Bullseye Polytope: A Scalable Clean-Label Poisoning Attack with Improved Transferability

Hojjat Aghakhani, Dongyu Meng, Yu-Xiang Wang, Christopher Kruegel, and Giovanni Vigna

University of California, Santa Barbara

{hojjat, dmeng, yuxiangw, chris, vigna}@cs.ucsb.edu

Abstract—A recent source of concern for the security of neural networks is the emergence of clean-label dataset poisoning attacks, wherein correctly labeled poison samples are injected into the training dataset. While these poison samples look legitimate to the human observer, they contain malicious characteristics that trigger a targeted misclassification during inference. We propose a scalable and transferable clean-label poisoning attack against transfer learning, which creates poison images with their center close to the target image in the feature space. Our attack, Bullseye Polytope, improves the attack success rate of the current state-of-the-art by 26.75% in end-to-end transfer learning, while increasing attack speed by a factor of 12. We further extend Bullseye Polytope to a more practical attack model by including multiple images of the same object (e.g., from different angles) when crafting the poison samples. We demonstrate that this extension improves attack transferability by over 16% to unseen images (of the same object) without using extra poison samples.

I. INTRODUCTION

Machine-learning-based systems are being increasingly deployed in security-critical applications, such as face recognition [28], [41], fingerprint identification [45], and cybersecurity [40], as well as applications with a high cost of failure such as autonomous driving [6]. The possibility of generating adversarial examples in deep neural networks has raised serious doubt on the security of these systems [10], [2], [43]. In these *evasion* attacks, a targeted input is perturbed by imperceptible amounts at test time to trigger misclassification by a trained network. But neural networks are also vulnerable to malicious manipulation during the *training* process. As neural networks require large datasets for training, it is common practice to use training samples collected from other, often untrusted, sources (e.g., the Internet), and it is expensive to have these datasets carefully vetted. While neural networks are strong enough to learn powerful models in the presence of *natural* noise, they are vulnerable to carefully crafted *malicious* noise introduced deliberately by adversaries. In particular, gathering data from untrusted sources makes neural networks susceptible to *data poisoning attacks*, where an adversary injects data into the training set to manipulate or degrade the system performance.

Our work focuses on *clean-label poisoning attacks*, a branch of poisoning attacks wherein the attacker does not have any control over the labeling process. In this threat model, the poison samples are created by introducing imperceptible (yet malicious) alterations that will result in model misbehavior in response to specific target inputs. These perturbations are small enough to justify the original images' labels in the eye of a domain expert. The stealth of the attack increases its success
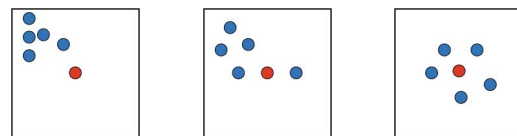
(a) Original images. (b) Convex Polytope (c) Bullseye Polytope

Fig. 1: Simplified representation of poison samples in a two-dimensional feature space. The blue circles are poison samples and the red circle is the target. Convex Polytope moves poison samples until the target is inside their convex hull, making no further refinements to move the target away from the polytope boundary, whereas Bullseye Polytope enforces that the target resides close to the center.

rate in real-world scenarios compared to other types of data-poisoning attacks, as the poison data (1) will not be identified by human labelers, and (2) does not degrade test accuracy except for misclassification of particular target samples.

Clean-label poisoning on *transfer learning* was first studied in a *white-box* setting [35], where the attacker leverages complete knowledge of the pre-trained network ϕ that the victim employs to either (1) extract features for training a (linear) classifier (*linear transfer learning*) or (2) fine-tune on a similar task (*end-to-end transfer learning*). The *Feature Collision* attack [35] selects a base image x_b from the intended misclassification class, and creates a poison sample, x_p, by adding small (bounded) adversarial perturbations to x_b that brings it close to the target image x_t in the feature space, i.e., $\phi(x_t) \approx \phi(x_p)$. This triggers misclassification of x_t to the targeted class by any linear classifier that is trained on the features of a dataset containing x_p. This approach fails when the feature extractor ϕ is unknown to the attacker. To mitigate such limitation, Zhu et al. proposed *Convex Polytope* [50], which, instead of finding poison samples close to the target, finds a set of poison samples that form a convex polytope around it, increasing the probability that the target lies within (or at least close to) this "attack zone" in the victim's feature space. Convex Polytope relies on the fact that every *linear* classifier that classifies a set of points into label l will classify every point in the convex hull of these points as label l.

As we will show later, Convex Polytope suffers from one inherent flaw. The target feature vector tends to be close to the boundary of the attack zone, potentially hampering the attack transferability. Furthermore, the Convex Polytope algorithm is very slow. For example, crafting a set of five poison samples

159

for a single target takes $\sim$17 GPU-hours on average.

To address these limitations, we propose Bullseye Polytope, which refines the constraints of Convex Polytope such that the target is pushed toward the "center" of the attack zone (i.e., the convex hull of poison samples). The geometrical comparison of Bullseye Polytope and Convex Polytope is shown in Figure 1. Bullseye Polytope improves both the transferability and speed of the attack. When the victim adopts linear transfer learning, our method improves the attack success rate by 7.44% on average, while being 11x faster. In end-to-end transfer learning, Bullseye Polytope outperforms Convex Polytope by 26.75% on average, while being 12x faster. For some victim models, the attack success rate of Bullseye Polytope is $\sim 50\%$ higher than Convex Polytope. In a weaker threat model, where the adversary has limited knowledge of the training set of the victim's feature extractor ϕ, Bullseye Polytope provides a 9.27% higher attack success rate in linear transfer learning.

We also extend Bullseye Polytope to a more *practical* threat model. Current clean-label poisoning attacks are designed to target only one image at a time, rendering them ineffective against unpredictable variations in real-world image acquisition. Such attacks disregard the following major point: to succeed in real-world scenarios, the attack needs to cope with a spectrum of test inputs. By including a larger number of target images (of the same object) when crafting the poison samples, we are able to obtain an attack transferability of 49.56% against *unseen* images (of the same object), without increasing the number of poison samples. This is an improvement of over 16%, compared to the single-target mode, when testing against the same set of images (in linear transfer learning).

We further evaluate Bullseye Polytope against l_2-norm centroid and Deep k-NN defenses [29], which are shown to be effective against poisoning attacks on transfer learning. These defenses employ neighborhood conformity tests to sanitize the training data. Our evaluation shows that Bullseye Polytope is much more resilient than Convex Polytope against less aggressive defense configurations. To completely mitigate the attacks, Deep k-NN and l_2-norm centroid defenses need to remove 5% and 10% of the training data, respectively, of which 1% are the poison data. We show that increasing the number of poison samples makes the l_2-norm centroid defense completely ineffective, as it needs to aggressively prune the dataset, which, in turn, degrades the model's performance. This gives our attack a major advantage, as, unlike Convex Polytope, Bullseye Polytope can incorporate more poison samples into the attack process, with virtually no cost in attack-execution time. As we will show later, Convex Polytope scales poorly as the number of poison samples increases. In particular, running the Convex Polytope attack for 800 iterations to craft ten poison samples takes 603 minutes on GPU, while Bullseye Polytope takes only seven minutes.

The Deep k-NN defense is able to completely mitigate the attack by increasing the neighborhood size until poison samples cannot become a majority, but it suffers from low detection precision (20%). On the other hand, if the number of poison samples is larger than the number of samples in the target object's original class, the majority test can be overwhelmed, leaving many poison samples undetected. Furthermore, in some applications, the target object does not belong to one of the classes in the training set, but rather is an unclassified object that the adversary aims to "smuggle in." In this case, poison samples are not likely to have nearby neighbors in the fine-tuning set from a single class other than the poison class. Therefore, to fully mitigate the attack, the Deep k-NN defense needs to adopt a much larger neighborhood size, which results in discarding a higher number of clean samples.

Concurrent to our work, a recent study was published on arXiv [34]. That study develops standardized benchmarks for data poisoning and backdoor attacks to promote fair comparison. Interestingly, the authors already include our work as presented in this paper. The results for linear transfer learning settings demonstrate that Bullseye Polytope outperforms all other attacks. Especially in the white-box setting, the independent third-party study showed that our attack achieved more than 50% higher success rates across experiments compared to the runner-up. The study also benchmarks *from-scratch training scenarios*, where the victim's network is trained from random initialization on the poisoned dataset. This is a much more challenging scenario for attacks that are designed for transfer learning settings (like Bullseye Polytope). However, it is a scenario that is specifically taken into consideration by another attack, *Witches' Brew* (WiB) [8], which was also recently published on arXiv (and parallel to this work). The from-scratch benchmarks are evaluated on two datasets: CIFAR-10 [17] and TinyImageNet [18]. On the former dataset, WiB demonstrated a success rate of 26%, while all other attacks (including Bullseye Polytope) succeeded less than 3% of the time. Interestingly, however, for the TinyImageNet benchmark, our attack achieved the highest success rate (44%), 12% higher than the runner-up (WiB), while other attacks failed most of the times.

To some readers, Bullseye Polytope might appear as a simple extension of prior work, such as Convex Polytope. We argue that this would be myopic — compelling ideas often appear simple in hindsight. Our experiments show that Bullseye Polytope is not only more successful than current state-of-the-art poisoning attacks on transfer learning, but, perhaps more importantly, it is also an order of magnitude faster. This performance improvement is significant, as it unlocks our practical ability to build defenses against this class of attacks with higher detection precision. When creating solutions to detect poisoning attacks, researchers have to experiment with ideas and parameters and perform statistical evaluations. These experiments take a significant amount of time, even when deploying substantial amounts of resources in the cloud. The proposed technique in this paper cuts down this time by a factor of 10, enabling a much faster cycle of experimentation. We also make all source code as well as poison samples available, which can be found at `github.com/ucsb-seclab/BullseyePoison`.

II. THREAT MODEL

In our threat model, we assume that the victim employs transfer learning, where a model trained for one task is reused as part of a different model for a second task. Transfer learning is shown to be a common practice, as it obtains high-quality models without incurring the cost of training a model from scratch [11]. We consider two transfer learning approaches that the victim may adopt; *linear transfer learning* and *end-to-end transfer learning*. In the former, a pre-trained but **frozen** network acts as a feature extractor ϕ, and an application-specific **linear** classifier is fine-tuned on $\phi(\Gamma)$, where Γ is the *fine-tuning training set*. In end-to-end transfer learning, the feature extractor and linear classifier are trained jointly on Γ, and, therefore, the feature extractor is altered during fine-tuning. In both scenarios, the attacker injects a small number of poison samples into Γ, obtained by imperceptibly perturbing some of the original samples. The attacker does not have any control over the labeling process, therefore, the poison samples remain correctly labeled according to their original class. We consider both *black-box* and *gray-box* settings. The attacker has no access to the victim model in the black-box setting. In the gray-box setting, only the victim network's architecture is known. We assume that the attacker knows the training set that is used to build ϕ.[1] The attacker uses this training set for training *substitute networks*, which will be used to craft poison samples. Unless explicitly stated, by "attack transferability" we mean the transferability of the poison samples' characteristics (i.e., targeted misclassification) to the victim's (fine-tuned) model. We do further evaluation in more limited settings where the adversary has no or partial knowledge of the *training set* of ϕ.

III. RELATED WORK

Data Poisoning Attacks. A well-studied portion of data-poisoning attacks aims to use malicious data to degrade the test accuracy of a model [24], [3], [46], [20], [4]. While such attacks are shown to be successful, they are easy to detect, as the performance of a model can always be assessed by testing the model on a private, trusted set of samples. Another important branch of data-poisoning attacks, known as *backdoor attacks* [11], fools models by imprinting a small number of training examples with a specific pattern (*trigger*) and changing their labels to a different target label. During inference, the attacker achieves misclassification by injecting the trigger into targeted examples. This strategy relies on the assumption that the labels of the poison data will not be inspected. To avoid injecting wrong labels, clean-label [44] and hidden-trigger [32] backdoor attacks are proposed, where poison samples are crafted with optimization procedures. In general, similar to evasion attacks, backdoor attacks present the following shortcoming: they require the modification of test samples during inference to enable misclassification.

Clean-label Poisoning Attacks. A recent branch of data-poisoning attacks has no control over the labeling process.

The first clean-label poisoning attack is Feature Collision [35], which mainly targets linear transfer learning, where the adversary has complete knowledge of the feature extractor network ϕ employed by the victim. Feature Collision suffers from one major problem; it tends to fail in black-box settings [50]. To mitigate such limitations, Zhu et al. proposed the Convex Polytope attack [50], which crafts a set of poison samples that contain the target's feature vector within their convex hull. In particular, this attack outperforms Feature Collision by 20% on average in terms of success rate. As we will show in Section VI, Convex Polytope suffers from two shortcomings; (1) *Speed*: Convex Polytope is significantly slow. (2) *Robustness*: The target's feature vector tends to be close to the boundary of the polytope formed by the poison samples, leaving the full potential for attack transferability untapped.

