Witches' Brew: Industrial Scale Data Poisoning via Gradient Matching

Jonas Geiping[*]
Dep. of Electrical Engineering and Computer Science
University of Siegen
`jonas.geiping@uni-siegen.de`

Liam Fowl[*]
Department of Mathematics
University of Maryland
`lfowl@umd.edu`

W. Ronny Huang
Department of Computer Science
University of Maryland
`wronnyhuang@gmail.com`

Wojciech Czaja
Department of Mathematics
University of Maryland
`wojtek@math.umd.edu`

Gavin Taylor
Computer Science
US Naval Academy
`taylor@usna.edu`

Michael Moeller[†]
Dep. of Electrical Engineering and Computer Science
University of Siegen
`michael.moeller@uni-siegen.de`

Tom Goldstein[†]
Department of Computer Science
University of Maryland
`tomg@umd.edu`

Abstract

Data Poisoning attacks involve an attacker modifying training data to maliciously control a model trained on this data. Previous poisoning attacks against deep neural networks have been limited in scope and success, working only in simplified settings or being prohibitively expensive for large datasets. In this work, we focus on a particularly malicious poisoning attack that is both "from scratch" and "clean label", meaning we analyze an attack that successfully works against new, randomly initialized models, and is nearly imperceptible to humans, all while perturbing only a small fraction of the training data. The central mechanism of this attack is matching the gradient direction of malicious examples. We analyze why this works, supplement with practical considerations. and show its threat to real-world practitioners, finding that it is the first poisoning method to cause targeted misclassification in modern deep networks trained from scratch on a full-sized, poisoned ImageNet dataset. Finally we demonstrate the limitations of existing defensive strategies against such an attack, concluding that data poisoning is a credible threat, even for large-scale deep learning systems.

1 Introduction

Machine learning models have quickly become the backbone of many applications from photo processing on mobile devices and ad placement to security and surveillance [26]. These applications often rely on large training datasets that aggregate samples of unknown origins, and the security implications of this are not yet fully understood [38]. Data is often sourced in a way that lets malicious outsiders contribute to the dataset, such as scraping images from the web, farming data from website users, or using large academic datasets scraped from social media [58]. *Data Poisoning* is a security

[*]Authors contributed equally.
[†]Authors contributed equally.
Correspondence to: Jonas Geiping, Liam Fowl, Michael Moeller, and Tom Goldstein.

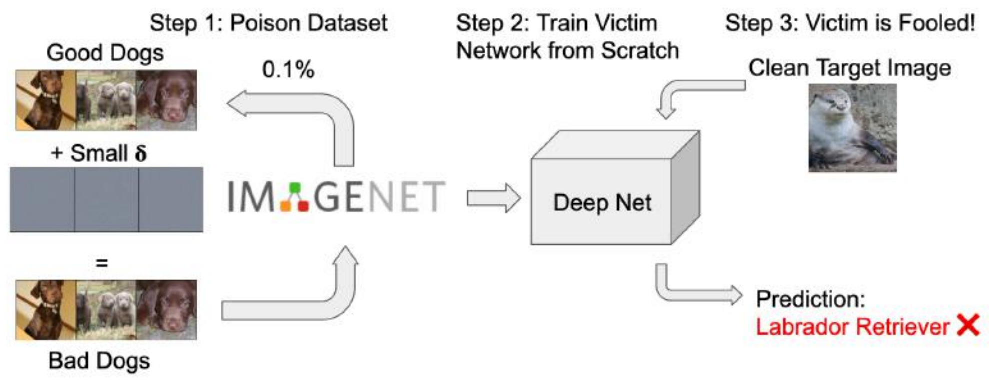


Figure 1: A schematic of the poisoning pipeline - including poison images with imperceptible noise that successfully poisoned a ResNet-18 trained from scratch on ImageNet [45]. Poisoned images (labrador retriever class) successfully caused a randomly initialized victim ResNet-18 to mis-classify a target (otter) image. This is accomplished with a poison budget of 0.1% and an ℓ_∞ bound of $\varepsilon = 8$. Further visualizations of poisoned data can be found in the supp. material.

threat in which an attacker makes imperceptible changes to data that can then be disseminated through social media, smartphones, or public datasets without being caught by human supervision. The goal of a poisoning attack is to impact a trained model to achieve a malicious goal. Mainstream poisoning research has focused on attacks that achieve mis-classification of predetermined inference data [5, 56, 49] and can be applied in scenarios such as social recommendation [18], content management [27, 11], algorithmic fairness [54] and biometric recognition [30]. Other potential goals of the attacker include denial-of-service [55, 52], concealment of users [51], and introduction of fingerprint information [31]. Accordingly, industry practitioners ranked data poisoning as the most serious attack on ML systems in a recent survey of corporations [25].

In this work we discuss how to efficiently create poisoned data - even in the setting of deep neural networks trained on large image classification tasks, such as ImageNet [44]. Previous work on data poisoning has often focused on either linear classification tasks [5, 60, 23] or poisoning of transfer learning and fine tuning [49, 22, 63] rather than a full end-to-end training pipeline. Poison attacks on deep neural networks (and especially on ones trained from scratch) have proven difficult [35, 49]. Only very recently was it possible to adapt these methods to neural networks retrained from scratch on CIFAR-10 [19]. However, this comes with significant costs that render scaling to more complex settings, including the ImageNet dataset, prohibitively expensive [19].

We formulate data poisoning as the problem of solving a gradient matching problem and analyze a novel attack algorithm that scales to unprecedented dataset size and effectiveness. Crucially, the new poisoning objective is orders-of-magnitude more efficient than a previous formulation based on on meta learning [19]. We follow with an experimental evaluation, showing that poisoned datasets created by this method are more robust and effective than other approaches, when comparing on CIFAR-10. We then demonstrate reliably successful attacks on commonly used models trained on ImageNet in realistic training scenarios. For example, the attack successfully compromises a ResNet-18 by manipulating only 0.01% of the data points with perturbations less than 8 pixel values in ℓ_∞-norm. We close by discussing previous defense strategies and how strong differential privacy [1] is the only existing defense that can partially mitigate the effects of the attack.

2 Related Work

The task of data poisoning is closely related to the problem of adversarial attacks at test time, also referred to as evasion attacks [57, 33, 6, 2], where the attacker alters a target test image to fool an already-trained model. This attack is applicable in scenarios where the attacker has control over the target image, but not over the training data. An intermediary between data poisoning and adversarial attacks are backdoor trigger attacks [59, 28, 46, 14]. These attacks involve inserting a trigger – often an image patch – into training data, which is later activated by also applying the trigger to test images to cause mis-classification. Backdoor attacks require perturbations to both training and test-time data – a more permissive threat model than either poisoning or evasion.

In contrast to evasion and backdoor attacks, data poisoning attacks consider a setting where the attacker can modify training data, but does not have access to test data. Within this setting we focus on *targeted attacks* – attacks that aim to cause a specific target test image (or set of target test images) to be mis-classified. For example, an attack may cause a certain target image of a dog to be classified as a bird by victim models at test time. This is difficult to detect, because it does not noticeably degrade either training or validation accuracy [49, 19].

Two basic schemes for targeted poisoning are label flipping [4, 41], and watermarking [56, 49]. In the case of label flipping attacks, an attacker is allowed to change the class label of training examples. Conversely, in a watermarking attack, the attacker perturbs the training image, not label, by superimposing a target image onto training images.These attacks can be successful, yet they are easily detected by human or machine supervision [39]. This is in contrast to *clean-label* attacks which maintain the semantic labels of data.

