Adversarial Examples Make Strong Poisons

Anonymous Author(s)
Affiliation
Address
email

Abstract

The adversarial machine learning literature is largely partitioned into evasion attacks on testing data and poisoning attacks on training data. In this work, we show that adversarial examples, originally intended for attacking pre-trained models, are even more effective for data poisoning than recent methods designed specifically for poisoning. To neural networks, adversarial examples look fundamentally different from clean images. In fact, adversarial examples with labels re-assigned by the crafting network remain effective for training, suggesting that adversarial examples contain useful semantic content, just with the "wrong" labels (according to a network, but not a human). Our method, *adversarial poisoning*, is substantially more effective than existing poisoning methods for secure dataset release, and we release a poisoned version of ImageNet, ImageNet-P, to encourage research into the strength of this form of data obfuscation.

1 Introduction

The threat of malicious dataset manipulations is ever proliferating as the training data curation process for machine learning systems is increasingly void of human supervision. Automated scraping has become necessary to satisfy the exploding demands of cutting-edge deep models [Bonawitz et al., 2019, Brown et al., 2020], but the same automation that enables massive performance boosts exposes these models to security vulnerabilities [Bagdasaryan et al., 2020, Chen et al., 2020]. *Data poisoning* attacks manipulate training data in order to cause the resulting models to misclassify samples during inference [Koh and Liang, 2017], while *backdoor attacks* embed exploits which can be triggered by pre-specified input features [Chen et al., 2017]. In this work, we focus on a flavor of data poisoning known as *availability attacks*, which aim to degrade overall testing performance [Barreno et al., 2010, Biggio et al., 2012].

Adversarial attacks, on the other hand, focus on manipulating samples at test-time, rather than during training [Szegedy et al., 2013]. In this work, we connect adversarial and poisoning attacks by showing that adversarial examples form stronger availability attacks than any existing poisoning method, even though the latter were designed specifically for manipulating training data while adversarial examples were not. We compare our method, *adversarial poisoning*, to existing availability attacks for neural networks, and we exhibit consistent performance boosts (i.e. lower test accuracy). In fact, models trained on adversarial examples may exhibit test-time performance below that of random guessing.

Intuitively, adversarial examples look dramatically different from the originals in the eye of neural networks despite the two looking similar to humans. Thus, models trained only on such perturbed examples are completely unprepared for inference on the clean data. In support of this intuition, we observe that models trained on adversarially perturbed training data cannot even classify the original clean training samples.

But does this phenomenon occur simply because adversarial examples are off the "natural image manifold" or because they actually contain informative features from other classes? Popular belief

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

assumes that adversarial examples live off the natural image manifold causing a catastrophic mismatch when digested by models trained on only clean data [Khoury and Hadfield-Menell, 2018, Stutz et al., 2019, Zhao et al., 2017]. However, models trained on data with random additive noise (rather than adversarial noise) perform well on noiseless data, suggesting that there may be more to the effects of adversarial examples than simply moving off the manifold (see Table 2). We instead find that since adversarial attacks inject features that a model associates with incorrect labels, training on these examples is similar to training on *mislabeled* training data. After re-labeling adversarial examples with the "wrong" prediction of the network with which they were crafted, models trained on such label-corrected data perform substantially better than models trained on uncorrected adversarial examples and almost as well as models trained on clean images. While this label-correction is infeasible for a practitioner defending against adversarial poisoning, since it assumes possession of the crafting network which requires access to the clean dataset, this experiment strengthens the intuition that adversarial examples contain a strong signal just from the "wrong" class - cf. Ilyas et al. [2019].

2 Related Work

Data poisoning. Classical approaches to availability attacks involve solving a bilevel optimization problem which minimizes loss with respect to parameters in the inner problem while maximizing loss with respect to inputs in the outer problem [Biggio et al., 2012, Huang et al., 2020]. On simple models, the inner problem can be solved exactly, and many works leverage this [Biggio et al., 2012, Mei and Zhu, 2015, Xiao et al., 2015], but on neural networks, obtaining exact solutions is intractable. To remedy this problem, [Muñoz-González et al., 2017] approximates a solution to the inner problem using a small number of descent steps, but the authors note that this method is ineffective against deep neural networks. Other works adopt similar approximations but for integrity attacks on deep networks, where the attacker only tries to degrade performance on one particular test sample [Geiping et al., 2020, Huang et al., 2020]. The bilevel approximation of [Geiping et al., 2020] can also adapted to availability attacks for neural networks by substituting the original targeted objective with an indiscriminate one [Fowl et al., 2021]. Another related approach harnesses *influence functions* which estimate the impact of each training sample on a resulting model [Fang et al., 2020, Koh and Liang, 2017, Koh et al., 2018]. However, influence functions are brittle on deep networks whose loss surfaces are highly irregular [Basu et al., 2020]. In order to conduct availability attacks on neural networks, recent works have instead modified data to cause gradient vanishing either explicitly or by minimizing loss with respect to the image, thus removing the influence of such data on training [Huang et al., 2021, Shen et al., 2019]. We compare to these methods and find that the proposed adversarial poisoning is significantly more effective.