To mitigate such limitations, we design Bullseye Polytope by crafting poison samples centered around the target image in the feature space. As we will show later, our attack accelerates poison construction by an order of magnitude compared to Convex Polytope, while achieving higher attack success rates in both transfer learning setups. We further improve the attack robustness by incorporating multiple images of a target object. Current clean-label poisoning attacks are designed to target only one image at a time, rendering them ineffective against unpredictable variations in real-world image acquisition. We show that the resulting attack is effective on *unseen* images of the target while maintaining good baseline test accuracy on non-targeted images. To the best of our knowledge, Bullseye Polytope is the first clean-label poisoning attack being proposed for a multi-target threat model, which is an important feature for practical implementations on real-world systems.

Concurrent to our work, a recent paper [8] – published on arXiv – proposed a clean-label poisoning attack, named WiB, against from-scratch training scenarios, where the victim's model is trained from random initialization on the poisoned dataset. Such a setting is more challenging for previous clean-label poisoning attacks and Bullseye Polytope, as they are designed for transfer learning scenarios. However, as we will show later, our attack outperforms WiB in some experiments. This is quite interesting, as unlike WiB, Bullseye Polytope is not originally designed for from-scratch training scenarios.

Defenses Against Clean-label Poisoning. Parallel to this work, a recent study by Peri et al. [29] proposed defenses against clean-label poisoning attacks, i.e., Feature Collision [35] and Convex Polytope [50]. They adopted defenses that are shown to be effective against both evasion and backdoor attacks [27], [39].[2] For the Feature Collision attack, they observed that a Deep k-NN based method applied to the penultimate layer (i.e., the feature layer) of the neural network outperforms other types of defenses, such as adversarial training or l_2-norm centroid defenses. In the Convex Polytope attack, Deep k-NN and l_2-norm centroid defenses demonstrate comparable resilience, however, the Deep k-NN defense removes fewer

---

[1]Note that the attacker has no knowledge of Γ (other than the added poisons).

[2]A detailed discussion of defenses against evasion and backdoor attacks is provided in the Appendix F.

clean samples from the training data. In this work, we evaluate Bullseye Polytope and Convex Polytope against both Deep k-NN and l_2-norm centroid defenses. As we will show in Section VI-D, Bullseye Polytope is generally more robust than Convex Polytope against less aggressive defense configurations.

IV. BACKGROUND

As discussed earlier, Feature Collision fails when the victim's feature extractor ϕ is unknown to the attacker. To mitigate such limitation, Zhu et al. [50] proposed Convex Polytope (CP), which crafts a set of poison samples that contain the target within their convex hull. CP exploits the following mathematical guarantee: if the victim's linear classifier associates the poison samples with the targeted class, it will label any point inside their convex hull as the targeted class. CP creates a larger "attack zone" in the feature space, thus increasing the chance of transferability, as argued by the authors. In particular, CP solves the following optimization problem:

$$\underset{\{c^{(i)}\},\{x_p^{(j)}\}}{\text{minimize}} \frac{1}{2m} \sum_{i=1}^{m} \frac{\left\| \phi^{(i)}(x_t) - \sum_{j=1}^{k} c_j^{(i)} \phi^{(i)}(x_p^{(j)}) \right\|^2}{\left\| \phi^{(i)}(x_t) \right\|^2}$$

$$\text{subject to } \sum_{j=1}^{k} c_j^{(i)} = 1, c_j^{(i)} \geq 0, \forall i, j,$$

$$\left\| x_p^{(j)} - x_b^{(j)} \right\|_\infty \leq \epsilon, \forall j, \quad (1)$$

where $x_b^{(j)}$ is the original image of the j-th poison sample, and ϵ determines the maximum allowed perturbation. Eq. 1 finds a set of poison samples $\{x_p^{(j)}\}_{j=1}^{k}$ such that the target x_t lies inside, or at least close to, the convex hull of the poison samples in the feature spaces defined by m substitute networks $\{\phi^{(i)}\}_{i=1}^{m}$. In the i-th substitute network, the target feature vector $\phi^{(i)}(x_t)$ is ideally a convex combination of the feature vectors of poison images, i.e., $\phi^{(i)}(x_t) = \sum_{j=1}^{k} c_j^{(i)} \phi^{(i)}(x_p^{(j)})$, where $c_j^{(i)}$ determines the j-th poison's coefficient. To solve the non-convex problem in Eq. 1 (i.e., find the optimal poison samples), CP repeats the following steps for 4,000 iterations:

1) Freezing $\{x_p^{(j)}\}_{j=1}^{k}$, use forward-backward splitting [9] to optimize the coefficients for each individual network $\{c^{(i)}\}$.
2) Given $\{c^{(i)}\}$, optimize $\{x_p^{(j)}\}_{j=1}^{k}$ using one gradient step.
3) Clip $\{x_p^{(j)}\}_{j=1}^{k}$ to the ϵ-ball around the base images $\{x_b^{(j)}\}_{j=1}^{k}$.

Poor Scalability of Convex Polytope. We observed that when using 18 substitute networks, solving Eq. 1 for five poison samples takes $\sim$17 GPU-hours on average.[3] Of this time, step one alone takes $\sim$15 hours. We list the details of step one in the Appendix (Algorithm 1). Within this process, we noticed two major time-consuming operations: (1) checking whether the new coefficients result in a smaller loss compared to the old coefficients (this is done in every iteration of coefficient optimization), and (2) projection onto the probability simplex,

[3]This is the exact same setting used in the original paper [50].

which happens whenever the new coefficients satisfy the above condition. While we believe that there is room for improvement of this algorithm, e.g., by checking the condition every few steps rather than each step, we did not make any such changes in order to avoid degradation of the attack success rate, and to allow for a fair comparison.

V. BULLSEYE POLYTOPE

Apart from scalability, CP has an inherent flaw: as soon as the target crosses the boundary into the interior of the convex polytope, there is no incentive to refine further and move the target deeper inside the attack zone (Figure 1). Therefore, the target will lie close to the boundary of the resulting poison polytope, which reduces robustness and generalizability. We design Bullseye Polytope (BP) based on the insight that, by fixing the relative position of the target with respect to the poison samples' convex hull, we speed up the attack while also improving its robustness. Instead of searching for coefficients by optimization, which is neither efficient nor effective, BP *predetermines* the k coefficients as equal, i.e., $\frac{1}{k}$, to enforce that the target resides close to the "center" of the poison samples' polytope.[4] BP then solves the special case of:

$$\underset{\{x_p^{(j)}\}}{\text{minimize}} \frac{1}{2m} \sum_{i=1}^{m} \frac{\left\| \phi^{(i)}(x_t) - \frac{1}{k} \sum_{j=1}^{k} \phi^{(i)}(x_p^{(j)}) \right\|^2}{\left\| \phi^{(i)}(x_t) \right\|^2}$$

$$\text{subject to } \left\| x_p^{(j)} - x_b^{(j)} \right\|_\infty \leq \epsilon, \forall j. \quad (2)$$

As we show later, BP indeed improves attack transferability by effectively pushing the target toward the center of the attack zone. Also, by precluding the most time-consuming step of computing coefficients, BP is an order of magnitude faster than CP. It should be noted that, while BP seems to be a special case of CP, the objective loss of Eq. 2 has a significant difference with respect to Eq. 1. That is, the closer the target gets to the polytope's center, the smaller the loss becomes, which is not true for Eq. 1. For this reason, the solution of Eq. 2 (BP) is not necessarily a special case of Eq. 1 (CP), since an optimizer that uses Eq. 1 might never find such a solution. Although CP initially sets the k coefficients as equals (i.e., $\frac{1}{k}$), we observed that the coefficients become skewed from the very beginning. This happens because at each step of optimizing the coefficients, the solution of Eq. 1 is skewed towards poison samples that are closer to the target.

Kernel Embedding-view of Bullseye Polytope. Besides improved computational efficiency, our approach in Eq. 2 can be viewed as optimizing *distribution* of poison samples via its mean embedding. Informally speaking, when ϕ is a sufficiently descriptive feature map,[5] then $\mathbb{E}_{x \sim P}[\phi(x)] = \mathbb{E}_{x \sim Q}[\phi(x)]$ *if and only if* distributions P and Q are identical (see, e.g., [38, Theorem 1]; also see a recent survey of kernel mean embedding [22]). Deep neural networks are closely related

[4]Our notion of center coincides with the *center of mass* of the poison set.
[5]For example, $\phi(x) = k(x, \cdot)$ for a *characteristic* reproducing kernel k, e.g., the Gaussian-RBF kernel $k(x, y) = e^{-\|x-y\|^2}$.

to kernel methods [23], [31], [15]. The pre-trained network is a powerful feature extractor, thus is often viewed as an even better descriptor of the input feature x than kernels for prediction purposes. As a result, if $\{x_p^{(j)}\}_{j=1}^k$ are drawn from a distribution P, then Eq. 2 is *essentially* optimizing this distribution using the *plug-in estimator* of mean embedding: $\frac{1}{k}\sum_{j=1}^k \phi(x_p^{(j)})$.

Deep Sets. Bullseye Polytope is also backed by the more recent approach of *deep sets* [49], which establishes that for *any* function f of a set of poison samples $x_p^{(1)}, ..., x_p^{(k)}$ that enjoys *permutation invariance* admits a decomposition: $f = \rho(\frac{1}{k}\sum_{j=1}^k \phi(x_p^{(j)}))$ for some function ρ, ϕ. Notice that due to the random reshuffling steps in training machine learning models, the learned prediction function (i.e., classifier) is *permutation invariant* by construction with respect to the training dataset (containing the set of poison samples). That is, the classifier's prediction f can be decomposed to $\rho(\frac{1}{k}\sum_{j=1}^k \phi(x_p^{(j)}))$ — a function of the mean embedding. Thus, our simplification from Eq. 1 to Eq. 2 that optimizes the mean embedding rather than a more general convex combination is arguably *without loss of generality* (See Appendix G for a more detailed discussion).

A. Improved Transferability via Multi-Draw Dropout

Attack transferability improves when we increase the number of substitute networks for crafting poison samples. While it is impractical to ensemble a large number of networks due to memory and time constraints, introducing dropout randomization provides some of the diversification afforded by a larger ensemble. With dropout, the substitute network $\phi^{(i)}$ provides a different feature vector for the same poison sample each time. This randomization was observed to result in a much higher variance in the (training) loss of Eq. 2 compared to that of Eq. 1. Since the solution space of Eq. 2 is much more restrictive than Eq. 1, and moves around for different realizations of dropout, gradient descent has a harder time converging for Eq. 2. We use averaging over multiple draws to alleviate this problem. In each iteration, we compute the feature vector of poison samples R times for each network, and use their average in optimizing Eq. 2. Of course, increasing R results in higher attack execution time, but even a modest choice of $R=3$ is enough to achieve an 8.5% higher success rate compared to when $R=1$ is used for end-to-end transfer learning. Even in this case, BP is 12 times faster than CP.

B. Multi-target Mode

We further improve the robustness of BP by incorporating multiple images of a target object. This is similarly achieved by simply replacing $\phi(x_t)$ in (2) with a *mean embedding of the distribution* of the targets $\frac{1}{N_{\text{im}}}\sum_{j=1}^n \phi(x_t^{(j)})$ where $x_t^{(1)}, ..., x_t^{(N_{\text{im}})}$ are drawn i.i.d from a target distribution that captures the natural variations in lighting conditions, observation angles, and other unpredictable stochasticity in real-world image acquisition. To say it differently, instead of attacking one individual instance, we are now attacking a distribution of instances by creating a set of poison samples that *match* the target distribution in terms of the mean embedding as much as possible. In Section VI-B, we demonstrate that the resulting attack is highly effective not only on the "training" instances of the targets but also *generalizes* to *unseen* images of the target, while maintaining good baseline test accuracy on images of non-targeted objects. In contrast, current clean-label poisoning attacks only work with one image at a time, rendering them ineffective in more realistic attack scenarios.

C. End-to-End Transfer Learning

In end-to-end transfer learning, the victim retrains both the feature extractor and the linear classifier, altering the feature space in the process. This causes unpredictability in the attack zone, even in the white-box setting. To tackle this issue, inspired by Zhu et al. [50], we jointly apply BP to multiple layers of the network, crafting poison samples that satisfy Eq. 2 on the feature space created by each layer. This adds to the complexity of the problem, which is especially problematic for the already slow CP algorithm.