Mathematically speaking, data poisoning is a *bilevel* optimization problem [3, 5]; the attacker optimizes image pixels to enforce (malicious) criteria on the resulting network parameters, which are themselves the solution to an "inner" optimization problem that minimizes the training objective. Direct solutions to the bilevel problem have been proposed where feasible, for example, SVMs [5] or logistic regression [9]. However, direct optimization of the poisoning objective is intractable for deep neural networks because it requires backpropagating through the entire SGD training procedure [35]. As such, the bilevel objective has to be approximated. Recently, MetaPoison [19] proposed to approximately solve the bi-level problem based on methods from the meta-learning community [12]. The bilevel gradient is approximated by backpropagation through several unrolled gradient descent steps. This is the first attack to succeed against deep networks on CIFAR-10 as well as providing transferability to other models. However, [19] uses a complex loss function averaged over a wide range of models trained to different epochs and a single unrolling step necessarily involves both clean and poisoned training data, making it roughly as costly as one epoch of standard training. With an ensemble of 24 models, [19] requires 3 (2 unrolling steps + 1 clean update step) x 2 (backpropagation through unrolled steps) x 60 (first-order optimization steps) x 24 (ensemble of models) equivalent epochs of normal training to create the poisons, as well as ($\sum_{k=0}^{23} k = 253$) epochs of pretraining. All in all, this equates to 8893 training epochs.

In contrast to bilevel approaches stand heuristics for data poisoning of neural networks. The most prominent heuristic is *feature collision*, as in Poison Frogs [49], which seeks to cause a target test image to be misclassified by perturbing training data to collide with the target image in feature space. The approach can be modified by surrounding the target image in feature space with a convex polytope formed by poisoned training images [63]. These methods are efficient, but designed to attack fine-tuning scenarios where the feature extractor is fixed. When applied to deep networks trained from scratch, their performance drops significantly.

3 Efficient Poison Brewing

We discuss a poisoning objective that combines the best of both worlds. Gradient matching allows for data poisoning as efficiently as in Poison Frogs [49], requiring only a single pretrained model and a time budget on the order of one epoch of training for optimization - but is still capable of poisoning the from-scratch setting considered in [19]. This combination allows us to "brew" poisons that successfully attack realistic models on ImageNet.

In this section, we will discuss an intriguing weakness of neural network training based on first-order optimization and derive an attack against it. Our attack modifies training images within a small ℓ_∞ ball so they produce a malicious gradient signal during training.

3.1 Threat Model

We define two parties, the *attacker*, which has limited control over the training data, and the *victim*, which trains a model based on this data. We first consider a gray-box setting, where the attacker has knowledge of the model architecture used by its victim. The attacker is permitted to poison a fraction of the training dataset (usually less than 1%) by changing images within an ℓ_∞-norm ε-bound (usually with $\varepsilon \leq 16$). This setup enforces *clean-label* attacks, meaning that the semantic label of a poisoned image is still unchanged. The attacker has no knowledge of the training procedure - neither about the initialization of the victim's model, nor about the (randomized) mini-batching and data augmentation that is standard in the training of deep learning models.

We formalize our threat model as bilevel problem for a machine learning model $F(x, \theta)$ with inputs $x \in \mathbb{R}^n$ and parameters $\theta \in \mathbb{R}^p$, and a loss function $\mathcal{L}$. We denote the N training samples by $(x_i, y_i)_{i=1}^N$, from which a subset of P samples are poisoned. For notation simplicity we assume the first P training images are poisoned, and this is done by adding a perturbation Δ_i to the i^{th} training image. The perturbation is constrained to be smaller than ε in the ℓ_∞-norm. The task is to optimize this perturbation so that a set of T target samples $(x_i^t, y_i^t)_{i=1}^T$ is reclassified with the new adversarial labels y_i^{adv}:

$$\min_{\Delta \in \mathcal{C}} \sum_{i=1}^{T} \mathcal{L}\left(F(x_i^t, \theta(\Delta)), y_i^{\text{adv}}\right) \quad \text{s.t. } \theta(\Delta) \in \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(F(x_i + \Delta_i, \theta), y_i). \tag{1}$$

We subsume the constraints in the set $\mathcal{C} = \{\Delta \in \mathbb{R}^{N \times n} : ||\Delta||_\infty \leq \varepsilon, \Delta_i = 0 \; \forall i > P\}$. We call the main objective on the left the *adversarial loss*, and the objective that appears in the constraint on the right is the *training loss*. For the remainder of the paper, we consider a single target image ($T = 1$) as in [49, 63], but stress that this is not a general limitation of targeted methods, as shown in [19].

3.2 Motivation

What is the optimal alteration of the training set that causes a victim neural network $F(x, \theta)$ to mis-classify a specific target image x^t? We know that the expressivity of deep networks allows them to fit arbitrary training data [61]. Thus, if an attacker was unconstrained, a straightforward way to cause targeted mis-classification of an image is for the attacker to insert the target image, with the incorrect label y^{adv}, into the victim network's training set. Then, when the victim minimizes the training loss they simultaneously minimize the adversarial loss, based on the gradient information about the target image. In a more realistic setting, the attacker may not be able to insert the mis-labeled target. They could, however, still mimic the gradient of the target by creating poison images whose training gradient correlates with the adversarial target gradient. If the attacker can enforce

$$\nabla_\theta \mathcal{L}(F(x^t, \theta), y^{\text{adv}}) \approx \frac{1}{P} \sum_{i=1}^{P} \nabla_\theta \mathcal{L}(F(x_i + \Delta_i, \theta), y_i) \tag{2}$$

to hold for any θ encountered during training, then the victim's gradient steps that minimizes the training loss on the poisoned images (right hand side) will also minimize the attackers adversarial loss on the targeted images (left side).

3.3 The Central Mechanism: Gradient Alignment

Gradient magnitudes vary dramatically across different stages of training, and so finding poisoned images that satisfy eq. (2) for all θ encountered during training may be infeasible. Instead we *align* the target and poison gradients in the same direction, that is we minimize their negative cosine similarity. We do this by training a clean model F with parameters θ, keeping θ fixed, and then optimizing the straightforward objective

$$\mathcal{B}(\Delta, \theta) = 1 - \frac{\left\langle \nabla_\theta \mathcal{L}(F(x^t, \theta), y^{\text{adv}}), \sum_{i=1}^{P} \nabla_\theta \mathcal{L}(F(x_i + \Delta_i, \theta), y_i) \right\rangle}{\|\nabla_\theta \mathcal{L}(F(x^t, \theta), y^{\text{adv}})\| \cdot \| \sum_{i=1}^{P} \nabla_\theta \mathcal{L}(F(x_i + \Delta_i, \theta), y_i)\|} \tag{3}$$

We optimize $\mathcal{B}(\Delta)$ using signed Adam updates with decaying step size, projecting onto $\mathcal{C}$ after every step. This produces an alignment between the average poison gradients and the average target gradients. In contrast to Poison Frogs [49], all layers of the network are included (via their parameters) in this objective, not just the last feature layer.

Each optimization step of this attack requires only a *single* differentiation of the parameter gradient w.r.t to its input, instead of differentiating through several unrolled steps as in MetaPoison [19]. This is not significantly more expensive than the feature matching of Poison Frogs [49]. Furthermore, as in Poison Frogs we differentiate through a loss that only involves the (small) subset of poisoned data instead of involving the entire dataset. Finally, the method is able to create poisons using only a single parameter vector, θ (like Poison Frogs in fine-tuning setting, but not the case for MetaPoison) and does not require updating this parameter vector after each poison optimization step.

