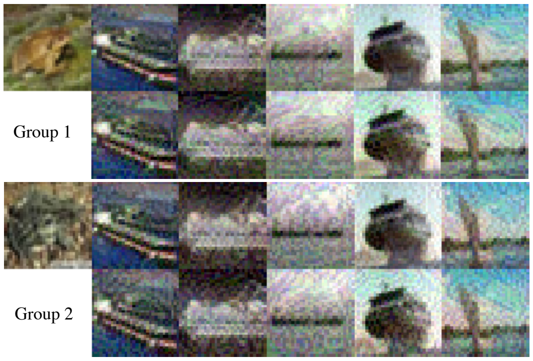


Figure 4. Qualitative results of the poisons crafted by FC and CP. Each group shows the target along with five poisons crafted to attack it, where the first row is the poisons crafted with FC, and the second row is the poisons crafted with CP. In the first group, the CP poisons fooled a DenseNet121 but FC poisons failed, in the second group both succeeded. The second target's image is more noisy and is probably an outlier of the frog class, so it is easier to attack. The poisons crafted by CP contain fewer patterns of $\boldsymbol{x}_t$ than with FC, and are harder to detect.

To avoid ensembling too many models, we randomize the networks with Dropout (Srivastava et al., 2014), turning it on when crafting the poisons. In each iteration, each substitute network $\phi^{(i)}$ is randomly sampled from its function class by shutting off each neuron with probability p, and multiplying all the "on" neurons with $1/(1-p)$ to keep the expectation unchanged. In this way, we can get an exponential (in depth) number of different networks for free in terms of memory. Such randomized networks increase transferability in our experiments. One qualititative example is given in Figure 3.

4. Experiments

In the following, we will use CP and FC as abbreviations for Convex Polytope Attacks and Feature Collision attacks respectively. The code for the experiments is available at `https://github.com/zhuchen03/ConvexPolytopePosioning`.

Datasets In this section, all images come from the CIFAR10 dataset. If not explicitly specified, we take the first 4800 images from each of the 10 classes (a total of 48000 images) in the training set to pre-train the victim models and the substitute models ($\phi^{(i)}$). We leave the test set intact so that the accuracy of these models under different settings can be evaluated on the standard test set and compared directly with the benchmarks. A successful attack should not only have unnoticeable image perturbations, but also unchanged test accuracy after fine-tuning on the clean and poisoned datasets. As shown in the supplementary, our attack preserves the accuracy of the victim model compared with the accuracy of those tuned on the corresponding clean dataset.

The remaining 2000 images of the training set serve as a pool for selecting the target, crafting the poisons, and fine-tuning the victim networks. We take the first 50 images from each class (a total of 500 images) in this pool as the clean fine-tuning dataset. This resembles the scenario where pre-trained models on large datasets like Imagenet (Krizhevsky et al., 2012) are fine-tuned on a similar but usually disjoint dataset. We randomly selected “ship” as the target class, and “frog” as the targeted image’s class, i.e., the attacker wants to cause a particular frog image to be misclassified as a ship. The poison images $\boldsymbol{x}_p^{(j)}$ across *all* experiments are crafted from the first 5 images of the ship class in the 500-image fine-tuning dataset. We evaluate the poison’s efficacy on the next 50 images of the frog class. Each of these images is evaluated independently as the target $\boldsymbol{x}_t$ to collect statistics. Again, the target images are not included in the training and fine-tuning set.

Networks Two sets of substitute model architectures are used in this paper. Set S1 includes SENet18 (Hu et al., 2018), ResNet50 (He et al., 2016), ResNeXt29-2x64d (Xie et al., 2017), DPN92 (Chen et al., 2017b), MobileNetV2 (Sandler et al., 2018) and GoogLeNet (Szegedy et al., 2015). Set S2 includes all the architectures of S1 except for MobileNetV2 and GoogLeNet. S1 and S2 are used in different experiments as specified below. ResNet18 and DenseNet121 (Huang et al., 2017) were used as the black-box model architectures. The poisons are crafted on models from the set of substitute model architectures. We evaluate the poisoning attack against victim models from the 6 different substitute model architectures as well as from the 2 black-box model architectures. Each victim model, however, was trained with different random seeds than the substitute models. If the victim’s architecture appears in the substitute models, we call it a gray-box setting; otherwise, it is a black-box setting.

We add a Dropout layer at the output of each Inception block for GoogLeNet, and in the middle of the convolution layers of each Residual-like blocks for the other networks. We train these models from scratch on the aforementioned 48000-image training set with Dropout probabilities of 0, 0.2, 0.25 and 0.3, using the same architecture and hyperparameters (except for Dropout) of a public repository[3]. The victim models that we evaluate were not trained with Dropout.