VI. EXPERIMENTS

We first evaluate BP in single-target mode and compare against CP, and then demonstrate its transferability on unseen images of the target object (multi-target mode). BP-3x and BP-5x represent the case where multi-draw dropout is enabled, with R set to 3 and 5, respectively. Unless stated otherwise, we use the same settings as used by Zhu et al. [50] to provide a fair comparison. We also study the effect of the perturbation budget ϵ and the number of poison samples on the attack success rate through ablation studies. Furthermore, we evaluate both BP and CP against defenses that are proposed by a recent study [29]. In the end, we further evaluate BP using standard benchmarks that are developed in a recent study [34]. We ran all the attacks using `NVIDIA Titan RTX` graphics cards.

A. Single-target Mode

Datasets. We use the CIFAR-10 dataset. If not explicitly stated, all the substitute and victim models are trained using the first 4,800 images from each of the 10 classes. In all experiments, we use the standard test set from CIFAR-10 to evaluate the *baseline test accuracy* of the poisoned models and compare them with their unpoisoned counterparts. The attack targets, base images of poison samples, and victim's fine-tuning set are selected from the remaining 2,000 images of the dataset. We assume that the victim models are fine-tuned on a training set consisting of the first 50 images from each class, i.e., the *fine-tuning dataset*, containing a total of 500 images. Zhu et al. [50] randomly selected "ship" as the misclassification class, and "frog" as the target's image class. We assume the same choice for comparison fairness. Specifically, the attacker crafts clean-label poison samples from ship images to cause a particular frog image to be misclassified as a ship. We craft the poison images $x_p^{(j)}$ from the first five images of the ship class in the fine-tuning dataset. We run CP and BP attacks with 50 different target images of the frog class (indexed from 4,851 to 4,900)
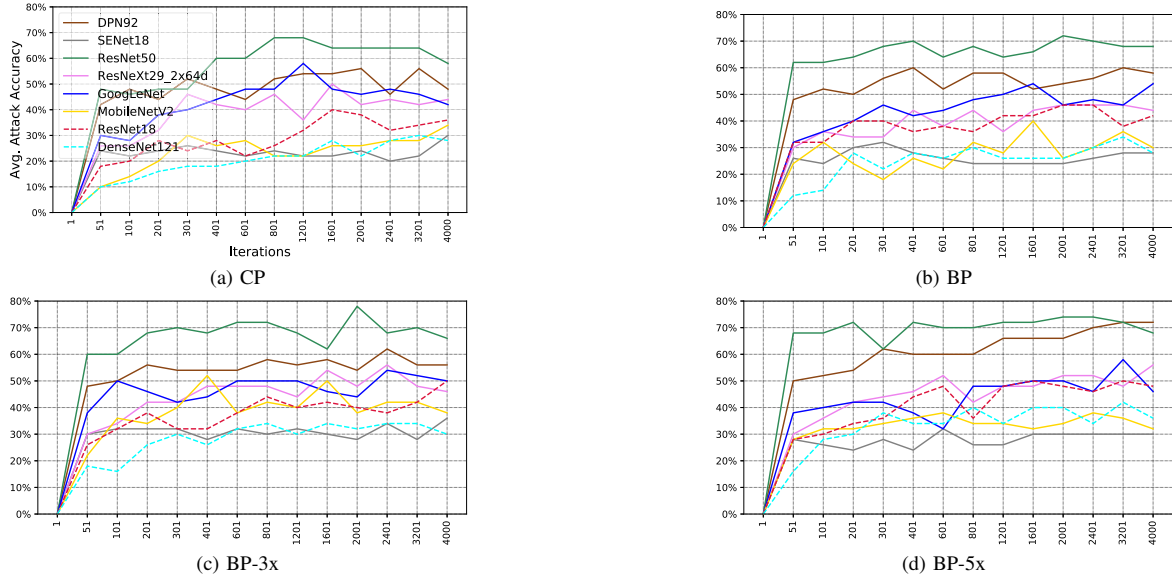
Fig. 2: Linear transfer learning - success rates of CP, BP, BP-3x, and BP-5x on victim models. Notice `ResNet18` and `DenseNet121` are the black-box setting.
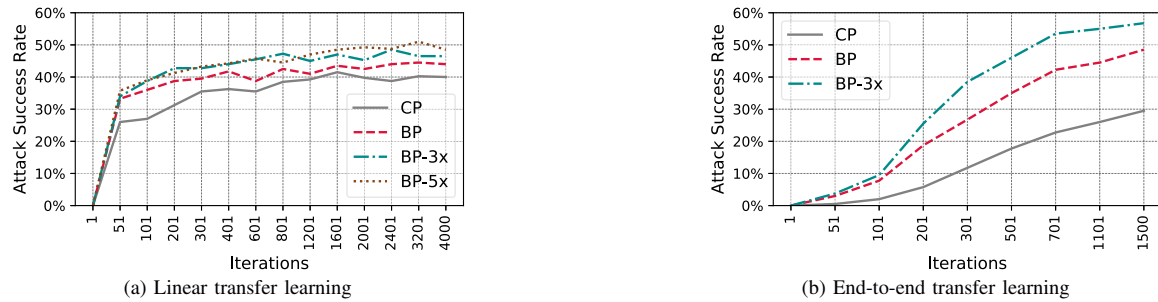

Fig. 3: Attack success rates of CP, BP, BP-3x, and BP-5x, averaged over all eight victim models.

to collect performance statistics. Thus we ensure that target images, training set, and fine-tuning set are mutually exclusive subsets. We set an ℓ_∞ perturbation budget of $\epsilon = 0.1$.

Linear Transfer Learning. For substitute networks, we use SENet18 [13], ResNet50 [12], ResNeXt29-2x64d [48], DPN92 [7], MobileNetV2 [33], and GoogLeNet [42]. Each network architecture is trained with dropout probabilities of 0.2, 0.25, and 0.3, which results in a total of 18 substitute models. To evaluate the attacks under gray-box settings, we use the aforementioned architectures (although trained with a different random seed). For black-box settings, we use two new architectures, ResNet18 [12] and DenseNet121 [14]. Dropout remains activated when crafting the poison samples to improve attack transferability. However, all eight victim models are trained without dropout, and dropout is disabled during evaluation. We perform both CP and BP for 4,000 iterations with the same hyperparameters used by CP. The only difference is that BP forces the coefficients to be uniform, i.e., $c_j^{(i)} = \frac{1}{5}$.

We use Adam [16] with a learning rate of 0.1 to fine-tune the victim models on the poisoned dataset for 60 epochs.

Figure 2 shows the progress of CP, BP, BP-3x, and BP-5x over the number of iterations of the attack against each individual victim model. Figure 3a shows the attack progress wherein the attack success rate is averaged over eight victim models. In general, BP outperforms CP and converges faster. In particular, on average over all iterations, BP-3x and BP-5x demonstrate 7.44% and 8.38% higher attack success rates than CP. Both CP and BP hardly affect the baseline test accuracy of models (Figure 6a).[6] BP is almost 21 times faster than CP, as it excludes the computation-heavy step of optimizing the coefficients. Figure 5a shows the attack execution time based on the number of iterations. Running CP for 4,000 iterations takes 1,002 minutes on average, while BP takes only 47 minutes. BP-3x and BP-5x take 88 and 141 minutes, respectively. It is worth noting that BP needs fewer iterations than CP to achieve

---

[6]BP has slightly less severe effect on the baseline test accuracy.
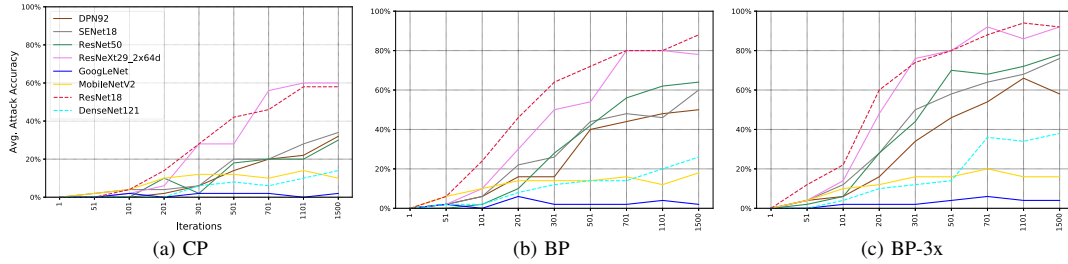
(a) CP   (b) BP   (c) BP-3x

Fig. 4: End-to-end transfer learning - success rates of CP, BP, and BP-3x on victim models. Notice `MobileNetV2`, `GoogLeNet`, `ResNet18` and `DenseNet121` are the black-box setting.
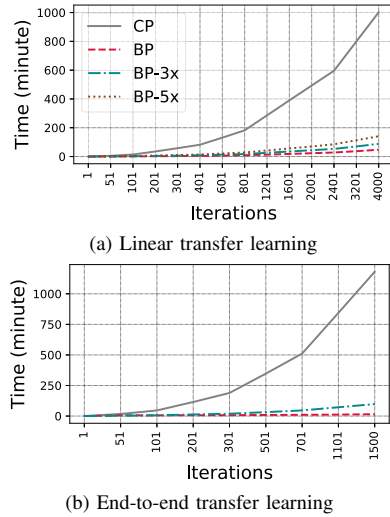

(a) Linear transfer learning

(b) End-to-end transfer learning

Fig. 5: Attack execution time.

(a) Linear transfer learning

(b) End-to-end transfer learning

Fig. 6: Average baseline test accuracy variation.

the same attack success rate for some victim models (Figure 2).

End-to-end Transfer Learning. In this mode, the victim feature extractor is altered during the fine-tuning process, which results in a (slightly) different feature space. This causes the conventional CP attack to have a success rate of less than 5%. To tackle this problem, CP creates convex polytopes in different layers of the substitute models. We follow the same strategy for BP, this time limiting each attack to 1,500 iterations to meet time and resource constraints. For substitute networks, we use SENet18, ResNet50, ResNeXt29-2x64d, and DPN92, with dropout values of 0.2, 0.25, and 0.3 (a total of 12 substitute models). For gray-box testing, we evaluate the attacks against these four architectures. In the black-box setting, MobileNetV2, GoogLeNet, ResNet18, and DenseNet121 are used as victim networks. We use Adam with a learning rate of 10^{-4} to fine-tune the victim models on the poisoned dataset for 60 epochs.

Similar to what we observed for linear transfer learning, but with a wider margin, BP presents higher attack transferability than CP, especially in the black-box setting. Figure 4 shows the progress of CP, BP, and BP-3x over the number of iterations of the attack against each individual victim model. Here we report attack success rates after 1,500 iterations. BP and BP-3x
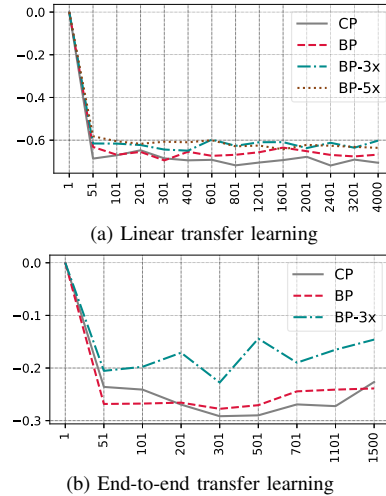
improve average attack transferability (over victim models) by 18.25% and 26.75%, respectively (Figure 3b). Figure 20 in the Appendix shows attack success rates against each individual victim model. BP and BP-3x have 10-30% and 10-50% higher attack transferability than CP, respectively (except against GoogLeNet). Poor transferability against GoogLeNet is also reported for CP [50]. Since the GoogLeNet architecture differs significantly from the substitute models, it is, therefore, more difficult for the "attack zone" to survive end-to-end transfer learning. For other black-box models (MobileNetV2, ResNet18, and DenseNet121), BP and BP-3x improve attack transferability by ∼18% and ∼24%, respectively. Both CP and BP have hardly any effect on the baseline test accuracy of models (Figure 6b). As Figure 5b shows, BP and BP-3x take 15 and 98 minutes, while CP takes 1,180 minutes, which is 36x slower.

It is worth noting that we found multi-draw dropout not beneficial to CP in the experiments. Since using multi-draw dropout makes CP (much) slower, with no gain in attack success rate, to fairly compare the execution time of BP with CP, multi-draw dropout is always disabled for CP.