Remark. Inner-product loss functions like eq. (3) work well in other contexts. In [13], cosine similarity between image gradients was minimized to uncover training images used in federated learning. If we disable our constraints, setting $\varepsilon = 255$, and consider a single poison image and a single target, then we minimize the problem of recovering image data from a normalized gradient as a special case. In [13], it was shown that minimizing this problem can recover the target image. This means that we can indeed return to the motivating case in the unconstrained setting - the optimal choice of poison data is insertion of the target image in an unconstrained setting for one image.

3.4 Making attacks that transfer and succeed "in the wild"

A practical and robust attack must be able to poison different random initializations of network parameters and a variety of architectures. To this end, we test several techniques:

Differentiable Data Augmentation and Resampling: Data augmentation is a standard tool in deep learning, and transferable image perturbations must survive this process. At each step minimizing eq. (3), we randomly draw a translation, crop, and possibly a horizontal flip for each poisoned images, then use bilinear interpolation to resample to the original resolution. When updating Δ, we differentiate through this grid sampling operation as in [20]. This creates an attack which is robust to data augmentation and leads to increased transferability.

Restarts: The efficiency we gained in section 3.3 allows us to incorporate restarts, a common technique in the creation of adversarial examples at test time [43, 34]. We minimize eq. (3) several times from random starting perturbations, and select the set of poisons that give us the lowest alignment loss. This allows us to trade off reliability with computational effort.

Model Ensembles: A known approach to improving transferability is to attack an ensemble of model instances trained from different initializations [19, 63, 29]. However, ensembles are highly expensive, increasing the pre-training cost for a modest, but stable, increase in performance.

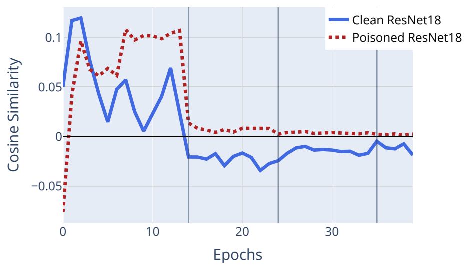
We show the effects of these techniques via CIFAR-10 experiments (see table 1 and section 5.1). To keep the attack within practical reach, we do not consider ensembles for our experiments on ImageNet data, opting for the cheaper techniques of restarts and data augmentation. A summarizing description of the attack can be found in algorithm 1. Lines 8 and 9 of algorithm 1 are done in a stochastic (mini-batch) setting (which we omitted in algorithm 1 for notation simplicity).

4 Theoretical Analysis

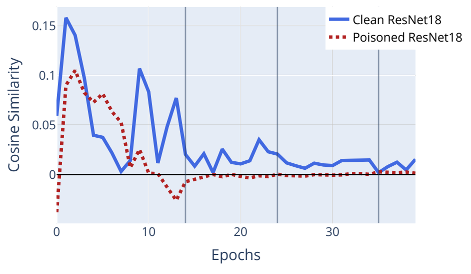
Can gradient alignment cause network parameters to converge to a model with low adversarial loss? To simplify presentation, we denote the adversarial loss and normal training loss of eq. (1) as $\mathcal{L}_{\text{adv}}(\theta) =: \mathcal{L}(F((x^t, \theta), y^{\text{adv}})$ and $\mathcal{L}(\theta) =: \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(x_i, y_i, \theta)$, respectively. Also, recall that $1-\mathcal{B}\left(\Delta, \theta^k\right)$, defined in eq. (3), measures the cosine similarity between the gradient of the adversarial

Algorithm 1 Poison Brewing via the discussed approach.

```
 1: Require Pretrained clean network $\{F(\cdot, \theta)\}$, a training set of images and labels $(x_i, y_i)_{i=1}^N$, a
    target $(x^t, y^{\text{adv}})$, $P < N$ poison budget, perturbation bound $\varepsilon$, restarts $R$, optimization steps $M$
 2: Begin
 3: Select $P$ training images with label $y^{\text{adv}}$
 4: For $r = 1, \ldots, R$ restarts:
 5:     Randomly initialize perturbations $\Delta^r \in \mathcal{C}$
 6:     For $j = 1, \ldots, M$ optimization steps:
 7:         Apply data augmentation to all poisoned samples $(x_i + \Delta_i^r)_{i=1}^P$
 8:         Compute the average costs, $\mathcal{B}(\Delta^r, \theta)$ as in eq. (3), over all poisoned samples
 9:         Update $\Delta^r$ with a step of signed Adam and project onto $||\Delta^r||_\infty \leq \varepsilon$
10: Choose the optimal $\Delta^*$ as $\Delta^r$ with minimal value in $\mathcal{B}(\Delta^r, \theta)$
11: Return Poisoned dataset $(x_i + \Delta_i^*, y_i)_{i=1}^N$
```



(a) Alignment of $\nabla\mathcal{L}_{\text{adv}}(\theta)$ and $\nabla\mathcal{L}(\theta)$



(b) Alignment of $\nabla\mathcal{L}_{\text{t}}(\theta)$ (orig. label) and $\nabla\mathcal{L}(\theta)$

Figure 2: Average batch cosine similarity, per epoch, between the adversarial gradient and the gradient of each mini-batch (left), and with its clean counterpart $\nabla\mathcal{L}_{\text{t}}(\theta) := \nabla_\theta\mathcal{L}(x^t, y^t)$ (right) for a poisoned and a clean ResNet-18. Each measurement is averaged over an epoch. Learning rate drops are marked with gray vertical bars.

loss and the gradient of normal training loss. We adapt a classical result of Zoutendijk [37, Thm. 3.2] to shed light on why data poisoning can work even though the victim only performs standard training on a poisoned dataset:

Proposition 1 (Adversarial Descent Lemma)**.** *Let $\mathcal{L}_{adv}(\theta)$ be bounded below and have a Lipschitz continuous gradient with constant $L > 0$ and assume that the victim model is trained by gradient descent with step sizes α_k, i.e. $\theta^{k+1} = \theta^k - \alpha_k \nabla\mathcal{L}(\theta^k)$. If the gradient descent steps $\alpha_k > 0$ satisfy*

$$\alpha_k L < \beta \left(1 - \mathcal{B}(\Delta, \theta^k)\right) \frac{||\nabla\mathcal{L}(\theta^k)||}{||\nabla\mathcal{L}_{adv}(\theta^k)||} \tag{4}$$

for some fixed $\beta < 1$, then $\mathcal{L}_{adv}(\theta^{k+1}) < \mathcal{L}_{adv}(\theta^k)$. If in addition $\exists \varepsilon > 0$, k_0 so that $\forall k \geq k_0$, $\mathcal{B}(\Delta, \theta^k) < 1 - \varepsilon$, then

$$\lim_{k\to\infty} ||\nabla\mathcal{L}_{adv}(\theta^k)|| \to 0. \tag{5}$$

Proof. See supp. material. □

Put simply, our poisoning method aims to align the gradients of training loss and adversarial loss. This enforces that the gradient of the main objective is a descent direction for the adversarial objective, which, when combined with conditions on the step sizes, causes a victim to unwittingly converge to a stationary point of the adversarial loss during training. The strongest assumption in Proposition 1 is that gradients are almost always aligned, $\mathcal{B}(\Delta, \theta^k) < 1 - \epsilon, k \geq k_0$. We directly maximize alignment during creation of the poisoned data, but only for a selected θ^*, and not for all θ^k encountered during gradient descent from any possible initialization.