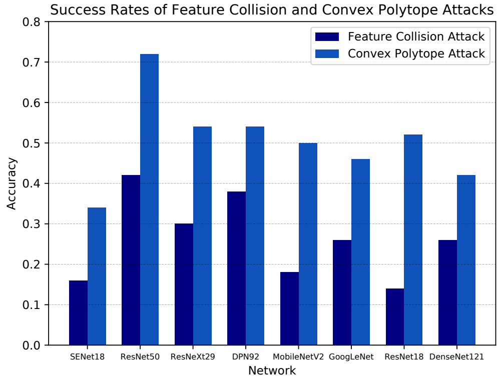


Figure 5. Success rates of FC and CP attacks on various models. Notice the first six entries are the gray-box setting where the models with same architecture but different weights are in the substitute networks, while the last two entries are the black-box setting.

Attacks We use the same 5 poison ship images to attack each frog image. For the substitute models, we use 3 models trained with Dropout probabilities of 0.2, 0.25, 0.3 from each architecture, which results in 18 and 12 substitute models for S1 and S2 respectively. When crafting the poisons, we use the same Dropout probability as the models were trained with. For all our experiments, we set $\epsilon = 0.1$. We use Adam (Kingma & Ba, 2014) with a relatively large learning rate of 0.04 for crafting the poisons, since the networks have been trained to have small gradients on images similar to the training set. We perform no more than 4000 iterations on the poison perturbations in each experiment. Unless specified, we only enforce Eq. 3 on the features of the last layer.

For the victim, we choose its hyperparameters during fine-tuning such that it overfits the 500-image training set, which satisfies the aforementioned rational victim assumption. In the transfer learning setting, where only the final linear classifier is fine-tuned, we use Adam with a large learning rate of 0.1 to overfit. In the end-to-end setting, we use Adam with a small learning rate of 10^{-4} to overfit.

4.1. Comparison with Feature Collision

We first compare the transferability of poisons generated by FC and CP in the transfer learning training context. The results are shown in Figure 5. We use set S1 of substitute architectures. FC never achieves a success rate higher than 0.5, while CP achieves success rates higher or close to 0.5 in most cases. A qualitative example of the poisons crafted by the two approaches is shown in Figure 4.

4.2. Importance of Training Set

Despite being much more successful than FC, questions remain about how reliable CP will be when we have no knowledge of the victim’s training set. In the last section, we trained the substitute models on the same training set as the victim. In Figure 6 we provide results for when the

[3] https://github.com/kuangliu/pytorch-cifar

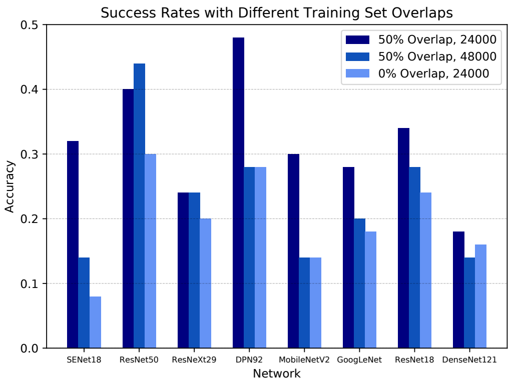


Figure 6. Success rates of Convex Polytope Attack, with poisons crafted by substitute models trained on the first 2400 images of each class of CIFAR10. The models corresponding to the three settings are trained with samples indexed from 1201 to 3600, 1 to 4800 and 2401 to 4800, corresponding to the settings of 50%, 50% and 0% training set overlaps respectively.

substitute models' training sets are similar to (but mostly different from) that of the victim. Such a setting is sometimes more realistic than the setting where no knowledge of the victim's training set is required, but query access to the victim model is needed (Papernot et al., 2017), since query access is not available for scenarios like surveillance. We use the less ideal `S2`, which has 12 substitute models from 4 different architectures. Results are evaluated in the transfer learning setting. Even with no data overlap, CP can still transfer to models with very different structure than the substitute models in the black-box setting. In the 0% overlap setting, the poisons transfer better to models with higher capacity like DPN92 and DenseNet121 than to low-capacity ones like SENet18 and MobileNetV2, probably because high capacity models overfit more to their training set. Overall, we see that CP may remain powerful without access to the training data in the transfer learning setting, as long as the victim's model has good generalization.