Adversarial examples. Adversarial attacks probe the blindspots of trained models where they catastrophically misclassify inputs that have undergone small perturbations [Szegedy et al., 2013]. Prototypical algorithms for adversarial attacks simply maximize loss with respect to the input while constraining perturbations. The resulting adversarial examples exploit the fact that as inputs are even slightly perturbed in just the right direction, their corresponding deep features and logits change dramatically, and gradient-based optimizers can efficiently find these directions. The literature contains a wide array of proposed loss functions and optimizers for improving the effectiveness of attacks [Carlini and Wagner, 2017, Gowal et al., 2019]. A number of works suggest that adversarial examples are off the image manifold, and others propose methods for producing on-manifold attacks [Khoury and Hadfield-Menell, 2018, Stutz et al., 2019, Zhao et al., 2017].

Adversarial training. The most popular method for producing neural networks which are robust to attacks involves crafting adversarial versions of each mini-batch and training on these versions [Madry et al., 2017]. On the surface, it might sound as if adversarial training is very similar to training on poisons crafted via adversarial attacks. After all, they both involve training on adversarial examples. However, adversarial training ensures that the robust model classifies inputs correctly within a ball surrounding each training sample. This is accomplished by updating perturbations to inputs *throughout* training. This process desensitizes the adversarially trained model to small perturbations to its inputs. In contrast, a model trained on adversarially poisoned data is only encouraged to fit the exact, fixed perturbed data.

3 Adversarial Examples as Poisons

3.1 Threat Model and Motivation

We introduce two parties: the *poisoner* (sometimes called the attacker), and the *victim*. The poisoner has the ability to perturb the victim's training data but does not know the victim's model initialization, training routine, or architecture. The victim then trains a new model from scratch on the poisoned data. The poisoner's success is determined by the accuracy of the victim model on clean data.

Early availability attacks that worked best in simple settings, like SVMs, often modified only a small portion of the training data. However, recent availability attacks that work in more complex settings have instead focused on applications such as secure data release where the poisoner has access to, and modifies, *all* data used by the victim [Shen et al., 2019, Feng et al., 2019, Huang et al., 2021, Fowl et al., 2021].

These methods manipulate the entire training set to cause poor generalization in deep learning models trained on the poisoned data. This setting is relevant to practitioners such as social media companies who wish to maintain the competitive advantage afforded to them by access to large amounts of user data, while also protecting user privacy by making scraped data useless for training models. Practically speaking, companies could employ methods in this domain to imperceptibly modify user data *before* dissemination through social media sites in order to degrade performance of any model which is trained on this disseminated data.

To compare our method to recent works, we focus our experiments in this setting where the poisoner can perturb the entire training set. However, we also poison lower proportions of the data in Tables 5, 4. We find that on both simple and complex datasets, our method produces poisons which are useless for training, and models trained on data including poisons would have performed just as well had they identified and thrown out the poisoned data altogether.

3.2 Problem Setup

Formally stated, availability poisoning attacks aim to solve the following bi-level objective in terms of perturbations $\delta = \{\delta_i\}$ to elements x_i of a dataset $\mathcal{T}$:

$$\max_{\delta \in \mathcal{S}} \; \mathbb{E}_{(x,y)\sim\mathcal{D}}\Big[\mathcal{L}\left(F(x;\theta(\delta)), y\right)\Big] \tag{1}$$

$$\text{s.t. } \theta(\delta) \in \arg\min_{\theta} \sum_{(x_i,y_i)\in\mathcal{T}} \mathcal{L}(F(x_i+\delta_i;\theta), y_i), \tag{2}$$

where $\mathcal{S}$ denotes the constraint set of the perturbations. As is common in both the adversarial attack and poisoning literature, we employ an ℓ_∞ bound on each δ_i. Unless otherwise stated, our attacks are bounded by ℓ_∞-norm $\epsilon = 8/255$ as is standard practice on CIFAR-10, and ImageNet data in both adversarial and poisoning literature [Madry et al., 2017, Geiping et al., 2020]. Simply put, the attacker wishes to cause a network, F, trained on the poisons to generalize poorly to distribution $\mathcal{D}$ from which $\mathcal{T}$ was sampled.