However, poison perturbations made from one parameter vector, θ, can transfer to other parameter vectors encountered during training. For example, if one allows larger perturbations, and in the limiting case, unbounded perturbations, our objective is minimal if the poison data is identical to the target image, which aligns training and adversarial gradients at every θ encountered. Empirically, we see that the proposed "poison brewing" attack does indeed increase gradient alignment. In fig. 2, we see that in the first phase of training all alignments are positive, but only the poisoned model maintains a positive similarity for the adversarial target-label gradient throughout training. The clean model consistently shows that these angles are negatively aligned - i.e. normal training on a clean dataset will increase adversarial loss. However, after the inclusion of poisoned data, the gradient alignment is modified enough to change the prediction for the target. Figure 2b shows the reverse effect: under standard training (no poisoned data), the clean gradient is a descent direction for the loss on the target's original label ("dog" for a "dog" target image) and not for the poisoned model.

Remark. An analysis of how gradient alignment often transfers between different parameters and even between architectures has been conducted, e.g. in [8] and [22]. It was shown in [9] that the performance loss when transferring an adversarial attack to another model is governed by the gradient alignment of both models. In the same vein, optimizing alignment appears to be a useful metric in

Table 1: CIFAR-10 ablation. $\varepsilon = 16$, budget is 1%. Differentiable data augmentation yields robust strong as an 8-model ensemble, but without increasing computational effort.

Ensemble	**Diff. Data Aug.**	**Victim does data aug.**	**Poison Accuracy** ($\%(\pm$SE$)$)
1	X	X	100.00% (±0.00)
1	X	✓	32.50% (±12.27)
8	X	X	78.75% (±11.77)
1	✓	✓	91.25% (±6.14)

Table 2: CIFAR-10 Comparison to other possible poisoning objectives with a budget of 1% within our efficiency constraints, for two scenarios. MetaPoison* denotes only the objective of [19], not the full framework. Each cell shows the avg. poison success and its standard error.

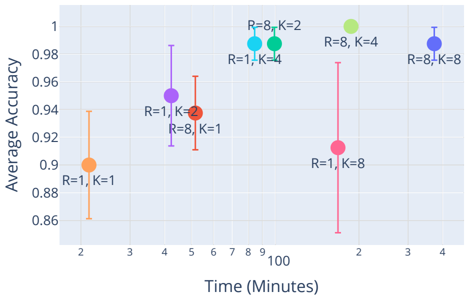
	Gradient Matching (Prop.)	Poison Frogs [49]	MetaPoison* [19]
6-layer ConvNet ($\varepsilon = 32$)	*86.25%* (±9.43)	78.75% (±7.66)	0.00% (±0.00)
ResNet-18 ($\varepsilon = 16$)	*90.00%* (±3.87)	3.75% (±3.56)	0.00% (±0.00)

the case of data poisoning. Furthermore [17] note that previous poisoning algorithms might already cause gradient alignment as a side effect, even without explicitly optimizing for it.

5 Experimental Evaluation

We evaluate poisoning approaches in each experiment by sampling 10 random poison-target cases. We compute poisons for each and evaluate them on 8 newly initialized victim models (see supp. material Sec. A.1 for details of our methodology). We use the following hyperparameters for our experiments: $\tau = 0.1$, $R = 8$, $M = 250$. We train victim models in a realistic setting, considering data augmentation, SGD with momentum, weight decay and learning rate drops. Code for all experiments can be found at `https://github.com/JonasGeiping/poisoning-gradient-matching`.

5.1 Baseline Evaluations on CIFAR-10



As a baseline on CIFAR-10, we compare the number of restarts R and the number of ensembled models K, showing that the proposed method is successful in creating poisons even with just a single model (instead of an ensemble). The inset figure shows poison success versus time necessary to compute the poisoned dataset for a budget of 1%, $\varepsilon = 16$ on CIFAR-10. The network is a ResNet-18. We find that as the number of ensemble models, K, increases, it is beneficial to increase the number of restarts as well, but increasing the number of restarts independently also improves performance. We validate the differentiable data augmentation discussed in section 3.4 in table 1, finding it crucial for scalable data poisoning, being as efficient as a large model ensemble in facilitating robustness.

Next, to test different poisoning methods, we fix our "brewing" framework of efficient data poisoning, with only a single network and diff. data augmentation. We evaluate the discussed gradient matching cost function, replacing it with either the feature-collision objective of Poison Frogs [49] (thereby effectively replicating their method, but in our context of from-scratch training), or the MetaPoison [19] objective of an unrolled adversarial loss.

The results of this comparison are collated in table 2. While poison frogs succeeds in a finetuning setting [49], we find that its feature collision objective is only successful in the shallower network when using from-scratch training. We also discover that the unrolled adversarial loss of MetaPoison fails when applied to just one source model, requiring the computationally expensive staggered ensemble of models that are updated after every evaluation of the objective. Without this construction it does not lead to a successful poison attack within the threat model of table 2. We also find that our method does not require knowledge of the victim's entire training set, and a victim can still be poisoned when using only a small fraction of the training images (see table 7 in supp. material).

Table 3: Results on the benchmark of [48]. Avg. accuracy of poisoned CIFAR-10 (budget 1%, $\varepsilon = 8$) over 100 trials is shown. (*) denotes rows replicated from [48]. Poisons are created assuming a ResNet-18 except for the last row, where the ensemble consists of two models of each architecture.

Attack	**ResNet-18**	**MobileNet-V2**	**VGG11**	**Average**
Poison Frogs* [49]	0%	1%	3%	1.33%
Convex Polytopes* [63]	0%	1%	1%	0.67%
Clean-Label Backdoor* [59]	0%	1%	2%	1.00%
Hidden-Trigger Backdoor* [46]	0%	4%	1%	2.67%
Proposed Attack ($K = 1$)	45%	36%	8%	29.67%
Proposed Attack ($K = 4$)	$\mathbf{55}\%$	37%	7%	33.00%
Proposed Attack ($K = 6$, Heterogeneous)	49%	$\mathbf{38}\%$	$\mathbf{35}\%$	$\mathbf{40.67}\%$

Benchmark results on CIFAR-10:
To evaluate our results against a wider range of poison attacks, we consider the recent benchmark proposed in [48]. In the category "Training From Scratch", this benchmark evaluates poisoned CIFAR-10 datasets with a budget of 1% and $\varepsilon = 8$ against various model architectures, averaged over 100 fixed scenarios. Most notably it shows not only that Poison Frogs [49] and Convex Polytopes [63] struggle under this threat model, but also the stronger backdoor attacks, such as clean-label backdoor attacks [59] and hidden-trigger backdoor attacks [46], which also allow for a modification of the target image.

We find that the discussed poison brewing attack is significantly more potent, even in the more difficult benchmark setting (ε is halved, compared to our CIFAR-10 experiments). As in our experiments, increasing the number of ensembled models K increases performances, but a single model already performs well. An additional feature of the benchmark is *transferability*. Poisons are created using a ResNet-18 model, but evaluated also on two other architectures. We find that datasets poisoned by the discussed attack transfer reasonably well to the similar MobileNet-V2 [47] architecture, but not as well to VGG11 [53]. However, we also show that this advantage can be easily circumvented by using an ensemble of different models as in [63]. If we use a heterogeneous ensemble of $K = 6$, consisting of 2 ResNet-18, 2 MobileNet-V2 and 2 VGG11 models (last row), then the same poisoned dataset can compromise all models and generalize across architectures.

5.2 Poisoning ImageNet models

The ILSVRC2012 challenge, "ImageNet", consists of 1000 classes and over 1 million training examples, making it infeasible for most actors to train large model ensembles or run extensive hyperparameter optimizations. However, as the new attack requires only a single sample of pretrained parameters θ, and operates only on the poisoned subset, we can create poisoned ImageNet images using publicly available pretrained models without ever training an ImageNet classifier from scratch. Poisoning ImageNet with previous methods would be infeasible. For example, following the calculations in section 2, it would take over 500 GPU days (relative to our hardware) to create a poisoned ImageNet for a ResNet-18 via MetaPoison. In contrast, the new attack can poison ImageNet in less than four GPU hours.