Transferability to Unseen Training Sets. Until now, we have assumed that the substitute models are trained on the same

165

Fig. 7: Comparison of CP, BP, and BP-3x in linear transfer learning, with zero and 50% overlap between training sets of the substitute networks and the victim's network.

training set (Ψ) on which the victim's feature extractor network is trained. In this section, we evaluate CP and BP using substitute models that are trained on a training set that has (1) **zero** or (2) **50%** overlap with Ψ. Such a setting is more realistic compared to when the attacker has complete knowledge of Ψ. We use the same setting as in linear transfer learning except for the following changes: (i) We train the victim models on the first 2,400 images of each class; (ii) In the zero overlap setting, we train substitute models on samples indexed from 2,401 to 4,800 for each class; (ii) For the 50% overlap setting, we train substitute models on samples indexed from 1,201 to 3,600 for each class. Figure 7 shows the attack success rates (averaged over victims) for both zero overlap and 50% overlap setups. When we have 50% overlap, BP, BP-3x, and BP-5x demonstrate 5.82%, 8.56%, and 9.27% higher attack success rates compared to CP (on average over all iterations), with BP converging significantly faster than CP. For the zero overlap setup, BP provides hardly any improvement over CP. They both achieve much lower attack success rates of 20-25%. It should be noted that the zero overlap scenario is much more restricted than what is usually assumed in threat models for poisoning attacks. The victim's network, training set, and even the fine-tuning training set (except for, of course, the poison samples) are all unseen to the adversary. All attacks hardly affect the baseline test accuracy (Figure 17 in the Appendix).

Effectiveness of the Bullseye Idea. We have argued that the effectiveness (robustness and transferability) of BP stems from the fact that predetermining the convex coefficients as uniform weights draws the target to the "center" of the attack zone, increasing its distance from the poison polytope boundary. In order to evaluate this claim quantitatively, we run the attack with different sets of *nonuniform* coefficients, to see if the improvement is truly due to target centering (i.e., the "bullseye" idea) or simply from "fixing" the coefficients instead of searching for them. We evaluate BP against nine alternatives $\{\mathrm{BP}'_t\}_{t=1}^9$, each with a different set of positive predefined coefficients that satisfy $\sum_{j=1}^k c_j = 1$. Figure 8 depicts a geometrical example for each set (sorted from left to right based on the entropy of the coefficient vector), with BP having the highest possible entropy of $\log_2 5 \simeq 2.32$. As Figure 9 shows, variations of BP with higher coefficient entropy generally demonstrate higher attack success rates compared to

those with smaller entropy, especially in the black-box setting. This finding indicates that predetermining the coefficients to uniform weights (BP) is preferable to simply fixing them to some other plausible values. This backs our intuition behind BP that the further the target is from the polytope boundary, the lower its chances of jumping out of the attack zone in the victim's feature space. In fact, the average entropy of coefficients in CP roughly converges to 1.70, which means the coefficient distribution is more skewed, with some poison samples having a relatively small contribution to the attack. Figure 10 shows the mean values of the (sorted) coefficients to provide a sense of the coefficient distributions used by CP.

Different Pairs of <original class, poison class>. Until now, the adversarial goal in all experiments was to make the victim's model identify an image of a frog (original class) as a ship (poison class). For comparison fairness, we have followed Zhu et al. [50] for this selection of the original and poison classes. To assess the impact of selecting different original and poison classes on the performance of the attack, we evaluate BP-3x for all 90 pairs of <original class, poison class>; each with 5 different target images (indexed from 4,851 to 4,855 in the original class), resulting in a total of 450 attack instances. We focus on linear transfer learning and limit each attack to 800 iterations to meet time and resource constraints. On average, against all eight victim networks, BP-3x achieved a success rate of 40.83%. In the original setting of <frog, ship>, BP-3x showed a success rate of 47.25% (Figure 3a). See Appendix D for the attack performance against individual victim networks as well as a comparison of different class pairs.

B. Multi-target Mode

We now consider a more realistic setting where the target *object* is known, but there is unpredictable variability in the target *image* at test time (e.g., unknown observation angles). This is the first attempt at crafting a *clean-label* and *training-time* dataset poisoning attack that is effective on multiple (unseen) images of the target object at test time. To this end, we consider a slight variation of BP that takes multiple images of the target object (capturing as much observation variability as possible), and performs BP on the averages of their feature vectors. We use the Multi-View Car dataset [26], which contains images from 20 different cars as they are rotated by 360 degrees at increments of 3-4 degrees. We expect to
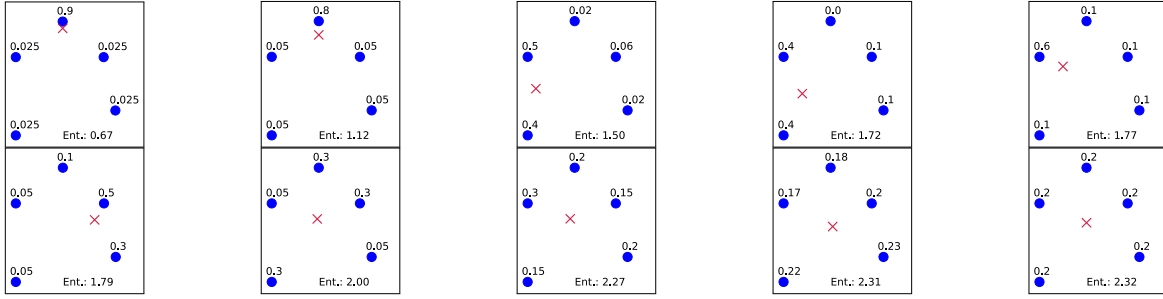
Fig. 8: Nine alternatives of Bullseye Polytope with different sets of nonuniform coefficients. The blue circles are poison samples with their coefficients written next to them, and the red cross is the target. The entropy of the coefficients increases from left to right. Note that the bottom right represents BP.

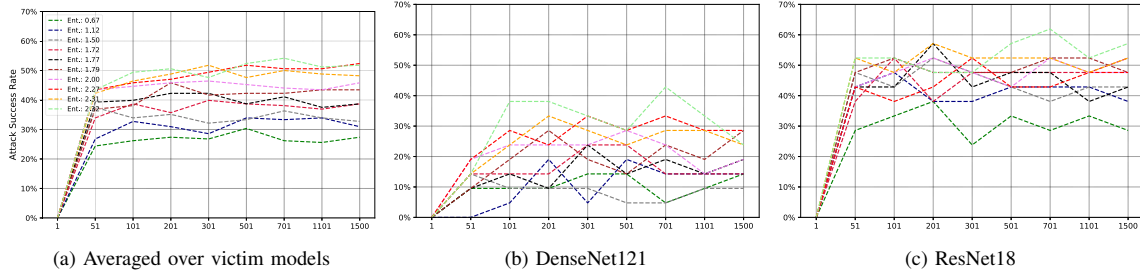(a) Averaged over victim models   (b) DenseNet121   (c) ResNet18

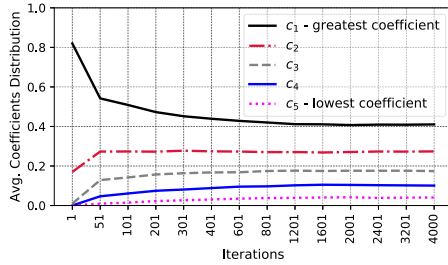Fig. 9: Comparison between BP and the other nine alternatives.

Fig. 10: Distribution of poison samples' coefficients defined in Eq. 1 (averaged over all targets and victim networks). The coefficients are sorted, c_1 denotes the highest coefficient, and c_5 denotes the lowest coefficient. In Convex Polytope, the coefficient distribution is more skewed with some poison samples having a relatively small contribution to the attack.

see lower accuracy when testing the substitute models on the Multi-View Car dataset, as it contains a different distribution of images compared to CIFAR-10. We observed that images from the car dataset are most commonly misclassified as "ship," therefore to avoid contamination from this inherent similarity, this time we choose "frog" as the intended misclassification label, and perform the attacks only for the 14 cars with baseline accuracy of over 90% to obtain pessimistic results. We use the same settings as the single-target mode. We discuss in the Appendix E how the car images of the Multi-View Car dataset are adapted for our models, which are trained on CIFAR-10.

We evaluate both CP and BP setting the number of target images N_{im} to $\{1, 2, 3, 4, 5, 10\}$ to verify the effect of N_{im} on the attack robustness against unseen angles. Note that when $N_{\text{im}} = 1$, the attack is in single-target mode. To select the N_{im} target images, we take one image every $\frac{360}{N_{\text{im}}}$ degree rotation of the target car. Figure 11 and Figure 12 show the attack success rates against **unseen** images. In linear transfer learning, using five targets instead of one improves attack robustness against unseen angles by over 16%. In end-to-end transfer learning, BP-3x demonstrates an improvement of 12%. When $N_{\text{im}} = 5$, BP achieves 14% higher attack success rate compared to CP, while being 59x faster. We emphasize that the total number of poison samples crafted for multi-target attacks is the same as single-target mode (i.e., 5). Figure 19 in the Appendix depicts poison samples crafted for one particular target car.

More Realistic Transfer Learning. We argue that the setting of this (multi-target) experiment is also relevant for another reason: the source of the fine-tuning set is different from the source of the original training set. In particular, we assume that the victim fine-tunes a model – pre-trained on CIFAR-10 – on the Multi-View Car dataset. Our results show that BP achieves comparable success rates in such a more realistic setting. For example, when $N_{\text{im}} = 1$ in end-to-end transfer learning, the attack success rates of BP and CP are 51% and 34%.

C. Attack Budget

Until now, we have used five poison samples with an ℓ_∞ perturbation budget of $\epsilon = 0.1$. Here, we discuss the impact of the number of poison samples and the perturbation amount on the attack success rate. Since we observed the same trend
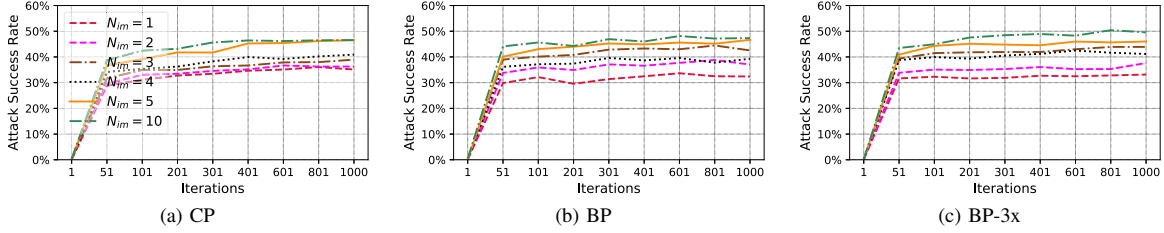
Fig. 11: Attack transferability to unseen angles in linear transfer learning.
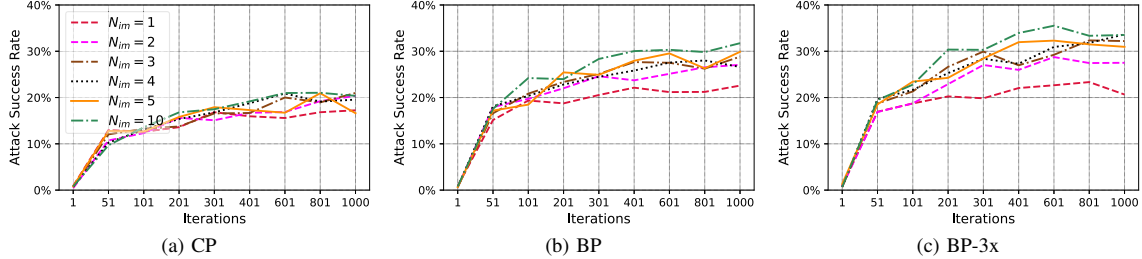

Fig. 12: Attack transferability to unseen angles in end-to-end transfer learning.

TABLE I: Evaluation of BP (after 800 iterations), when different poison budget is used. The first row shows the accuracy that the victim's fine-tuned model classifies poison samples into the poison class label. The second row shows the baseline test accuracy of the model on the standard test set from CIFAR-10. The last row shows the attack success rate.

	# Poisons						Perturbation Budget ϵ					
	3	5	7	10			0.01	0.03	0.05	0.1	0.2	0.3
Poisons Acc. (%)	82.33	84.45	86.57	88.98		Poisons Acc. (%)	96.05	82.25	82.87	84.45	85.4	86.1
Clean Test Acc. (%)	91.92	91.76	91.67	91.60		Clean Test Acc. (%)	92.01	91.69	91.75	91.76	91.80	91.82
Attack Success Rate (%)	28.00	42.50	49.50	57.75		Attack Success Rate (%)	4.50	33.00	40.43	42.50	39.75	43.25

(a) Different number of poisons used ($\epsilon = 0.1$).   (b) Different levels of perturbation ϵ used (# poisons = 5).

for single-target and multi-target mode, we only report the numbers for single-target mode. We limit each attack to 800 iterations to meet time and resource constraints.