Directly solving this optimization problem is intractible for neural networks as it requires unrolling the entire training procedure found in the inner objective (Equation (2)) and backpropagating through it to perform a single step of gradient descent on the outer objective. Thus, the attacker must approximate the bilevel objective. Approximations to this objective often involve heuristics. For example, TensorClog [Shen et al., 2019] aims to cause gradient vanishing in order to disrupt training, while more recent work aims to align poison gradients with an adversarial objective [Geiping et al., 2020]. We opt for an entirely different strategy and instead replace the bi-level problem with an empirical loss maximization problem, thus turning the poison generation problem into an adversarial example problem. Specifically, we optimize the following objective:

$$\max_{\delta \in \mathcal{S}} \Bigg[\sum_{(x_i,y_i)\in\mathcal{T}} \mathcal{L}\left(F(x_i+\delta_i;\theta^*), y_i\right)\Bigg], \tag{3}$$

where θ^* denotes the parameters of a model trained on *clean* data, which is fixed during poison generation. We call this model the *crafting* model. Fittingly, we call our method *adversarial poisoning*.

We also optimize a variant of this objective which defines a *class targeted* adversarial attack. This modified objective is defined by:

$$\min_{\delta \in \mathcal{S}} \left[\sum_{(x_i, y_i) \in \mathcal{T}} \mathcal{L}\left(F(x_i + \delta_i; \theta^*), g(y_i)\right) \right], \tag{4}$$

where g is a permutation (with no fixed points) on the label space of $\mathcal{S}$.

Projected Gradient Descent (PGD) has become the standard method for generating adversarial examples for deep networks [Madry et al., 2017]. Accordingly, we craft our poisons with 250 steps of PGD on this loss-maximization objective. In addition to the adversarial attack introduced in Madry et al. [2017], we also experiment with other attacks such as FGSM [Goodfellow et al., 2014] and Carlini-Wagner [Carlini and Wagner, 2017] in Appendix Table 2. We find that while other adversaries do produce effective poisons, a PGD based attack is the most effective in generating poisons. Finally, borrowing from recent targeted data poisoning works, we also employ differentiable data augmentation in the crafting stage [Geiping et al., 2020].

An aspect of note for our method is the ease of crafting perturbations - we use a straightforward adversarial attack on a fixed pretrained network to generate the poisons. This is in contrast to previous works which require pretraining an adversarial auto-encoder [Feng et al., 2019] (5 - 7 GPU days for simple datasets), or require iteratively updating the model and perturbations [Huang et al., 2021], which requires access to the entire training set all at once - an assumption that does not hold for practitioners like social media companies who acquire data sequentially. In addition to the performance boosts our method offers, it is also the most flexible compared to each of the availability attacks with which we compare.

3.3 Baseline CIFAR-10 Results

We first experiment in a relatively simple setting, CIFAR-10 [Krizhevsky et al., 2009], consisting of $50,000$ low-resolution images from 10 classes. All poisons in this setting are tested on a variety of architectures in a setting adapted from a commonly used repository [1]. We find that our poisoning method reliably degrades the test accuracy of a wide variety of popular models including VGG19 [Simonyan and Zisserman, 2014], ResNet-18 [He et al., 2015], GoogLeNet [Szegedy et al., 2015], DenseNet-121 [Huang et al., 2016], and MobileNetV2 [Sandler et al., 2018]. Even under very tight ℓ_∞ constraints, our poisons more than halve the test accuracy of these models. These results are presented in Table 1.

Table 1: **Comparison of different ε-bounds for our adversarial poisoning method.** All poisons generated by ResNet-18 crafted with 250 steps of PGD, with differentiable data augmentation. CT denotes poisons crafted with the class targeted adversarial attack. Note that we would expect random guessing to achieve 10% validation accuracy.

BOUND \ VICTIM	VGG19	RESNET-18	GOOGLENET	DENSENET-121	MOBILENETV2
CLEAN	92.24 ± 0.09	94.56 ± 0.06	94.68 ± 0.02	94.90 ± 0.03	92.31 ± 0.06
$\varepsilon = 4/255$	52.97 ± 1.23	40.14 ± 0.50	41.06 ± 0.22	42.91 ± 0.51	33.99 ± 0.45
$\varepsilon = 8/255$	10.98 ± 0.27	6.25 ± 0.17	7.03 ± 0.12	7.16 ± 0.16	6.11 ± 0.17
$\varepsilon = 4/255$ (CT)	28.49 ± 0.65	23.79 ± 0.27	23.05 ± 0.16	20.90 ± 0.22	20.53 ± 0.02
$\varepsilon = 8/255$ (CT)	7.03 ± 0.04	7.25 ± 0.44	7.08 ± 0.10	6.22 ± 0.44	7.15 ± 0.03

Additionally, we study the effect of the optimizer, data augmentation, number of steps, and crafting network on poison generation in Appendix Tables 10, 11, and 12. These tables demonstrate that a wide variety of adversarial attacks with various crafting networks and hyperparameters yield effective poisons. Also note that while the results we present here are for poisons generated with a ResNet18, we find that the poisons remain effective when generated from other network architectures (see Appendix Table 12).