4.3. End-to-End Training

A more challenging setting is when the victim adopts end-to-end training. Unlike the transfer learning setting where models with better generalization turn out to be more vulnerable, here good generalization may help such models classify the target correctly despite the poisons. As shown in Figure 7, CP attacks on the last layer's feature is not enough for transferability, leading to almost zero successful attacks. It is interesting to see that the success rate on ResNet50 is 0, which is an architecture in the substitute models, while the success rate on ResNet18 is the highest, which is not an architecture in the substitute models but should have worse generalization.

Therefore, we enforce CP in multiple layers of the substitute models, which breaks the models into lower capacitie ones and leads to much better results. In the gray-box setting, all of the attacks achieved more than 0.6 success rates. However, it remains very difficult to transfer to GoogLeNet, which has a more different architecture than the substitute models. It is therefore more difficult to find a direction to make the convex polytope survive end-to-end training.

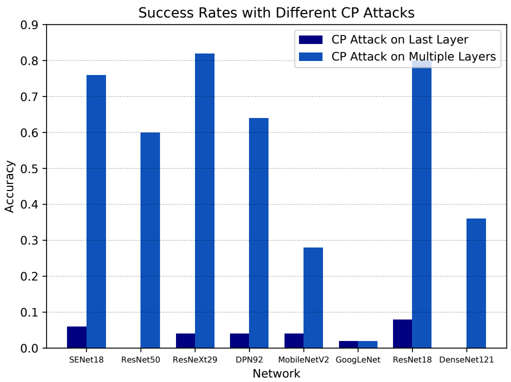


Figure 7. Success rates of Convex Polytope Attack in the end-to-end training setting. We use `S2` for the substitute models.

5. Conclusion

In summary, we have demonstrated an approach to enhance the transferability of clean-labeled targeted poisoning attacks. The main contribution is a new objective which constructs a convex polytope around the target image in feature space, so that a linear classifier which overfits the poisoned dataset is guaranteed to classify the target into the poisons' class. We provided two practical ways to further improve transferability. First, turn on Dropout while crafting poisons, so that the objective samples from a variety (i.e. ensemble) of networks with different structures. Second, enforce the convex polytope objective in multiple layers, which enables attack success even in end-to-end learning contexts. Additionally, we found that transferability can depend on the data distribution used to train the substitute model.

6. Acknowledgements

Goldstein, Shafahi, and Chen were supported by the Office of Naval Research (N00014-17-1-2078), DARPA Lifelong Learning Machines (FA8650-18-2-7833), the DARPA YFA program (D18AP00055), and the Sloan Foundation. Studer was supported in part by Xilinx, Inc. and by the US National Science Foundation (NSF) under grants ECCS-1408006, CCF-1535897, CCF-1652065, CNS-1717559, and ECCS-1824379.

References

Biggio, B., Nelson, B., and Laskov, P. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

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, X., Liu, C., Li, B., Lu, K., and Song, D. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. *arXiv preprint arXiv:1712.05526*, 2017a. URL http://arxiv.org/abs/1712.05526.

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.

Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., and Li, J. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9185–9193, 2018.

Goldstein, T., Studer, C., and Baraniuk, R. A field guide to forward-backward splitting with a fasta implementation. *arXiv preprint arXiv:1411.3406*, 2014.

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.

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.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. *arXiv preprint arXiv:1703.04730*, 2017.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

Liu, Y., Chen, X., Liu, C., and Song, D. Delving into transferable adversarial examples and black-box attacks. *CoRR*, abs/1611.02770, 2016. URL http://arxiv.org/abs/1611.02770.

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

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.

Mahloujifar, S., Diochnos, D. I., and Mahmoody, M. Learning under p-tampering attacks. *CoRR*, abs/1711.03707, 2017. URL http://arxiv.org/abs/1711.03707.

Mei, S. and Zhu, X. Using machine teaching to identify optimal training-set attacks on machine learners. In *AAAI*, pp. 2871–2877, 2015.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pp. 506–519. ACM, 2017.

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

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. *arXiv preprint arXiv:1804.00792*, 2018.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

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.

Suciu, O., Mărginean, R., Kaya, Y., Daumé III, H., and Dumitraş, T. When does machine learning fail? generalized transferability for evasion and poisoning attacks. *arXiv preprint arXiv:1803.06975*, 2018.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

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.

Tramèr, F., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017.

Turner, A., Tsipras, D., and Madry, A. Clean-label backdoor attacks, 2019. URL https://people.csail.mit.edu/madry/lab/cleanlabel.pdf.

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.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.