Table Ia shows the attack performance of BP, when different numbers of poison samples are injected into the victim's fine-tuning dataset. In general, using more poison samples results in a higher attack success rate, which can be due to two reasons; First, BP achieves a lower "bullseye" loss (Eq. 2) when more poison samples are used. In fact, we confirmed that this is not the case. While in some scenarios, the loss value slightly decreases, generally, across different target samples, the loss does not decrease by simply adding more poison samples. So, if the attack fails to find poison samples shaping a convex polytope around some particular target, increasing the number of poison samples will not help us to find a "better" polytope.

Second, having more poison samples in the fine-tuning dataset will cause the classifier to learn the malicious characteristics of the poison samples with a higher probability. This indeed contributes to a higher attack success rate. During our analysis, we noticed that the main reason for the attack failure for a particular target is the following; In the fine-tuning dataset of the victim, there exist samples from the target's original class that are close "enough" to the target so that the victim's model classifies the target into its true class. In most cases, a few of the poison samples are even classified into the target's

original class, which indeed downgrades the malicious effect of poison samples. Therefore, by adding more poison samples to the fine-tuning dataset, the chance that poison samples in the adjacency of the target outnumber samples from the true class is higher. Note that we do not consider a white-box threat model in this work, thus the convex polytope created for the substitute networks will not necessarily transfer to the victim's feature space, which means the condition of the mathematical guarantee discussed in Section IV will not always hold.

We also evaluate CP when the number of poison samples is ten. As Table II shows, BP demonstrates a 6.5% higher attack success rate than CP. Running BP for 800 iterations takes only seven minutes on average, while CP takes 603 minutes, which is 86 times slower. This happens because CP poorly scales as the number of poison samples increases. In each iteration of solving Eq. 1, CP needs to find the optimal set of coefficients for each poison. If we increase the number of poison samples from five to ten, at each iteration of the attack, ten optimization problems need to be solved to find the best coefficients (instead of five). This is not the case for BP, as increasing the number of poison samples does not necessarily make solving Eq. 2 harder. The problem is still finding the solution of Eq. 2 using backpropagation, with ten poison samples as the parameters, instead of five. In fact, our evaluation shows that BP takes roughly the same time as when we use five poison samples.

TABLE II: Evaluation of BP and CP (after 800 iterations), when ten poison samples are used, and ϵ is set to 0.1.

	BP	CP
Poisons Acc. (%)	88.98	85.20
Clean Test Acc. (%)	91.60	91.43
Attack Success Rate (%)	57.75	51.25
Attack Execution Time (min.)	7	603

Table Ib shows the attack performance of BP, when five poison samples are crafted, yet with different levels of perturbation. In general, the "bullseye" loss does not change for ϵ values greater than 0.05, and increasing ϵ further has a negligible impact on the attack success rate. We argue this happens for the same reason that an attack fails for a particular target when there are some samples from the target class in the victim's fine-tuning dataset that are very close to the target in the victim's feature space. In such a scenario, increasing the perturbation budget is not enough to move the target from the proximity of its class into the attack zone. Due to resource and time constraints, we evaluated CP in these settings on a smaller set of targets, and we have observed a trend similar to what we discussed above.

D. Defenses

Concurrent to this work, a recent study has been published [29], which studies defenses against clean-label poisoning attacks, i.e., Feature Collision [35] and Convex Polytope [50]. In their evaluation, Deep k-NN and l_2-norm centroid defenses generally outperformed other types of defenses, such as adversarial training. In this work, we evaluate both BP and CP against these two defenses. **Deep k-NN Defense:** For each sample in the training set, this defense flags the sample as anomalous and discards it from the training set if the point's label is not the mode amongst the labels of its k nearest neighbors. Euclidean distance is used to measure the distance between data points in feature space. **l_2-norm Outlier Defense:** For each class c, the l_2-norm centroid defense removes a fraction μ of points from class c that are farthest in feature space from their centroid.

It should be noted that both defenses are vulnerable to data-poisoning attacks. In the Deep k-NN defense, a naïve adversary might expand the set of poison samples such that the extra poison samples are close "enough" to the old poison samples, so that more poison samples might survive the k nearest neighbor filtration process. In l_2-norm centroid defense, the position of the centroid can be adjusted towards the poison samples (e.g., by adding more poison samples), especially when the per-class data size is small, which is the case in transfer learning. While clean-label poisoning attacks can be more powerful by considering neighborhood conformity tests when crafting the poison samples, in this work, we assume the adversary does not know that such defenses will be employed by the victim. In particular, we evaluate both BP and CP against these two defenses. In our evaluation, we ran BP and CP against linear transfer learning for 50 different targets, which results in 50 different sets of poison samples. We report

here the aggregated statistics averaged over these sets of poison samples and eight victim models. We also evaluated the attacks when the number of poison samples is increased from five to ten. To meet resource and time constraints, the attacks are limited to 800 iterations.

Table III shows the performance of the Deep k-NN defense against BP and CP for various choices of k. Regardless of how many poison samples are used, the Deep k-NN defense becomes more effective against both attacks as k increases, while eliminating roughly the same number of samples from the training set (i.e., 26 and 31 for when five and ten poison samples are crafted, respectively). BP generally demonstrates much higher resilience compared to CP. For small values of k, the Deep k-NN defense discards fewer poison samples of BP compared to CP. When using five poison samples, setting $k = 1$ is enough to reduce the attack success rate of CP from 37.25% to 6.75%, while BP still achieves an attack success rate of 20.50%, which is 4.75x higher. To completely diminish BP, k needs to be greater than eight, however, when ten poison samples are crafted, the attack success rate decreases only to 31.25%. It is worth noting that CP achieves 1.25% attack success rate in such a configuration.

Table IV presents the performance of the l_2-norm centroid defense against BP and CP for various choices of μ. When five poison samples are used, BP demonstrates a superior resilience against the defense compared to CP for $\mu < 0.1$. For larger values of μ, both attacks are completely thwarted. However, the larger μ is, the more samples are discarded from the training set, which can degrade the model performance on the fine-tuning dataset, and, henceforth, the new task. For example, when $\mu = 0.1$, the l_2-norm centroid defense eliminates five samples from each class of the dataset. This represents 10% of the fine-tuning dataset. Compared to the Deep k-NN defense, the l_2-norm centroid defense tends to eliminate more samples from the dataset to achieve the same level of resilience. In particular, to completely mitigate the attacks, the l_2-norm centroid defense removes 50 samples, while Deep k-NN eliminates 26 samples. When ten poison samples are used, the l_2-norm centroid defense becomes less effective for small values of μ. To completely mitigate the attacks, μ needs to be greater than 0.18. In this setting, the l_2-norm centroid defense removes 90 samples in total, which is 18% of the fine-tuning dataset. For smaller values of μ, BP is more resilient than CP. For example, when $\mu = 0.12$ (i.e., 60 samples to be removed from the victim's dataset), the attack success rate of CP reduces to 20%, while BP demonstrates a 32.50% attack success rate.

In general, BP demonstrates higher attack robustness against Deep k-NN and l_2-norm centroid defenses compared to CP. Both defenses completely mitigate the attacks for high values of k and μ. Increasing the number of poison samples makes the l_2-norm centroid defense ineffective, as it needs to aggressively prune the dataset, which will result in lower performance on the victim's task. This gives BP a major advantage, as unlike CP, BP is able to incorporate more poison samples into the attack process, with virtually no cost in attack-execution time (Table II). On the other hand, the Deep k-NN defense seems

TABLE III: Evaluation of BP and CP (after 800 iterations) when the victim employs the Deep k-NN defense. Note that $k = 0$ means no defense is employed. Five and ten poison samples are used in the left and right table, respectively.

k	# Deleted Poisons		# Deleted Samples		Adv. Success Rate (%)	
	BP	CP	BP	CP	BP	CP
0	-	-	-	-	42.5	37.25
1	3.18	4.28	36.46	37.02	20.50	6.75
2	2.42	3.86	21.91	23.07	24.75	8.00
3	3.81	4.66	27.86	27.87	11.75	1.50
4	3.48	4.60	25.83	26.69	14.75	2.50
6	4.22	4.85	25.39	25.91	8.25	1.25
8	4.77	4.94	25.69	25.80	1.25	0.00
10	4.97	4.95	26.36	26.33	0.00	0.25
12	4.98	4.96	26.58	26.54	0.00	0.00
14	4.98	4.96	26.21	26.21	0.00	0.00
16	4.98	4.96	26.95	26.92	0.00	0.00
18	4.98	4.96	26.36	26.37	0.00	0.00
22	4.98	4.96	26.62	26.59	0.00	0.00

(a) # Poisons = 5

| k	# Deleted Poisons		# Deleted Samples		Adv. Success Rate (%)	
	BP	CP	BP	CP	BP	CP
0	-	-	-	-	57.75	51.25
1	4.30	7.56	38.77	41.22	49.25	14.00
2	2.71	6.38	22.75	25.77	51.75	21.25
3	4.92	8.16	30.36	31.88	38.75	11.00
4	3.94	7.76	26.74	29.72	46.75	12.50
6	4.82	8.51	26.57	29.44	40.00	7.25
8	5.68	9.03	27.24	29.87	31.25	3.25
10	6.53	9.31	28.30	30.54	26.50	2.25
12	7.42	9.44	29.19	30.82	17.75	1.25
14	8.17	9.54	29.42	30.54	15.25	0.25
16	8.86	9.59	30.63	31.20	8.00	0.00
18	9.50	9.61	30.60	30.63	3.00	0.00
22	9.91	9.61	31.18	30.85	0.25	0.00

(b) # Poisons = 10

TABLE IV: Evaluation of BP and CP when the victim employs the l_2-norm centroid defense.

μ	# Deleted Poisons		# Deleted Samples		Adv. Success Rate (%)	
	BP	CP	BP	CP	BP	CP
0.00	-	-	-	-	42.5	37.25
0.02	1.00	1.00	10.00	10.00	35.00	30.25
0.04	2.00	2.00	20.00	20.00	30.50	19.00
0.06	3.00	3.00	30.00	30.00	17.25	7.75
0.08	3.99	3.99	40.00	40.00	4.75	1.75
0.10	4.96	4.93	50.00	50.00	0.25	0.75
0.12	4.99	4.98	60.00	60.00	0.00	0.00
0.14	4.99	4.98	70.00	70.00	0.00	0.00
0.16	5.00	4.98	80.00	80.00	0.00	0.00
0.18	5.00	4.99	90.00	90.00	0.00	0.00
0.20	5.00	4.99	100.00	100.00	0.50	0.00

(a) # Poisons = 5

| μ	# Deleted Poisons		# Deleted Samples		Adv. Success Rate (%)	
	BP	CP	BP	CP	BP	CP
0.00	-	-	-	-	57.75	51.25
0.02	1.00	1.00	10.00	10.00	55.00	47.25
0.04	2.00	2.00	20.00	20.00	53.25	45.50
0.06	3.00	3.00	30.00	30.00	49.50	40.25
0.08	4.00	4.00	40.00	40.00	43.50	34.00
0.10	5.00	5.00	50.00	50.00	37.50	21.50
0.12	6.00	6.00	60.00	60.00	32.50	17.00
0.14	7.00	7.00	70.00	70.00	22.25	8.25
0.16	8.00	8.00	80.00	80.00	11.75	4.75
0.18	8.99	9.00	90.00	90.00	3.00	0.75
0.20	9.93	9.56	100.00	100.00	0.25	0.00

(b) # Poisons = 10
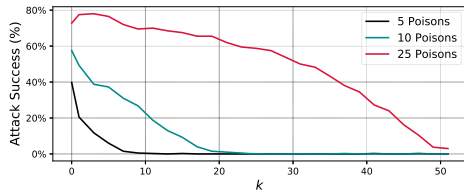
(a) BP vs. Deep k-NN
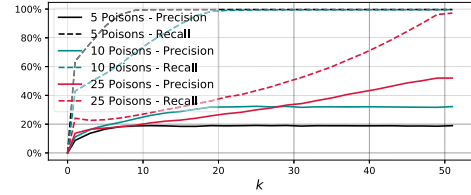
(b) Precision and Recall of the Deep k-NN defense.
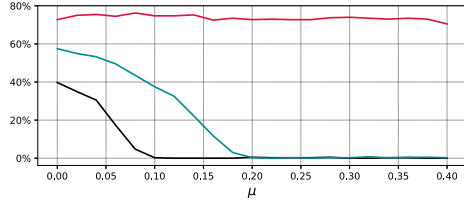
(c) BP vs. l_2-norm centroid
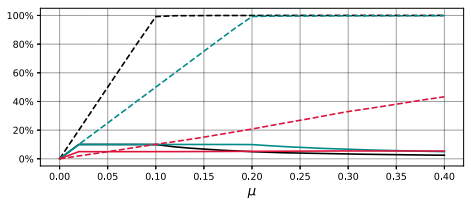
(d) Precision and Recall of the l_2-norm centroid defense.