In this popular setting, we can compare our method to existing availability attacks including Tensor-Clog [Shen et al., 2019], Loss Minimization [Huang et al., 2021], and a gradient alignment based method [Fowl et al., 2021]. Our method widely outperforms these previous methods. Compared with

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

the previous best method (loss minimization), we degrade the validation accuracy of a victim network by a factor of more than three.

Table 2: **Validation accuracies of models trained on data from different availability attacks.** Tested on randomly initialized ResNet-18 models on CIFAR-10. All crafted with $\varepsilon = 8/255$.

METHOD	VALIDATION ACCURACY (%, ↓)
NONE (CLEAN)	94.56
RANDOM NOISE	90.52
TENSORCLOG [SHEN ET AL., 2019]	84.24
ALIGNMENT [FOWL ET AL., 2021]	53.67
UNLEARNABLE EXAMPLES [HUANG ET AL., 2021]	19.85
DEEPCONFUSE [FENG ET AL., 2019]	31.10
ADVERSARIAL POISONING (OURS)	**6.25**

Note that we test our method in a completely black-box setting wherein the poisoner has no knowledge of the victim network's initialization, architecture, learning rate scheduler, optimizer, etc. We find our adversarial poisons transfer across these settings and reliably degrade the validation accuracy of all the models tested. See Appendix A.1 for more details about training procedures.

3.4 Large Scale Poisoning

In addition to validating our method on CIFAR-10, we also conduct experiments on ImageNet (ILSVRC2012), consisting of over 1 million images coming from 1000 different classes [Russakovsky et al., 2015]. This setting tests whether adversarial poisons can degrade accuracy on industrial-scale, high resolution datasets. We discover that while untargeted attacks successfully reduce the generalization, our class-targeted attack (CT) degrades clean validation accuracy significantly further (see Table 3). Furthermore, even at an almost imperceptible perturbation level of $4/255$, the class targeted poisons cripple the validation accuracy of the victim model to 3.57%. Visualizations of the poisons can be found in Figure 1.

Table 3: **Validation accuracies of models trained on poisoned ImageNet data**. Tested on randomly initialized ResNet-18 models.

METHOD	VALIDATION ACCURACY (%, ↓)
NONE (CLEAN)	66.56
$\varepsilon = 4/255$	45.07
$\varepsilon = 8/255$	36.63
$\varepsilon = 4/255$ (CT)	3.57
$\varepsilon = 8/255$ (CT)	**1.45**

Figure 1: Randomly selected clean image (left), with perturbed counterparts at level $\varepsilon = 4/255$ (center), and $\varepsilon = 8/255$ (right). The clean image is taken from class "hen" and poisons are generated via perturbations into class "ostrich".

We hypothesize that the noteable superiority of the class targeted attack in this setting arises from the larger label-space of ImageNet. Specifically, we find that untargeted adversarial attacks can result in

a concentrated attack matrix - i.e. attacks perturb data into a relatively small number of classes. This could lead to the perturbations being less effective at degrading generalization because they are not discriminatively useful for a network and are thus ignored. In contrast, a class-targeted attack ensures that the adversarially perturbed features can be associated with an incorrect class label when learned by a victim network.

3.5 Facial Recognition

In our third experimental setting, we test whether our poisons are effective against facial recognition models, along the lines of the motivations introduced in Section 3.1, where companies like social media sites could employ our method to protect user data from being scraped and used by facial recognition networks as demonstrated in [Cherepanova et al., 2021]. For the purposes of comparison, we constrain our attack to the same setting as Huang et al. [2021]. In this setting, we only assume ability to modify a subset of the face images, specifically, the faces images of the users that we want to protect. We test on the Webface dataset, and use pretrained Webface model from Yi et al. [2014] to generate the class targeted adversarial examples with $\varepsilon = 8/255$. Our class targeted adversarial examples are exceedingly effective in this domain. Our method reduces the protected classes' accuracy down to 8%, half the classification success of the unlearnable examples method of Huang et al. [2021] (see Table 4). We visualize perturbations in Appendix Figure 4. Note that these images are produced using the same ε bound as unlearnable examples for comparison, and for more discrete perturbations, one can easily turn down the ε radius to ensure more high fidelity user images.

Table 4: **Comparison of poisoning methods for facial recognition**. All poisons generated with $\varepsilon = 8/255$.

POISON METHOD	AVERAGE ACCURACY ($\%, \downarrow$)
CLEAN ACCURACY	86.00
UNLEARNABLE EXAMPLES [HUANG ET AL., 2021]	16.00
OURS (CT)	**8.00**

3.6 Less Data

So far, our comparisons to previous methods have been in the setting of full dataset poisoning, as was tested in Shen et al. [2019], Feng et al. [2019], Huang et al. [2021], Fowl et al. [2021]. This setting is relevant to practitioners like social media companies who wish to modify all user data to prevent others from using data scraped from their platforms.