Figure 3 shows that a standard ResNet-18 ImageNet model trained from scratch on a poisoned dataset "brewed" with the discussed attack, is reliably compromised - with examples of successful poisons generated by our method shown (left). We first study the effect of varying poison budgets, and ε-bounds (top right). Even at a budget of 0.05% and ε-bound of 8, the attack poisons a randomly initialized ResNet-18 80% of the time. These results extend to other popular models (bottom right).

Poisoning Cloud AutoML:
To verify that the discussed attack can compromise models in practically relevant *black-box setting*, we test our method on Google's Cloud AutoML. This is a cloud framework that provides access to black-box ML models based on an uploaded dataset. In [19] Cloud AutoML was shown to be vulnerable against poisoned CIFAR-10 datasets. We create and upload a poisoned ImageNet dataset (brewing model: ResNet18, budget 0.1%, $\varepsilon = 32$) for our first poison-target test case and upload the dataset. Even in this scenario, the attack is measurably effective, moving the adversarial label into the top-5 predictions of the model in 5 out of 5 runs, and the top-1 prediction in 1 out of 5 runs.

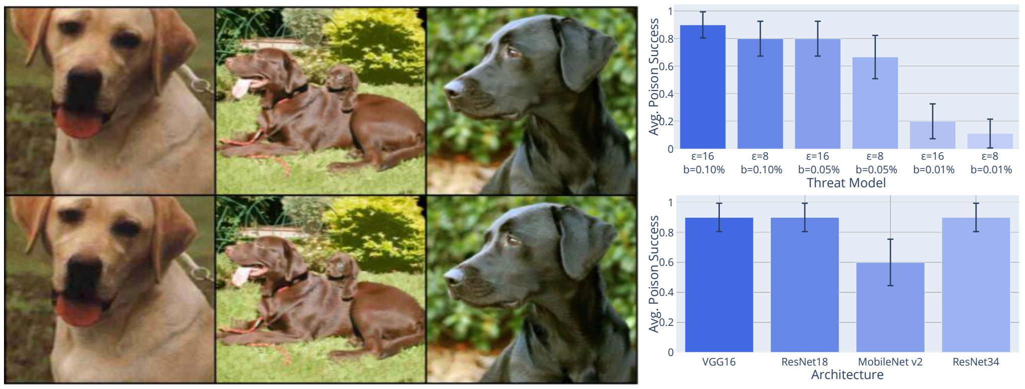
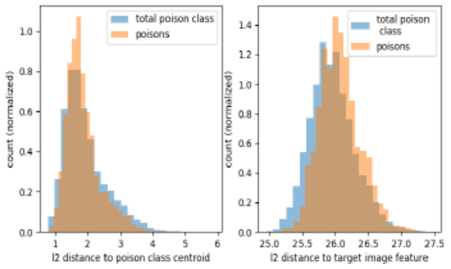
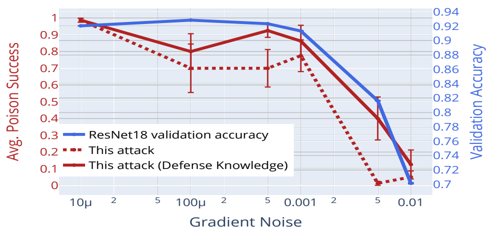


Figure 3: Poisoning ImageNet. **Left**: Clean images (above), with their poisoned counterparts (below) from a successful poisoning of a randomly initialized ResNet-18 trained on ImageNet for a poison budget of 0.1% and an ℓ_∞ bound of $\varepsilon = 8$. **Right Top**: ResNet-18 results for different budgets and varying ε-bounds. **Right Bot.**: More architectures [15, 47] with a budget of 0.1% and $\varepsilon = 16$.



(a) Feature space distance to base class centroid, and target image feature, for successfully poisoned victim model on CIFAR-10. Budget of 4.0% and an ℓ_∞ bound of $\varepsilon = 16$, showing sanitization defenses failing and no feature collision as in [49].



(b) Defending through differential privacy. CIFAR-10, 1% budget, $\varepsilon = 16$, ResNet-18. Differential privacy is only able to limit the success of poisoning by trading off with significant drops in accuracy even on a simple dataset.

Figure 4: Defense strategies against poisoning.

5.3 Deficiencies of Defense Strategies

Previous defenses against data poisoning [55, 40, 42] have relied mainly on data sanitization, i.e. trying to find and remove poisons by outlier detection (often in feature space). We demonstrate why sanitization methods fail in the face of the attack discussed in this work in fig. 4a. Poisoned data points are distributed like clean data points, reducing filtering based methods to almost-random guessing (see supp. material, table 6). Another defense using differentially private training diminishes the impact of individual training samples, in turn making poisoned data less effective [32, 17]. However, this come at a significant cost. Figure 4b shows that the natural (=normal) validation accuracy is reduced significantly when gradient noise from differential privacy is large enough to affect poisoning success. To push the Poison Success below 15%, one has to sacrifice over 20% validation accuracy, even on CIFAR-10. Training a diff. private ImageNet model is even more challenging. From this aspect, differentially private training can be compared to adversarial training [33] against evasion attacks. Both methods can mitigate the effectiveness of an adversarial attack, but only by significantly impeding natural accuracy.

6 Conclusion

We investigate data poisoning via gradient matching and discover that this mechanism allows for data poisoning attacks against fully retrained models that are unprecedented in scale and effectiveness. We motivate the attack theoretically and empirically, discuss additional mechanisms like diff. data augmentation and experimentally investigate modern deep neural networks in realistic training scenarios, showing that gradient matching attacks compromise even models trained on ImageNet. We close with discussing the limitations of current defense strategies.

Broader Impact - Poisoning is a Credible Threat to Deep Neural Networks

It is important to understand the security impacts of using unverified data sources for deep network training. Data poisoning attacks up to this point have been limited in scope. Such attacks focus on limited settings such as poisoning SVMs, attacking transfer learning models, or attacking toy architectures [5, 36, 49]. We demonstrate that data poisoning poses a threat to large-scale systems as well. The approach discussed in this work pertains only to the classification scenario, as a guinea pig for data poisoning, but applications to a variety of scenarios of practical interest have been considered in the literature, for example spam detectors mis-classifying a spam email as benign, or poisoning a face unlock based mobile security systems.

The central message of the data poisoning literature can be described as follows: From a security perspective, the data that is used to train a machine learning model should be under the same scrutiny as the model itself. These models can only be secure if the entire data processing pipeline is secure. This issue further cannot easily be solved by human supervision (due to the existence of clean-label attacks) or outlier detection (see fig. 4a). Furthermore, targeted poisoning is difficult to detect as validation accuracy is unaffected. As such, data poisoning is best mitigated by fully securing the data pipeline.

So far we have considered data poisoning from the industrial side. From the perspective of a user, or individual under surveillance, however, data poisoning can be a means of securing personal data shared on the internet, making it unusable for automated ML systems. For this setting, we especially refer to an interesting application study in [51] in the context of facial recognition.

Finally, we acknowledge that the publication of new attacks has the potential to empower exploitation of current security issues, but the only way forward is to openly discuss state of the art vulnerabilities in ML systems, and their limitations enabling further discussion and development of mitigation strategies.

Acknowledgments and Disclosure of Funding

We thank the University of Maryland Institute for Advanced Computer Studies, specifically Matthew Baney and Liam Monahan, for their help with the Center for Machine Learning cluster.