Fig. 13: Evaluation of BP against Deep k-NN and l_2-norm centroid defenses, when 5, 10, or 25 poison samples are used.
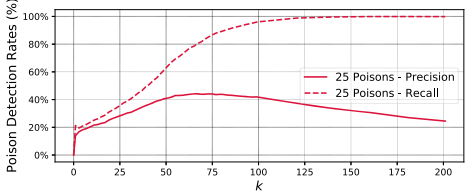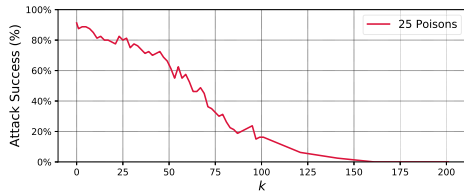

Fig. 14: Evaluation of BP against the Deep k-NN defense, when the target is classless.

to be quite effective, even when more poison samples are used. Increasing the number of poison samples from five to ten makes this defense remove five more samples on average. We should note that both attacks are completely mitigated after eliminating 6% of the victim's dataset, of which 4% are clean samples. The precision of poison detection is still low ($\sim 33\%$). To further see the effect of the number of poison samples on the precision and recall of poison detection, we evaluated BP, when crafting 25 poison samples. Figure 13 shows the performance of Deep k-NN and l_2-norm centroid defenses against BP when the number of poison samples increases from five to ten and then to 25. Figure 13c demonstrates that the l_2-norm centroid defense is not a plausible choice. When 25 poison samples are used, removing 40% of the dataset reduces the success rate of BP from 75% to 70%. This happens because as more clean samples are removed from the dataset, the poison samples will play a more important role in the training process.

As Figure 13b shows, the poison recall rate of the Deep k-NN defense reaches 100% as k becomes about two times the number of poison samples. This is not surprising, as it is almost impossible for the poison label to be identified as the plurality among samples in the neighborhood of the target in such a case. On the other hand, this defense is likely to fail if the number of poison samples is large enough to overwhelm the conformity test for each poison sample. This will happen with high probability when the number of poison samples is larger than the number of data points in the target's true class. In this case, the majority (or plurality) of points in the neighborhood of each poison sample will likely have the same label as the poison itself. In fact, we observed that when samples in the target's class are fewer than the number of poisons in the fine-tuning set, the poison samples pass the test undetected in most cases, hence, the attack remains active. Furthermore, if the target is classless, i.e., does not belong to any of the classes in the training set, the defense becomes less effective, as the poison samples surrounding the target are no longer part of a cluster related to the target's class. To evaluate this claim, we selected the first ten images of the 102 Category Flower dataset [25] as the targets, with "ship" being the misclassification class. As Figure 14 shows, setting k to 50 reduces the attack success rate to 66% for a classless target, whereas for a target from CIFAR-10 the attack is fully mitigated (Figure 13a). Complete mitigation of the attack requires $k > 150$, which results in discarding more than 70 samples from the fine-tuning set, of which 45 are clean.

E. Comparison On Standardized Benchmarks

A very recent paper [34] introduced standardized benchmarks for backdoor and poisoning attacks. In particular, the benchmarks include the following attacks.

- Clean-label poisoning attacks against transfer learning: FC [35], CP [50], and BP (our attack).
- Clean-label and hidden-trigger backdoor attacks: CLBD [44], and HTBD [32].

- A from-scratch attack: Witches' Brew (WiB). Unlike transfer learning, this attack assumes that the victim trains a new, randomly initialized model on the poisoned dataset.

Our attack was included in this benchmark evaluation, as we had made a pre-print version of our work available on arXiv. In the following, we summarize and expand on these third-party results.

Standardized Setup of Benchmarks. For the benchmarks, poisoning attacks are always restricted to generate poison samples that remain within the l_∞-ball of radius $\frac{8}{255}$ centered at the corresponding base images. On the other hand, backdoor attacks can use any 5×5 patch. Target and base images are chosen from the testing and training sets, respectively, according to a seeded, reproducible random assignment. This allows the benchmarks to use the same choices for each attack and remove a source of variation from the results. Each experiment uses 100 independent trials. In general, two different training modes are considered: (i) linear transfer learning, and (ii) from-scratch training, where the victim's network is trained from random initialization on the poisoned dataset.

Unlike our experiments in Section VI, the parameters of only one model are given to the attacker. In linear transfer learning, the attacks are evaluated in white-box and black-box scenarios. For white-box tests, the same frozen feature extractor that is given to the attacker is used for evaluation. In black-box settings, the attacks are evaluated against unseen feature extractor networks. Benchmarks can be divided into two sets of CIFAR-10 benchmarks and TinyImageNet benchmarks.

In CIFAR-10 benchmarks, for linear transfer learning, models are pre-trained on *CIFAR-100*, and the fine-tuning is done on a subset of CIFAR-10, which has the first 250 images from each class, allowing for 25 poison samples. The attacker has access to a ResNet-18 [12] network, and the victim uses either (1) the same ResNet-18 network (white-box scenario) or (2) VGG11 [37] and MobileNetV2 [33] networks (black-box scenario). We extend the benchmarks here by considering a gray-box scenario, where the attacks are evaluated against a ResNet-18 network with unseen parameters. Furthermore, for the black-box setting, we evaluate the attacks against ResNet-34 and ResNet-50 networks [12] as well. For these extra evaluations, we have used the poison samples that are shared by Schwarzschild et al. [34] in their GitHub repository,[7] and here we report the detailed numbers. When training from scratch, benchmarks use one of ResNet-18, VGG11, and MobileNetV2 networks, and report the average attack success rate. For this mode, benchmarks use 500 poison samples.

In TinyImageNet benchmarks, for linear transfer learning, models are pre-trained on the first 100 classes of the Tiny-ImageNet dataset [18] and fine-tuned on the second half of the dataset, allowing for 250 poison samples. The attacker has access to a VGG16 network, and black-box tests are done on ResNet-34 and MobileNetV2 networks. For the from-scratch setting, the benchmarks are evaluated against a VGG16 model that is trained on the entire dataset with 250 poison samples.

[7]Accessed Feb. 15 2021.

TABLE V: Success rates (%) of six attacks that are evaluated in the benchmark study [34]. The poison samples are not shared for some experiments, thus, we reported the exact numbers from the study. For example, in the black-box scenario of TinyImageNet benchmarks, the original paper reported the attack success rate, averaged over when ResNet-34 or MobileNetV2 networks are used. Therefore, we were not able to present individual attack success rates for these two settings.

| | Linear Transfer Learning | | | | | | | | Training From Scratch | |
| | CIFAR-10 | | | | | | TinyImageNet | | CIFAR-10 | TinyImageNet |
Attack	White-box ResNet18	Gray-box ResNet18	Black-box ResNet34	ResNet50	VGG11	MobileNetV2	White-box VGG16	Black-box $\frac{ResNet34+MobileNetV2}{2}$	$\frac{VGG16+ResNet34+MobileNetV2}{3}$	VGG16
FC	22	6	4	4	7	7	49	2	1.33	4
CP	33	7	5	4	8	7	14	1	0.67	0
BP	**85**	**10**	**8**	**6**	9	7	**100**	**10.5**	2.33	**44**
WiB	-	-	-	-	-	-	-	-	**26**	32
CLBD	5	5	4	4	7	6	3	1	1	0
HTBD	10	6	6	3	**14**	6	3	0.5	2.67	0

Results. Table V shows the success rates of the benchmarks. In linear transfer learning, our attack outperformed other attacks by a significant margin, especially in white-box settings. For example, in the TinyImageNet benchmark, BP achieved an attack success rate of 100%, while HTBD, CLBD, CP, and FC demonstrated success rates of 3%, 3%, 14%, and 49%, respectively. In the gray-box setting, BP showed only a modest improvement over other attacks. For the black-box settings in CIFAR-10 benchmarks, BP showed minimal improvement – on average 1-2% – in comparison to other attacks. In the black-box scenario of TinyImageNet benchmarks, BP achieved an attack success rate of 10.5%, while other attacks were below 2%. In general, BP has shown a superior performance with respect to other contenders in the linear transfer learning mode.[8] It is worth noting that backdoor attacks assume stronger threat models compared to poisoning attacks, as they need to manipulate both the training data and the target sample.

Before discussing the results in from-scratch training settings, we emphasize that BP is designed to attack transfer learning scenarios. Similar to FC and CP, BP does not consider from-scratch training scenarios. We expect the performance of BP to drop in such a scenario, as the feature space is constantly being altered during training. On the other hand, CLBD, and WiB attacks are specifically designed to target such scenarios. On CIFAR-10 benchmarks, all attacks succeeded less than 3% of the time. The only exception is WiB, which achieves a success rate of 26%. However, in TinyImageNet benchmarks, interestingly, BP demonstrated a success rate of 44%, surpassing the runner-up attack (WiB) by 12%. This shows that BP has the capability to produce poison samples that even survive from-scratch training scenarios for the higher dimensional TinyImageNet dataset.

VII. DISCUSSION

In Section VI-D, we have evaluated Bullseye Polytope against defenses presented in a (concurrent) paper [29]. We found that the Deep k-NN defense mitigates our attack completely if clean data points from the target's original class outnumber the poison samples. However, such a defense still suffers from a poor precision rate, i.e., it removes a considerable

---

[8]WiB is not evaluated in the transfer learning mode, as it is not considered in the original work [8].

number of clean samples, which, in turn, might have negative effects on the model performance. We believe future defenses need to be proposed with higher precision rates.

In our experiments, we have noticed that Bullseye Polytope adds noticeable amounts of noise to the poison samples. In fact, a recent study of clean-label poisoning attacks [34] acknowledges this limitation; poisoning attacks, which claim to be "clean label," often produce easily visible image artifacts and distortions. This study advocates using a perturbation budget ϵ of 0.03. In Section VI-C, we observed that by using $\epsilon = 0.03$, our attack produces much fewer distortions, while still achieving an attack success rate of 33.0% (see Figure 16 in the Appendix for the visual effect of ϵ on poison examples). In general, work in adversarial ML (in the image domain) suffers from the lack of a clear metric to determine what level of noise is imperceptible by the human eye. Clean-label poisoning definitely benefits from additional research on this issue to produce less perceptible perturbations.

VIII. CONCLUSIONS

In this work, we present a scalable and transferable clean-label poisoning attack, Bullseye Polytope, for transfer learning. Bullseye Polytope searches for poison samples that create, in the feature space, a convex polytope around the target image, ensuring that a linear classifier that trains on the poisoned dataset will classify the target into the poison class. By driving the polytope center close to the target, Bullseye Polytope outperforms Convex Polytope—a state-of-the-art attack against transfer learning— with success rate improvement of 7.44% and 26.75% for linear transfer learning and end-to-end transfer learning, respectively. At the same time, Bullseye Polytope achieves 10-36x faster poison sample generation, which is crucial for enabling future research toward the development of reliable defenses. Our evaluation of two neighborhood conformity defenses shows that Bullseye Polytope is more robust than Convex Polytope against less aggressive defense configurations. As the number of poison samples increases, the l_2-norm centroid defense becomes ineffective. The Deep k-NN defense also becomes vulnerable when poison samples outnumber the samples from the target's true class. In general, both defenses demonstrated low detection precision, which indicates further research needs to be done to improve the precision of such defenses.

REFERENCES

[1] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *International Conference on Machine Learning*. PMLR, 2018, pp. 274–283.

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

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

[4] C. Burkard and B. Lagesse, "Analysis of causative attacks against svms learning from data streams," in *Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics*. ACM, 2017, pp. 31–36.

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

[6] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2722–2730.

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

[8] J. Geiping, L. Fowl, W. R. Huang, W. Czaja, G. Taylor, M. Moeller, and T. Goldstein, "Witches' brew: Industrial scale data poisoning via gradient matching," *arXiv preprint arXiv:2009.02276*, 2020.

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

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

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

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

[13] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[14] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[15] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," in *Advances in neural information processing systems*, 2018, pp. 8571–8580.

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

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

[18] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," *CS 231N*, vol. 7, p. 7, 2015.

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

[20] S. Mei and X. Zhu, "Using machine teaching to identify optimal training-set attacks on machine learners," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

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

[22] K. Muandet, K. Fukumizu, B. Sriperumbudur, and B. Schölkopf, "Kernel mean embedding of distributions: A review and beyond," *Foundations and Trends in Machine Learning*, vol. 10, no. 1-2, pp. 1–144, 2017.