However, even in this setting, it may be the case that an actor scraping data has access to an amount of unperturbed data - either through previous scraping before a poisoning method was employed, or data scraped from another source. A concern that follows is how this mixture affects model training. Thus, we test the effectiveness of our poisons when different proportions of clean and perturbed data are used to train the victim model. Poisons are then considered effective if their use does not significantly increase performance over training on the clean data alone. In Tables 5, 6 we find that, poisoned data does not significantly improve results over training on only the clean data and often degrades results below what one would achieve using only the clean data. This is in comparison to similar experiments in Huang et al. [2021] where poisoned data was observed to be slightly helpful when combined with clean data in training a model.

Table 5: Effects of adjusting the amount of clean data included in the poisoned CIFAR-10.

POISON METHOD\CLEAN PROPORTION	0.1	0.2	0.5	0.8
NONE (ONLY CLEAN DATA)	82.30 ± 0.08	87.90 ± 0.04	92.48 ± 0.07	93.90 ± 0.06
$\varepsilon = 8/255$	85.34 ± 0.09	88.23 ± 0.09	92.20 ± 0.08	93.65 ± 0.07

3.7 Defenses

We have demonstrated the effects of our poisons in settings where the victim trains "normally". However, there have been several defenses proposed against poisoning attacks that a victim could

Table 6: Effects of adjusting the amount of clean data included in the poisoned ImageNet.

POISON METHOD\CLEAN PROPORTION	0.1	0.2	0.5	0.8
NONE (ONLY CLEAN DATA)	45.18	53.48	62.79	64.65
$\varepsilon = 8/255$	47.81	54.24	61.20	63.95

potentially utilize. Thus, in this section, we test the effectiveness of several popular defenses against our method.

Adversarial training: Because our poisons are constrained in an ℓ_∞ ball around the clean inputs, and because we find that clean inputs are themselves adversarial examples for networks trained on poisoned data (more on this in the next section), it is possible that adversarial training could "correct" the poisoned network's behavior on the clean distribution. This is hypothesized in Tao et al. [2021], where it is argued that adversarial training can be an effective defense against delusive poisoning. We find in Table 7 that this is indeed the case. In fact, it is known that adversarial training effectively mitigates the success of several previous availability attacks including Huang et al. [2021], Feng et al. [2019], Fowl et al. [2021]. However, it is worth noting that this might not be an ideal solution for the victim as adversarial training is expensive computationally, and it degrades natural accuracy to a level well below that of standard training. For example, on a large scale dataset like ImageNet, adversarial training can result in a significant drop in validation accuracy. With a ResNet-50, adversarial training results in validation accuracy dropping from 76.13% to 47.91% - a drop that would be further exacerbated when adversarial training on poisoned data (cf. `https://github.com/MadryLab/robustness`).

Data Augmentation: Because our poisoning method relies upon slight perturbations to training data in order to degrade victim performance, it is conceivable that further modification of data during training could counteract the effects of the perturbations. This has recently been explored in Borgnia et al. [2021] and Geiping et al. [2021]. We test several data augmentations *not* known to the poisoner during crafting, although adaptive attacks have been shown to be effective against some augmentations [Geiping et al., 2021]. Our augmentations range from straightforward random additive noise (of the same ε-bound as poisons) to Gaussian smoothing. We also include popular training augmentations such as Mixup, which mixes inputs and input labels during training [Zhang et al., 2017], Cutmix [Yun et al., 2019], which mixes patches of one image with another, and Cutout [Cubuk et al., 2019] which excises certain parts of training images.

DPSGD: Differentially Private SGD (DPSGD) [Abadi et al., 2016] was originally developed as a differential privacy tool for deep networks which aims to inure a dataset to small changes in training data, which could make it an effective antitode to data poisoning. This defense adds noise to training gradients, and clips them. It was demonstrated to be successful in defending against targeted poisoning in some settings [Hong et al., 2020, Geiping et al., 2020]. However, this defense often results in heavily degraded accuracy. We test this defense with a clipping parameter 1.0 and noise parameter 0.005.

Table 7: **Effects of defenses against adversarial poisons**. All results averaged over 5 runs of a ResNet-18 victim model trained on class targeted, $\varepsilon = 8/255$ poisons. Although adversarial training improves results, none of these effectively recover the accuracy of a model trained on clean data.