This work was supported by DARPA's GARD, QED4RML, and Young Faculty Award program. Additional support was provided by the National Science Foundation Directory of Mathematical Sciences. GT is supported by ONR grant N0001420WX00239, as well as the DoD HPC Modernization Program. WC is supported by NSF DMS 1738003 and MURI from the Army Research Office - Grant No. W911NF-17-1-0304.

References

[1] 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.

[2] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. *ArXiv180200420 Cs*, February 2018.

[3] Jonathan F. Bard and James E. Falk. An explicit solution to the multi-level programming problem. *Computers & Operations Research*, 9(1):77–100, January 1982.

[4] Marco Barreno, Blaine Nelson, Anthony D. Joseph, and J. D. Tygar. The security of machine learning. *Mach. Learn.*, 81(2):121–148, November 2010.

[5] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning Attacks against Support Vector Machines. *ArXiv12066389 Cs Stat*, June 2012.

[6] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. *ArXiv160804644 Cs*, August 2016.

[7] Nicholas Carlini and David Wagner. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, pages 3–14, Dallas, Texas, USA, November 2017. Association for Computing Machinery.

[8] Guillaume Charpiat, Nicolas Girard, Loris Felardos, and Yuliya Tarabalka. Input Similarity from the Neural Network Perspective. In *Advances in Neural Information Processing Systems 32*, pages 5342–5351. Curran Associates, Inc., 2019.

[9] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. Why Do Adversarial Attacks Transfer? Explaining Transferability of Evasion and Poisoning Attacks. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 321–338, 2019.

[10] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. A Rotation and a Translation Suffice: Fooling CNNs with Simple Transformations. *ArXiv171202779 Cs Stat*, December 2017.

[11] Minghong Fang, Guolei Yang, Neil Zhenqiang Gong, and Jia Liu. Poisoning Attacks to Graph-Based Recommender Systems. In *Proceedings of the 34th Annual Computer Security Applications Conference*, ACSAC '18, pages 381–392, San Juan, PR, USA, December 2018. Association for Computing Machinery.

[12] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *ArXiv170303400 Cs*, March 2017.

[13] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting Gradients – How easy is it to break privacy in federated learning? *ArXiv200314053 Cs*, March 2020.

[14] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. *IEEE Access*, 7:47230–47244, 2019.

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

[16] Wen Heng, Shuchang Zhou, and Tingting Jiang. Harmonic Adversarial Attack Method. *ArXiv180710590 Cs*, July 2018.

[17] 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.

[18] Rui Hu, Yuanxiong Guo, Miao Pan, and Yanmin Gong. Targeted Poisoning Attacks on Social Recommender Systems. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, December 2019.

[19] W. Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein. MetaPoison: Practical General-purpose Clean-label Data Poisoning. *ArXiv200400225 Cs Stat*, April 2020.

[20] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and koray kavukcuoglu. Spatial Transformer Networks. In *Advances in Neural Information Processing Systems 28*, pages 2017–2025. Curran Associates, Inc., 2015.

[21] Danny Karmon, Daniel Zoran, and Yoav Goldberg. LaVAN: Localized and Visible Adversarial Noise. *ArXiv180102608 Cs*, January 2018.

[22] Pang Wei Koh and Percy Liang. Understanding Black-box Predictions via Influence Functions. In *International Conference on Machine Learning*, pages 1885–1894, July 2017.

[23] Pang Wei Koh, Jacob Steinhardt, and Percy Liang. Stronger Data Poisoning Attacks Break Data Sanitization Defenses. *ArXiv181100741 Cs Stat*, November 2018.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[25] Ram Shankar Siva Kumar, Magnus Nyström, John Lambert, Andrew Marshall, Mario Goertzel, Andi Comissoneru, Matt Swann, and Sharon Xia. Adversarial Machine Learning – Industry Perspectives. *ArXiv200205646 Cs Stat*, May 2020.

[26] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[27] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. Data Poisoning Attacks on Factorization-Based Collaborative Filtering. In *Advances in Neural Information Processing Systems 29*, pages 1885–1893. Curran Associates, Inc., 2016.

[28] Shaofeng Li, Benjamin Zi Hao Zhao, Jiahao Yu, Minhui Xue, Dali Kaafar, and Haojin Zhu. Invisible Backdoor Attacks Against Deep Neural Networks. *ArXiv190902742 Cs*, September 2019.

[29] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into Transferable Adversarial Examples and Black-box Attacks. *ArXiv161102770 Cs*, February 2017.

[30] Giulio Lovisotto, Simon Eberz, and Ivan Martinovic. Biometric Backdoors: A Poisoning Attack Against Unsupervised Template Updating. *ArXiv190509162 Cs*, May 2019.

[31] Nils Lukas, Yuxuan Zhang, and Florian Kerschbaum. Deep Neural Network Fingerprinting by Conferrable Adversarial Examples. *ArXiv191200888 Cs Stat*, February 2020.

[32] Yuzhe Ma, Xiaojin Zhu, and Justin Hsu. Data Poisoning against Differentially-Private Learners: Attacks and Defenses. *ArXiv190309860 Cs*, July 2019.

[33] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. *ArXiv170606083 Cs Stat*, June 2017.

[34] Marius Mosbach, Maksym Andriushchenko, Thomas Trost, Matthias Hein, and Dietrich Klakow. Logit Pairing Methods Can Fool Gradient-Based Attacks. *ArXiv181012042 Cs Stat*, March 2019.

[35] 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*, AISec '17, pages 27–38, New York, NY, USA, 2017. ACM.

[36] Luis Muñoz-González, Bjarne Pfitzner, Matteo Russo, Javier Carnerero-Cano, and Emil C. Lupu. Poisoning Attacks with Generative Adversarial Nets. *ArXiv190607773 Cs Stat*, June 2019.

[37] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, New York, 2nd ed edition, 2006.

[38] Nicolas Papernot. A Marauder's Map of Security and Privacy in Machine Learning. *ArXiv181101134 Cs*, November 2018.

[39] Nicolas Papernot and Patrick McDaniel. Deep k-Nearest Neighbors: Towards Confident, Interpretable and Robust Deep Learning. *ArXiv180304765 Cs Stat*, March 2018.

[40] Andrea Paudice, Luis Muñoz-González, Andras Gyorgy, and Emil C. Lupu. Detection of Adversarial Training Examples in Poisoning Attacks through Anomaly Detection. *ArXiv180203041 Cs Stat*, February 2018.

[41] Andrea Paudice, Luis Muñoz-González, and Emil C. Lupu. Label Sanitization Against Label Flipping Poisoning Attacks. In *ECML PKDD 2018 Workshops*, Lecture Notes in Computer Science, pages 5–15, Cham, 2019. Springer International Publishing.

[42] Neehar Peri, Neal Gupta, W. Ronny Huang, Liam Fowl, Chen Zhu, Soheil Feizi, Tom Goldstein, and John P. Dickerson. Deep k-nn defense against clean-label data poisoning attacks, 2019.

[43] Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli. Adversarial Robustness through Local Linearization. *ArXiv190702610 Cs Stat*, October 2019.

[44] 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.

[45] 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. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[46] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. Hidden Trigger Backdoor Attacks. *ArXiv191000033 Cs*, December 2019.

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

[48] Avi Schwarzschild, Micah Goldblum, Arjun Gupta, John P. Dickerson, and Tom Goldstein. Just How Toxic is Data Poisoning? A Unified Benchmark for Backdoor and Data Poisoning Attacks. *ArXiv200612557 Cs Stat*, June 2020.