[23] R. M. Neal, *Bayesian learning for neural networks*. Springer Science & Business Media, 1996, vol. 118.

[24] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. Rubinstein, U. Saini, C. A. Sutton, J. D. Tygar, and K. Xia, "Exploiting machine learning to subvert your spam filter." *LEET*, vol. 8, pp. 1–9, 2008.

[25] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*. IEEE, 2008, pp. 722–729.

[26] M. Ozuysal, V. Lepetit, and P. Fua, "Pose estimation for category specific multiview object localization," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 778–785.

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

[28] O. M. Parkhi, A. Vedaldi, A. Zisserman *et al.*, "Deep face recognition." in *bmvc*, vol. 1, no. 3, 2015, p. 6.

[29] N. Peri, N. Gupta, W. R. Huang, L. Fowl, C. Zhu, S. Feizi, T. Goldstein, and J. P. Dickerson, "Deep k-nn defense against clean-label data poisoning attacks," in *European Conference on Computer Vision*. Springer, 2020, pp. 55–70.

[30] C. Qin, J. Martens, S. Gowal, D. Krishnan, K. Dvijotham, A. Fawzi, S. De, R. Stanforth, and P. Kohli, "Adversarial robustness through local linearization," in *Advances in Neural Information Processing Systems*, 2019, pp. 13 847–13 856.

[31] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in neural information processing systems*, 2008, pp. 1177–1184.

[32] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trigger backdoor attacks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 11 957–11 965.

[33] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[34] A. Schwarzschild, M. Goldblum, A. Gupta, J. P. Dickerson, and T. Goldstein, "Just how toxic is data poisoning? a unified benchmark for backdoor and data poisoning attacks," *arXiv preprint arXiv:2006.12557*, 2020.

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

[36] A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!" in *Advances in Neural Information Processing Systems*, 2019, pp. 3358–3369.

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[38] A. Smola, A. Gretton, L. Song, and B. Schölkopf, "A hilbert space embedding for distributions," in *International Conference on Algorithmic Learning Theory*. Springer, 2007, pp. 13–31.

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

[40] O. Suciu, R. Marginean, Y. Kaya, H. Daume III, and T. Dumitras, "When does machine learning {FAIL}? generalized transferability for

evasion and poisoning attacks," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 1299–1316.

[41] Y. Sun, X. Wang, and X. Tang, "Deep learning face representation from predicting 10,000 classes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1891–1898.

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

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

[44] A. Turner, D. Tsipras, and A. Madry, "Clean-label backdoor attacks," 2018.

[45] R. Wang, C. Han, Y. Wu, and T. Guo, "Fingerprint classification based on depth neural network," *arXiv preprint arXiv:1409.5188*, 2014.

[46] H. Xiao, H. Xiao, and C. Eckert, "Adversarial label flips attack on support vector machines." in *ECAI*, 2012, pp. 870–875.

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

[48] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[49] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Advances in neural information processing systems*, 2017, pp. 3391–3401.

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

APPENDIX A
POISON VISUALIZATION

Figure 18 depicts poison samples generated by Convex Polytope and Bullseye Polytope for one particular target. The first row shows the original images that are selected for crafting the poison samples. Figure!19 depicts poison samples generated by Convex Polytope and Bullseye Polytope in multi-target mode, when multiple images of the target object (from different angles) are considered for crafting poison samples. Note that we use the Multi-View Car Dataset [26] to select the target images.

APPENDIX B
COEFFICIENTS OPTIMIZATION STEP IN CONVEX POLYTOPE

As we discussed in Section IV, Convex Polytope performs three steps in each iteration of the attack. We observed that step one takes a significant amount of time compared to the other two steps. Algorithm 1 shows the details of step one, which searches for the (most) suitable coefficients for the current poison samples at the time.

APPENDIX C
BULLSEYE POLYTOPE VS. ENSEMBLE FEATURE COLLISION

To evaluate Convex Polytope, Zhu et al. [50] developed an ensemble version of Feature Collision [35] to craft *multiple* poison samples instead of one. They further used this ensemble version as a benchmark. The corresponding loss function is defined as:

$$L_{FC} = \sum_{i=1}^{m} \sum_{j=1}^{k} \frac{\left\| \phi^{(i)}(x_p^{(j)}) - \phi^{(i)}(x_t) \right\|^2}{\left\| \phi^{(i)}(x_t) \right\|^2}. \quad (3)$$

They argue that unlike Feature Collision, Convex Polytope's loss function (Eq. 1) allows the poison samples to lie further away from the target. Experiments showed that Convex Polytope outperforms Feature Collision, especially in black-box settings. It should be noted that, contrary to what is stated by Zhu et al. [50], the Ensemble

---

Algorithm 1 Convex Polytope - Coefficients Updating

1: **Input:** A $\leftarrow$ $\{\phi(x_p^{(j)})\}_{j=1}^{k}$
2: $\alpha \leftarrow \frac{1}{\|A^T A\|}$
3: **for** $i = 1$ **to** m **do**
4:    **while** *not converged* **do**
5:       $\widehat{c}^{(i)} \leftarrow c^{(i)} - \alpha A^T (A c^{(i)} - \phi^{(i)}(x_t))$
6:       **if** $\text{loss}(\widehat{c}^{(i)}) \geq \text{loss}(c^{(i)})$ **then**
7:          $\alpha \leftarrow \frac{1}{\alpha}$
8:       **else**
9:          $c^{(i)} \leftarrow \widehat{c}^{(i)}$
10:          project $c^{(i)}$ onto the probability simplex.
11:       **end if**
12:    **end while**
13: **end for**

Feature Collision attack objective described by Eq. 3 is not a special case of Eq. 1 (when the coefficients are set to $\frac{1}{k}$), rather, it optimizes completely decoupled objectives for different poison samples. While centering the target between poison samples allows for more flexibility in poison locations, Eq. 3 pushes all poison samples close to the target, which has the same drawbacks of collision attacks, namely, perceptible patterns showing up in poison images and limited transferability. By exploiting this approach of centering, we show that Bullseye Polytope improves both attack transferability and scalability.

APPENDIX D
DETAILED RESULTS FOR SINGLE-TARGET MODE

A. End-to-End Transfer Learning

Figure 20 shows the attack success rates of CP, BP, BP-3x and BP-5x, against each individual victim model when the victim employs end-to-end transfer learning. Among them, the last row presents the black-box setting. We note that none of CP, BP, and BP-3x shows attack transferability for GoogLeNet. Zhu et al. [50] have made a similar observation. They argued that since GoogLeNet has a more different architecture than the substitute models, it is more difficult for the "attack zone" to survive end-to-end transfer learning.

B. Different Pairs of <original class, poison class>

To assess the effect of original and poison classes on the attack performance, we evaluate BP-3x for all 90 pairs of <original class, poison class>; each with 5 different target images (indexed from 4,851 to 4,855 in the original class), resulting in a total of 450 attack instances. We focus on linear transfer learning and limit each attack to 800 iterations to meet time and resource constraints. Figure 15 shows the attack performance against individual victim networks in this setting as well as the original setting of <frog, ship>. Table VI shows the average attack performance for individual class pairs. In particular, we have found the attack much less successful when our targeted misclassification is one of `airplane` or `deer` classes.

APPENDIX E
IMPLEMENTATION DETAILS

The authors of Convex Polytope released the source code of CP along with the substitute networks. All models are trained with the same architecture and hyperparameters defined in `https://github.com/kuangliu/`, except for dropout. We used their implementation directly for comparison. For all experiments, we used `PyTorch-v1.3.1` over `Cuda 10.1`. We ran all the attacks using `NVIDIA Titan RTX` graphics cards. For solving Eq. 1 (Convex Polytope) and Eq. 3 (Bullseye Polytope), we used similar settings and parameters to what is practiced by Zhu et al. [50].
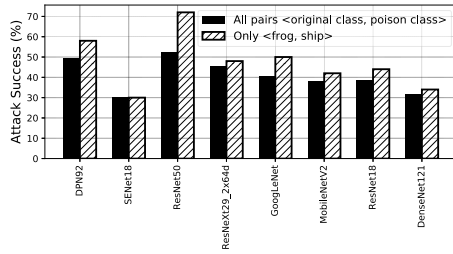
Fig. 15: Attack success rates of BP-3x for all 90 pairs of <original class, poison class> in linear transfer learning as well as the original setting <frog, ship> (for 50 target images indexed from 4,851 to 4,900).
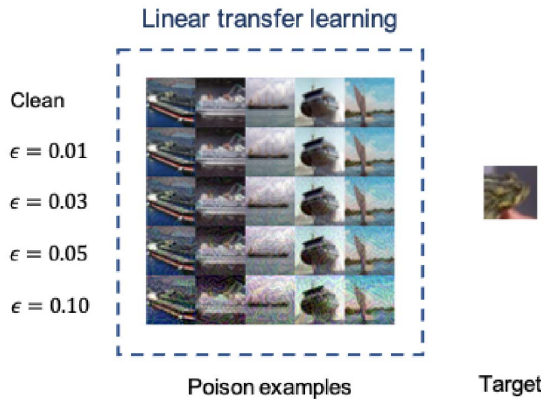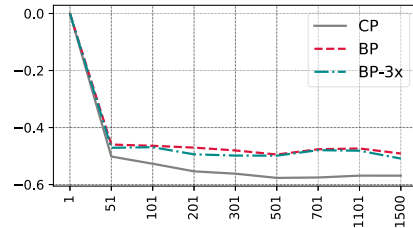

Linear transfer learning

Fig. 16: Poison samples crafted by Bullseye Polytope attacks in linear transfer learning using different values of ϵ.

Processing the Multi-View Car Dataset. The resolutions of the Multi-View Car dataset are 376×250. To resize the images of this dataset to 32×32 (the resolution of the CIFAR-10 images), we have used the `opencv-python` library. While resizing the images, we achieved the best performance of the models on the Multi-View Car dataset using the `cv2.INTER_AREA` interpolation. It should be noted that the Multi-View Car dataset provides the exact location of the cars in the images.

APPENDIX F
DEFENSES AGAINST EVASION AND BACKDOOR ATTACKS

Most adversarial defenses are proposed for mitigating evasion attacks, where a targeted input is perturbed by imperceptible amounts during inference to enable misclassification. Such perturbations are calculated using the gradients of the loss function on the victim network, or a set of surrogate networks if the victim network is unknown [10], [2], [43]. Many defenses against evasion attacks focus on obfuscating the gradients [1]. They achieve this in several ways, e.g., introducing randomness during test time, or using non-differentiable layers. Athalye et al. [1] demonstrate that such defenses can be easily defeated by introducing techniques to circumvent the absence of gradient information, like replacing non-differentiable layers with approximation differentiable layers. Robust defenses to evasion attacks must avoid relying on obfuscated gradients and provide a "smooth" loss surface in the data manifold. Variants of adversarial training [19],

(a) Zero overlap
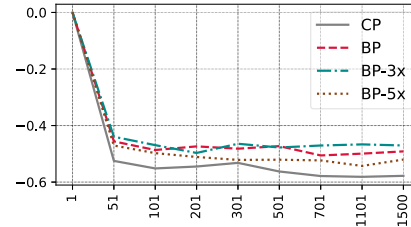
(b) 50% overlap

Fig. 17: Average variation in baseline test accuracy of models in linear transfer learning, when there is zero or 50% overlap between training sets of the victim and substitute networks.

[36], [47] and linearity or curvature regularizers [21], [30] are proposed to achieve this property. These defenses provide modest accuracy against strong multi-iteration PGD attacks [19]. Papernot et al. [27] proposed the Deep k-NN classifier, which combines the k-nearest neighbors algorithm with representations of the data learned by each layer of the neural network, as a way to detect outlier examples in feature space, with the hope that adversarial examples are the outliers.

Several defenses are proposed against backdoor attacks, primarily focusing on neighborhood conformity tests to sanitize the training data. Steinhardt et al. [39] exploited variants of l_2-norm centroid defense, where a data point is anomalous if it falls outside of a parameterized radius in feature space. Chen et al. [5] employed feature clustering to detect and remove the poison samples, with the assumption that backdoor triggers will cause poison samples to cluster in feature space.