DEFENSE	VALIDATION ACCURACY (%)
BASELINE (CLEAN)	94.56
ADV. TRAINING	83.01
GAUSSIAN SMOOTHING	11.94
RANDOM NOISE	6.55
MIXUP	15.86
CUTMIX	10.09
CUTOUT	8.11
DPSGD	24.61

Other than adversarial training, which has drawbacks discussed above, we find that previous proposed defenses are ineffective against our adversarial poisons, and can even degrade victim performance below the level of a model trained without any defense.

4 Analysis

Why do adversarial examples make such potent poisons? In Section 1, we motivate the effectiveness of adversarial poisons with the explanation that the perturbed data contains semantically useful information, but for the wrong class. For instance, an adversarially perturbed "dog" might be labeled as a "cat" by the crafting network because the perturbations contain discriminatory features useful for the "cat" class. In Ilyas et al. [2019], the authors discover that there exist image features which are both brittle under adversarial perturbations and useful for classification. Ilyas et al. [2019] explores these implications for creating robust models, as well as "mislabeled" datasets which produce surprisingly good natural accuracy. We investigate the implications that the existence of non-robust features have on data-poisoning attacks. We hypothesize that adversarial examples might poison networks so effectively because, for example, they teach the network to associate "cat" features found in perturbed data with the label "dog" of the original, unperturbed sample. Then, when the network tries to classify clean test-time data, it leverages the "mislabeled" features found in the perturbed data and displays low test-time accuracy. We confirm this behavior by evaluating *how* data is misclassified at test-time. We find that the distribution of predictions on *clean* data closely mimics the distributions of labels assigned by the network used for crafting after adversarial attacks, even though these patterns are generated from different networks.

To tease apart these effects, we conduct several experiments. First, we verify that the victim network does indeed train - i.e. reach a region of low loss - on the adversarial examples. This is in contrast to the motivation of [Shen et al., 2019, Huang et al., 2021] which try to *prevent* the network from training on poisoned data. We find that the victim network is able to almost perfectly fit the adversarial data. However, the accuracy of the victim network on the original, unperturbed training data is just as low as accuracy on the clean validation data, revealing an interesting duality - clean training data are adversarial examples for networks trained on their perturbed counterparts (see Table 8). But are adversarial examples simply so different from clean examples that learning on one is useless for performing inference on the other? Or do adversarial examples contain useful features but for the *wrong* class?

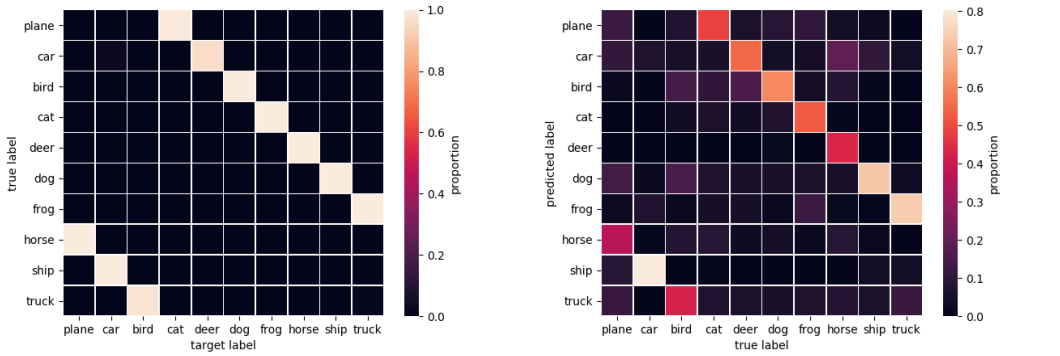
To investigate these hypotheses, we train models on the adversarial poisons crafted with class targeted objective found in Equation 4 so that every image from a given class is perturbed into the same target class. We then observe the classification patterns of the victim network and compare these to the attack patterns of the crafting network. We find that poisoned networks confuse exactly the same classes as the crafting network. Specifically, a network trained on dog images perturbed into the cat class will then misclassify clean cat images as dogs at test time as the network learns to associate "clean" cat features found in the perturbations to the training dog images with the label dog. This behavior is illustrated in Figure 2.

To further confirm our hypothesis, we employ the re-labeling trick introduced in Ilyas et al. [2019], and we train the victim network on the "incorrect" labels assigned to the poisons by the *crafting* network. For example, if a dog image is perturbed into the "cat" class by an adversarial attack on the crafting network, we train on the perturbed dog image but assign it the "cat" label. We find that this simple re-labeling trick boosts validation accuracy significantly, for example from 6.25% to 75.69% on a victim ResNet-18 model (see Appendix Table 14). This confirms the finding of Ilyas et al. [2019] concerning non-robust features. One might think that this observation is useful for defending against adversarial poisons. However, in order to perform this label correction, the victim must have access to the crafting model which requires the victim to already possess the original non-poisoned dataset.