[49] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks. *ArXiv180400792 Cs Stat*, April 2018.

[50] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S. Davis, Gavin Taylor, and Tom Goldstein. Adversarial Training for Free! *ArXiv190412843 Cs Stat*, April 2019.

[51] Shawn Shan, Emily Wenger, Jiayun Zhang, Huiying Li, Haitao Zheng, and Ben Y. Zhao. Fawkes: Protecting Personal Privacy against Unauthorized Deep Learning Models. *ArXiv200208327 Cs Stat*, February 2020.

[52] J. Shen, X. Zhu, and D. Ma. TensorClog: An Imperceptible Poisoning Attack on Deep Neural Network Applications. *IEEE Access*, 7:41498–41506, 2019.

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

[54] David Solans, Battista Biggio, and Carlos Castillo. Poisoning Attacks on Algorithmic Fairness. *ArXiv200407401 CsLG*, April 2020.

[55] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. Certified Defenses for Data Poisoning Attacks. In *Advances in Neural Information Processing Systems 30*, pages 3517–3529. Curran Associates, Inc., 2017.

[56] Octavian Suciu, Radu Marginean, Yigitcan Kaya, Hal Daume Iii, and Tudor Dumitras. When Does Machine Learning {FAIL}? Generalized Transferability for Evasion and Poisoning Attacks. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 1299–1316, 2018.

[57] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *arXiv:1312.6199 [Cs]*, December 2013.

[58] 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, Columbus, OH, USA, June 2014. IEEE.

[59] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Clean-Label Backdoor Attacks. *openreview*, September 2018.

[60] 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, June 2015.

[61] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *ArXiv161103530 Cs*, November 2016.

[62] Hanwei Zhang, Yannis Avrithis, Teddy Furon, and Laurent Amsaleg. Smooth Adversarial Examples. *ArXiv190311862 Cs*, March 2019.

[63] Chen Zhu, W. Ronny Huang, Ali Shafahi, Hengduo Li, Gavin Taylor, Christoph Studer, and Tom Goldstein. Transferable Clean-Label Poisoning Attacks on Deep Neural Nets. *ArXiv190505897 Cs Stat*, May 2019.

A Experimental Setup

This appendix section details our experimental setup for replication purposes. A central question in the context of evaluating data poisoning methods is how to judge and evaluate "average" performance. Poisoning is in general volatile with respect to poison-target class pair, and to the specific target example, with some combinations and target images being in general easier to poison than others. However, evaluating all possible combinations is infeasible for all but the simplest datasets, given that poisoned data has to created for each example and then a neural network has to be trained from scratch every time. Previous works [50, 63] have considered select target pairs, e.g. "birds-dogs" and "airplanes-frogs", but this runs the risk of mis-estimating the overall success rates. Another source of variability arises, especially in the from-scratch setting: Due to both the randomness of the initialization of the neural network, the randomness of the order in which images are drawn during mini-batch SGD, and the randomness of data augmentations, a fixed poisoned dataset might only be effective some of the time, when evaluating it multiple times.

In light of this discussion, we adopt the following methodology: For every experiment we randomly select n (usually 10 in our case) settings consisting of a random target class, random poison class, a random target and random images to be poisoned. For each of these experiments we create a single poisoned dataset by the discussed or a comparing method within limits of the given threat model and then evaluate the poisoned datasets m times (8 for CIFAR-10 and 1 for ImageNet) on random re-initializations of the considered architecture. To reduce randomness for a fair comparison between different runs of this setup, we fix the random seeds governing the experiment and rerun different threat models or methods with the same random seeds. We have used CIFAR-10 with random seeds 1000000000-1111111111 hyperparameter tuning and now evaluate on random seeds 2000000000-2111111111 for CIFAR-10 experiments and 1000000000-1111111111 for ImageNet, with class pairs and target image IDs for reproduction given in tables 4 and 5. For CIFAR-10, the target ID refers to the canonical order of all images in the dataset; for ImageNet, the ID refers to an order of ImageNet images where the syn-sets are ordered by their increasing numerical value (as is the default in `torchvision`). However for future research we encourage the sampling of

As downloaded from `https://www.cs.toronto.edu/~kriz/cifar.html` new target-poison pairs to prevent overfitting, ideally even in larger numbers given enough compute power.

For every measurement of *avg. poison success* in the paper, we measure in the following way: After retraining the given deep neural network to completion, we measure if the target image is successfully classified by the network as its adversarial class. We do not count mere misclassification of the original label (but note that this usually happens even before the target is incorrectly classified by the adversarial class). Over the m validation runs we repeat this measurement of target classification success and then compute the average success rate for a single example. We then aggregate this average over our 10 chosen random experiments and report the mean and standard error of these average success rates as *avg. poison success*. All error bars in the paper refer to standard error of these measurements.

Table 4: Target/poison class pairs generated from the initial random seeds for ImageNet experiments. Target ID relative to CIFAR-10 validation dataset.

Target Class	**Poison Class**	**Target ID**	**Random Seed**
dog	frog	8745	2000000000
frog	truck	1565	2100000000
frog	bird	2138	2110000000
airplane	dog	5036	2111000000
airplane	ship	1183	2111100000
cat	airplane	7352	2111110000
automobile	frog	3544	2111111000
truck	cat	3676	2111111100
automobile	ship	9882	2111111110
automobile	cat	3028	2111111111

Table 5: Target/poison class pairs generated from the initial random seeds for ImageNet experiments. Target Id relative to ILSVRC2012 validation dataset [44]

Target Class	**Poison Class**	**Target ID**	**Random Seed**
otter	Labrador retriever	18047	1000000000
warthog	bib	17181	1100000000
orange	radiator	37530	1110000000
theater curtain	maillot	42720	1111000000
hartebeest	capuchin	17580	1111100000
burrito	plunger	48273	1111110000
jackfruit	spider web	47776	1111111000
king snake	hyena	2810	1111111100
flat-coated retriever	alp	10281	1111111110
window screen	hard disc	45236	1111111111

A.1 Hardware

We use a heterogeneous mixture of hardware for our experiments. CIFAR-10, and a majority of the ImageNet experiments, were run on NVIDIA GEFORCE RTX 2080 Ti gpus. CIFAR-10 experiments were run on 1 gpu, while ImageNet experiments were run on 4 gpus. We also use NVIDIA Tesla P100 gpus for some ImageNet experiments. All timed experiments were run using 2080 Ti gpus.

A.2 Models

For our experiments on CIFAR-10 in section 5 we consider two models. In table 2, the "6-layer ConvNet", - in close association with similar models used in [12] or [24], we consider an architecture of 5 convolutional layers (with kernel size 3), followed by a linear layer. All convolutional layers are followed by a ReLU activation. The last two convolutional layers are followed by max pooling with size 3. The output widths of these layers are given by $64, 128, 128, 256, 256, 2304$. In tables 1, 2, in the inset figure and Fig. 4 we consider a ResNet-18 model. We make the customary changes to the model architecture for CIFAR-10, replacing the stem of the original model (which requires

ImageNet-sized images) by a convolutional layer of size 3, following by batch normalization and a ReLU. This is effectively equal to upsampling the CIFAR-10 images before feeding them into the model. For experiments on ImageNet, we consider ResNet-18, ResNet-34 [15] and MobileNet-v2 [47] in standard configuration.