APPENDIX G
DEEP SETS

One of the contributions of the "Deep Sets" paper is a characterization of all functions that take a set as input, which says that any such function f can be written as another function ρ of certain mean embedding ϕ of the elements of the sets. We are instantiating this theorem in the following way: (1) The set input is the set of poison samples $\{x_p^{(j)}\}_{j=1}^k$. (2) f is a prediction function:

$$f(\{x_p^{(j)}\}_{j=1}^k) = \text{Predict}(\text{Train}(X_C + \{x_p^{(j)}\}_{j=1}^k), x_t)$$

where $\text{Predict}(h, x)$ applies a classifier h to data point x. X_C denotes the clean data. This is a set-function due to the permutation-invariant training procedure (e.g., Shuffle + SGD) that is typically adopted. By the theorem, this function has an alternative representation $\rho(\frac{1}{k}\sum_{j=1}^k \phi(x_p^{(j)}))$ that depends only on a certain mean embedding ϕ of the poison samples. For this reason, it motivates us to set the $\{c^{(i)}\}$ in Eq. 1 to $\frac{1}{k}$, which results in Eq. 2. We acknowledge that this is not a formal theorem statement because the feature map ϕ that we used might not be the same as the feature map that is required in applying Deep Sets theory, but given the flexibility of neural networks, we believe if we end-to-end optimize over ϕ too, it is a reasonable approximation.
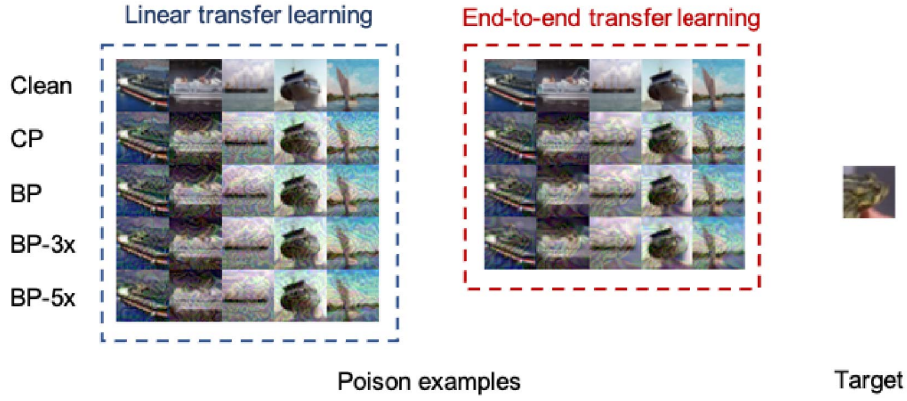
Fig. 18: Poison samples crafted by Convex Polytope and Bullseye Polytope attacks. The first row shows the original images selected for crafting the poison samples.
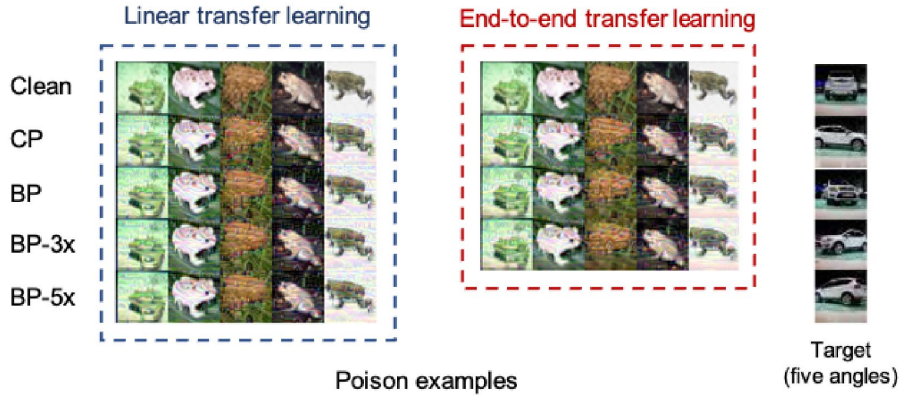

Fig. 19: Poison samples crafted by Convex Polytope and Bullseye Polytope attacks in multi-target mode. The first row shows the original images selected for crafting the poison samples.

TABLE VI: Evaluation of BP-3x against linear transfer learning for individual class pairs. Attacks are limited to 800 iterations. Each individual pair is tested using five different target images. Having considered eight victim networks, in total, we evaluate each pair against the victim's network 40 times. Each cell shows the number of times that the attack succeeded for each pair.

| | | Poison Class | | | | | | | | | | |
		airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck	Total (of 360)
Original Class	airplane	-	14	13	17	9	18	17	20	16	24	148
	automobile	7	-	17	19	12	15	20	16	25	24	155
	bird	6	13	-	24	7	14	20	22	20	22	148
	cat	6	9	11	-	10	18	15	13	14	22	118
	deer	6	17	15	24	-	15	17	24	17	23	158
	dog	8	15	13	31	7	-	17	16	15	22	144
	frog	5	17	16	20	10	15	-	15	35	23	156
	horse	10	11	14	22	12	17	23	-	19	26	154
	ship	5	20	17	24	9	19	0	18	-	23	135
	truck	6	16	13	23	12	18	20	23	23	-	154
	Total (of 360)	**59**	142	129	204	**88**	149	149	167	184	209	
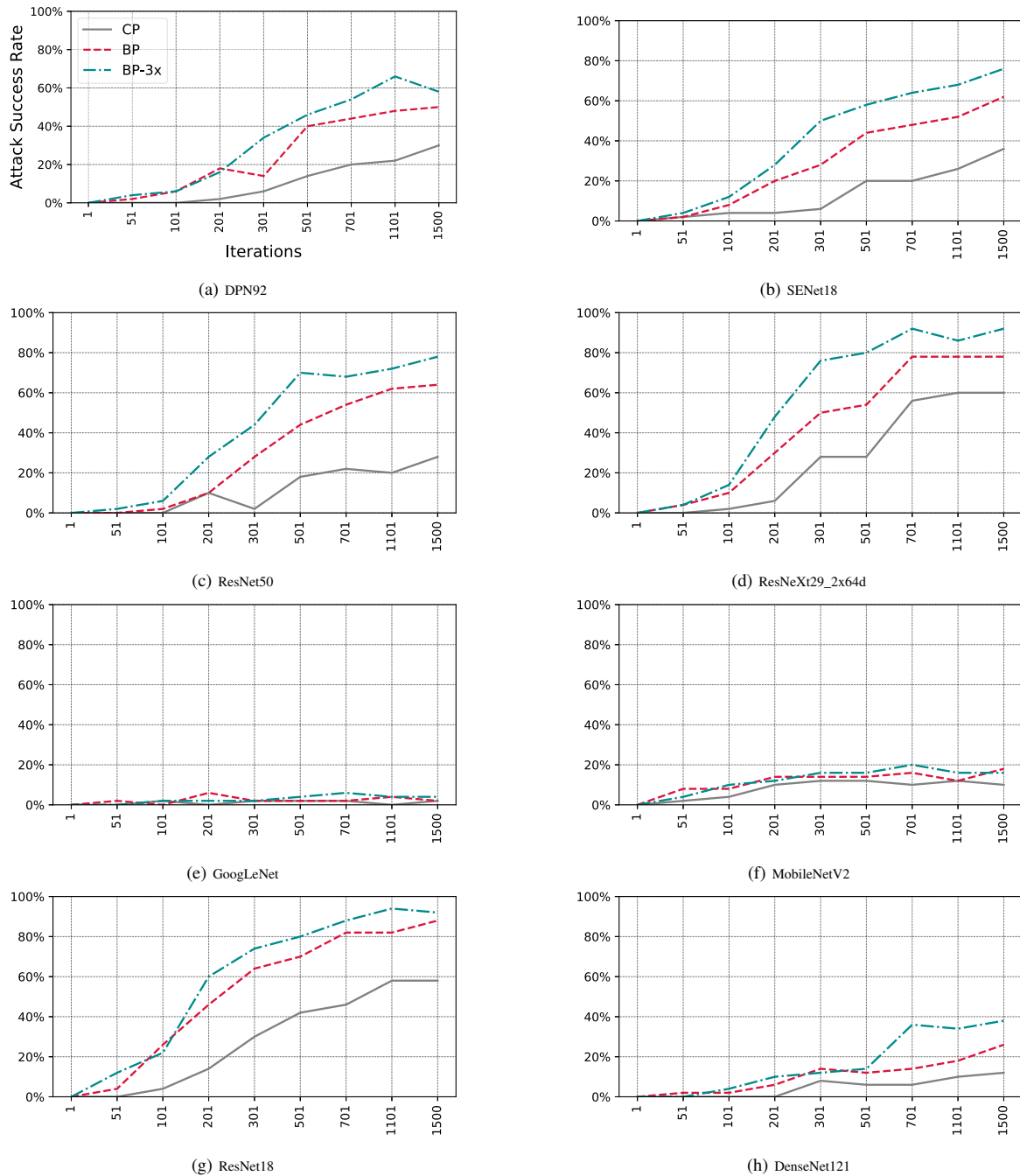
Fig. 20: End-to-end transfer learning: Success rates of CP, BP, BP-3x, and BP-5x, against each individual victim model. Notice `GoogLeNet`, `MobileNetV2`, `ResNet18` and `DenseNet121` are the black-box setting.
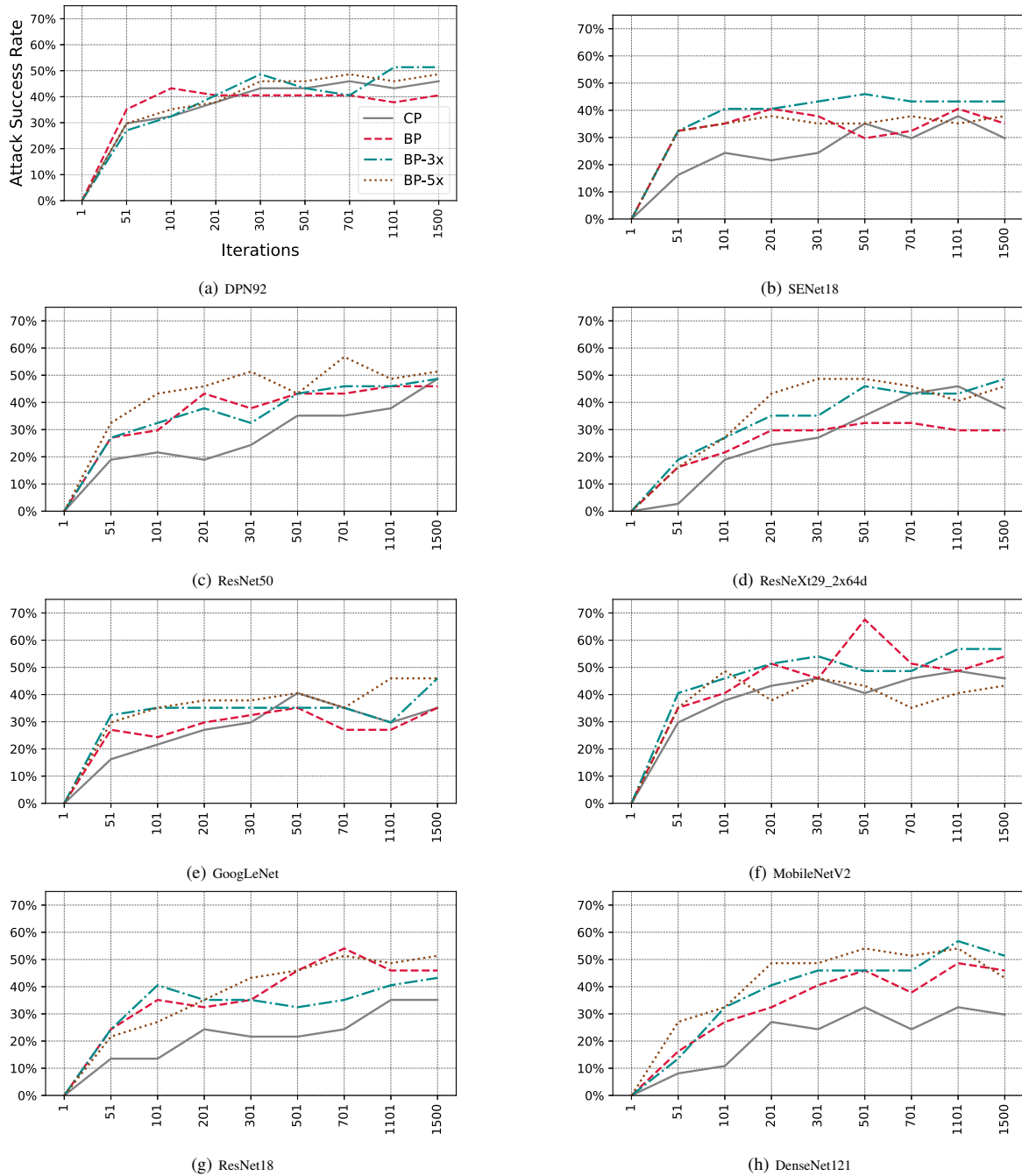
Fig. 21: Linear transfer learning when we have 50% overlap between the training sets of substitute and victim's networks: Success rates of CP, BP, and BP-3x, against each individual victim model.