One further intricacy remains; even if adversarial examples contained no useful features, they may still encode decision boundary information that accounts for the increased validation accuracy - behaviour that has previously been demonstrated with out-of-distribution data in Nayak et al. [2019]. However, we find that training on data from a drastically different distribution (SVHN), labeled with the CIFAR-10 crafting network's predictions, fails to achieve comparable CIFAR-10 validation accuracy. This confirms that the adversarial CIFAR-10 images contain useful features for the CIFAR-10 distribution but are simply mislabeled.

5 Limitations

While our adversarial poisoning method does outperform current art on availability poisoning attacks, there are limitations to our method that warrant future investigation. We identify two primary areas



(a) Adversarial attack predictions of network used to craft adversarial poisons.

(b) Test-time predictions of network trained on adversarial poisons.

Figure 2: Classification heatmaps. **Left** - heatmap of predictions after the adversarial attack on the network used for crafting. **Right** - heatmap of clean test predictions after training a new network on adversarial poisons.

Table 8: **Testing the victim on clean vs. adversarial training images.** Poisons are crafted on a CIFAR-10 trained ResNet-18 with 250 steps of PGD and differentiable data augmentation.

MEASUREMENT \ VICTIM	VGG19	RESNET-18	GOOGLENET	DENSENET-121	MOBILENETV2
TRAINING ACC. ON POISONS	99.95 ± 0.00	99.99 ± 0.00	99.99 ± 0.00	99.99 ± 0.00	99.94 ± 0.00
ACC. ON CLEAN TRAIN DATA	10.98 ± 0.32	6.16 ± 0.16	6.80 ± 0.08	7.06 ± 0.13	6.15 ± 0.17

of future research. First is the requirement of a clean trained model to generate adversarial examples, i.e. the θ^* parameter vector at which we craft poisons in Equation 3. This may be a reasonable assumption for practitioners like social media companies, but it does make the attack less general purpose. However, this is also a constraint with previous poisoning methods which require access to all the data being perturbed [Huang et al., 2011], or clean data to train an autoencoder [Feng et al., 2019], or a large amount of clean data to estimate an adversarial target gradient [Fowl et al., 2021]. It has been shown that adversarial examples often transfer across initialization, or even dataset, so it is not inconceivable that this limitation of our method could be overcome. The second limitation is the success of adversarial training as a defense against delusive poisoning. This appears to be the trend across all availability attacks against deep networks to date. This defense is not optimal though as it leads to a severe drop in natural accuracy, and can be computationally expensive.

6 Conclusions

Availability poisoning attacks against deep networks have increased in popularity recently, with important applications in mind like protecting released user data. Several techniques have been introduced leveraging heuristics such as loss minimization, and auto-encoder based noise. However, we find that adversarial attacks against a fixed network are more potent availability poisons than existing methods, often degrading accuracy below random guess levels. We study the effects of these poisons in multiple settings and analyze why these perturbations make such effective poisons. Our observations confirm a fundamental property of adversarial examples; they contain discriminatory features but simply for the wrong class. Our class-targeted attack leverages this property to effectively poison models on a variety of datasets.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep Learning with Differential Privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 308–318, Vienna, Austria, October 2016. Association for Computing Machinery. ISBN 978-1-4503-4139-4. doi: 10.1145/2976749.2978318.

Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 2020.

Marco Barreno, Blaine Nelson, Anthony D. Joseph, and J. D. Tygar. The security of machine learning. *Mach. Learn.*, 81(2):121–148, November 2010. ISSN 0885-6125. doi: 10.1007/s10994-010-5188-5.

Samyadeep Basu, Philip Pope, and Soheil Feizi. Influence functions in deep learning are fragile. *arXiv preprint arXiv:2006.14651*, 2020.

Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

Eitan Borgnia, Jonas Geiping, Valeriia Cherepanova, Liam Fowl, Arjun Gupta, Amin Ghiasi, Furong Huang, Micah Goldblum, and Tom Goldstein. Dp-instahide: Provably defusing poisoning and backdoor attacks with differentially private data augmentations. *arXiv preprint arXiv:2103.02079*, 2021.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

Xiaoyi Chen, Ahmed Salem, Michael Backes, Shiqing Ma, and Yang Zhang. Badnl: Backdoor attacks against nlp models. *arXiv preprint arXiv:2006.01043*, 2020.

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

Valeriia Cherepanova, Micah Goldblum, Harrison Foley, Shiyuan Duan, John Dickerson, Gavin Taylor, and Tom Goldstein. Lowkey: leveraging adversarial attacks to protect social media users from facial recognition. *arXiv preprint arXiv:2101.07922*, 2021.

Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. AutoAugment: Learning Augmentation Policies from Data. *ArXiv180509501 Cs Stat*, April 2019.

Minghong Fang, Neil Zhenqiang Gong, and Jia Liu. Influence function based data poisoning attacks to top-n recommender systems. In *Proceedings of The Web Conference 2020*, pages 3019–3025, 2020.

Ji Feng, Qi-Zhi Cai, and Zhi-Hua Zhou. Learning to Confuse: Generating Training Time Adversarial Data with Auto-Encoder. In *Advances in Neural Information Processing Systems 32*, pages 11994–12004. Curran Associates, Inc., 2019.

Liam Fowl, Ping-yeh Chiang, Micah Goldblum, Jonas Geiping, Arpit Bansal, Wojtek Czaja, and Tom Goldstein. Preventing unauthorized use of proprietary data: Poisoning for secure dataset release. *arXiv preprint arXiv:2103.02683*, 2021.

Jonas Geiping, Liam Fowl, W Ronny Huang, Wojciech Czaja, Gavin Taylor, Michael Moeller, and Tom Goldstein. Witches' brew: Industrial scale data poisoning via gradient matching. *arXiv preprint arXiv:2009.02276*, 2020.

Jonas Geiping, Liam Fowl, Gowthami Somepalli, Micah Goldblum, Michael Moeller, and Tom Goldstein. What doesn't kill you makes you robust (er): Adversarial training against poisons and backdoors. *arXiv preprint arXiv:2102.13624*, 2021.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Sven Gowal, Jonathan Uesato, Chongli Qin, Po-Sen Huang, Timothy Mann, and Pushmeet Kohli. An alternative surrogate loss for pgd-based adversarial testing. *arXiv preprint arXiv:1910.09338*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *ArXiv151203385 Cs*, December 2015.

Sanghyun Hong, Varun Chandrasekaran, Yiğitcan Kaya, Tudor Dumitraş, and Nicolas Papernot. On the Effectiveness of Mitigating Data Poisoning Attacks with Gradient Shaping. *ArXiv200211497 Cs*, February 2020.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely Connected Convolutional Networks. August 2016.

Hanxun Huang, Xingjun Ma, Sarah Monazam Erfani, James Bailey, and Yisen Wang. Unlearnable examples: Making personal data unexploitable. *arXiv preprint arXiv:2101.04898*, 2021.

Ling Huang, Anthony D. Joseph, Blaine Nelson, Benjamin I.P. Rubinstein, and J. D. Tygar. Adversarial Machine Learning. In *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, AISec '11, pages 43–58, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-1003-1. doi: 10.1145/2046684.2046692.

W Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein. Metapoison: Practical general-purpose clean-label data poisoning. *arXiv preprint arXiv:2004.00225*, 2020.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *arXiv preprint arXiv:1905.02175*, 2019.

Marc Khoury and Dylan Hadfield-Menell. On the geometry of adversarial examples. *arXiv preprint arXiv:1811.00525*, 2018.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894. PMLR, 2017.

Pang Wei Koh, Jacob Steinhardt, and Percy Liang. Stronger data poisoning attacks break data sanitization defenses. *arXiv preprint arXiv:1811.00741*, 2018.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 27–38, 2017.

Gaurav Kumar Nayak, Konda Reddy Mopuri, Vaisakh Shaj, Venkatesh Babu Radhakrishnan, and Anirban Chakraborty. Zero-shot knowledge distillation in deep networks. In *International Conference on Machine Learning*, pages 4743–4751. PMLR, 2019.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *Int J Comput Vis*, 115(3):211–252, December 2015. ISSN 1573-1405. doi: 10.1007/s11263-015-0816-y.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *ArXiv180104381 Cs*, January 2018.

J. Shen, X. Zhu, and D. Ma. TensorClog: An Imperceptible Poisoning Attack on Deep Neural Network Applications. *IEEE Access*, 7:41498–41506, 2019. ISSN 2169-3536. doi: 10.1109/ACCESS.2019.2905915.

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ArXiv14091556 Cs*, September 2014.

David Stutz, Matthias Hein, and Bernt Schiele. Disentangling adversarial robustness and generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6976–6987, 2019.

C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, June 2015. doi: 10.1109/CVPR.2015.7298594.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014. doi: 10.1109/CVPR.2014.220.

Lue Tao, Lei Feng, Jinfeng Yi, Sheng-Jun Huang, and Songcan Chen. Provable defense against delusive poisoning. *arXiv preprint arXiv:2102.04716*, 2021.

Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. Is feature selection secure against training data poisoning? In *International Conference on Machine Learning*, pages 1689–1698. PMLR, 2015.

Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. Mixup: Beyond Empirical Risk Minimization. *ArXiv171009412 Cs Stat*, October 2017.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. *arXiv preprint arXiv:1710.11342*, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 3.1
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A] We defer to the hosts of the datasets we use. We use common datasets that are available for research purposes, and we use the data purely for research purposes.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See supplementary material/code submission.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] We defer to the creators of the Webface dataset as to personally identifiable content in their dataset.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]