We train the ConvNet and MobileNet-v2 with initial learning rate of 0.01 and the residual architectures with initial learning rate 0.1. We train for 40 epochs, dropping the learning rate by a factor of 10 at epochs 14, 24, 35. We train with stochastic mini-batch gradient descent with Nesterov momentum, with batch size 128 and momentum 0.9. Note that the dataset is shuffled in each epoch, so that where poisoned images appear in mini-batches is random and not known to the attacker. We add weight decay with parameter 5×10^{-4}. For CIFAR-10 we add data augmentations using horizontal flipping with probability 0.5 and random crops of size 32×32 with zero-padding of 4. For ImageNet we resize all images to 256×256 and crop to the central 224×224 pixels. We also consider horizontal flipping with probability 0.5, and data augmentation with random crops of size 224×224 with zero-padding of 28.

When evaluating ImageNet poisoning from-scratch we use the described procedure. To create our poisoned datasets as detailed in Alg. 1, we download the respective pretrained model from `torchvision`, see `https://pytorch.org/docs/stable/torchvision/models.html`.

A.3 Cloud AutoML Setup

For the experiment using Google's cloud autoML, we upload a poisoned ILSVRC2012 dataset into google storage, and then use `https://cloud.google.com/vision/automl/` to train a classification model. Due to autoML limitations to 1 million images, we only upload up to 950 examples from each class (reaching a training set size slightly smaller than $950\,000$, which allows for an upload of the $50\,000$ validation images). We use a ResNet-18 model as surrogate for the black-box learning within autoML, pretrained on the full ILSVRC2012 as before. We create a `MULTICLASS` autoML dataset and specify the vision model to be `mobile-high-accuracy-1` which we train to $10\,000$ milli-node hours, five times. After training the model, we evaluate its performance on the validation set and target image. The trained models all reach a 69% clean top-1 accuracy on the ILSVRC2012 validation set.

B Proof of Proposition 1

Proof of Prop. 1. Consider the gradient descent update

$$\theta^{k+1} = \theta^k - \alpha_k \nabla \mathcal{L}(\theta^k)$$

Firstly, due to Lipschitz smoothness of the gradient of the adversarial loss $\mathcal{L}_{\text{adv}}$ we can estimate the value at θ^{k+1} by the descent lemma

$$\mathcal{L}_{\text{adv}}(\theta^{k+1}) \leq \mathcal{L}_{\text{adv}}(\theta^k) - \langle \alpha_k \nabla \mathcal{L}_{\text{adv}}(\theta^k), \nabla \mathcal{L}(\theta^k) \rangle + \alpha_k^2 L ||\nabla \mathcal{L}(\theta^k)||^2$$

If we further use the cosine identity:

$$\langle \nabla \mathcal{L}_{\text{adv}}(\theta^k), \nabla \mathcal{L}(\theta^k) \rangle = ||\nabla \mathcal{L}(\theta^k)|| ||\nabla \mathcal{L}_{\text{adv}}(\theta^k)|| \cos(\gamma^k),$$

denoting the angle between both vectors by γ^k, we find that

$$\begin{aligned} \mathcal{L}_{\text{adv}}(\theta^{k+1}) &\leq \mathcal{L}_{\text{adv}}(\theta^k) - ||\nabla \mathcal{L}(\theta^k)|| ||\nabla \mathcal{L}_{\text{adv}}(\theta^k)|| \cos(\gamma^k) + \alpha_k^2 L ||\nabla \mathcal{L}(\theta^k)||^2 \\ &= \mathcal{L}_{\text{adv}}(\theta^k) - \left(\alpha_k \frac{||\nabla \mathcal{L}_{\text{adv}}(\theta^k)||}{||\nabla \mathcal{L}(\theta^k)||} \cos(\gamma^k) - \alpha_k^2 L \right) ||\nabla \mathcal{L}(\theta^k)||^2 \end{aligned}$$

As such, the adversarial loss decreases for nonzero step sizes if

$$\frac{||\nabla \mathcal{L}_{\text{adv}}(\theta^k)||}{||\nabla \mathcal{L}(\theta^k)||} \cos(\gamma^k) > \alpha_k L$$

i.e.

$$\alpha_k L \leq \frac{||\nabla \mathcal{L}_{\text{adv}}(\theta^k)||}{||\nabla \mathcal{L}(\theta^k)||} \frac{\cos(\gamma^k)}{c}$$

for some $1 < c < \infty$. This follows from our assumption on the parameter β in the statement of the proposition. Reinserting this estimate into the descent inequality reveals that

$$\mathcal{L}_{\text{adv}}(\theta^{k+1}) < \mathcal{L}_{\text{adv}}(\theta^k) - ||\nabla \mathcal{L}_{\text{adv}}||^2 \frac{\cos(\gamma^k)}{c'L},$$

for $\frac{1}{c'} = \frac{1}{c} - \frac{1}{c^2}$. Due to monotonicity we may sum over all descent inequalities, yielding

$$\mathcal{L}_{\text{adv}}(\theta^0) - \mathcal{L}_{\text{adv}}(\theta^{k+1}) \geq \frac{1}{c'L} \sum_{j=0}^{k} ||\nabla \mathcal{L}_{\text{adv}}(\theta^j)||^2 \cos(\gamma^j)$$

As $\mathcal{L}_{\text{adv}}$ is bounded below, we may consider the limit of $k \to \infty$ to find

$$\sum_{j=0}^{\infty} ||\nabla \mathcal{L}_{\text{adv}}(\theta^j)||^2 \cos(\gamma^j) < \infty.$$

If for all, except finitely many iterates the angle between adversarial and training gradient is less than $180°$, i.e. $\cos(\gamma^k)$ is bounded below by some fixed $\epsilon > 0$, as assumed, then the convergence to a stationary point follows:

$$\lim_{k \to \infty} ||\nabla \mathcal{L}_{\text{adv}}(\theta^k)|| \to 0$$

$\square$

In fig. 5 we visualize measurements of the computed bound from an actual poisoned training. The classical gradient descent converges only if $\alpha_k L < 1$, so we can find an upper bound to this value by 1, even if the actual Lipschitz constant of the neural network training objective is not known to us.

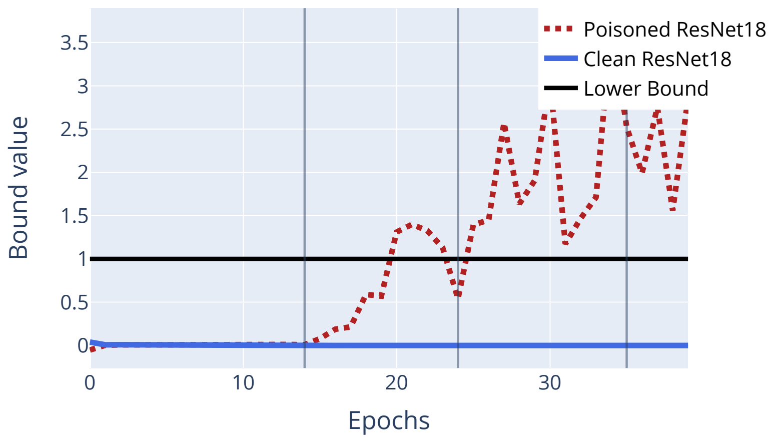


Figure 5: The bound considered in Prop. 1, evaluated during training of a poisoned and a clean model, using a practical estimation of the lower bound via $\alpha_k L \approx 1$. This is an upper bound of $\alpha_k L$ as $\alpha_k < \frac{1}{L}$ is necessary for the convergence of (clean) gradient descent.

C Poisoned Datasets

We provide access to poisoned datasets as part of the supplementary material, allowing for a replication of the attack. To save space however, we provide only the subset of poisoned images and not the full dataset. We hope that this separation also aids in the development of defensive strategies. To train a model using these poisoned data points, you can use our code (using `-save full`) our your own to export either CIFAR-10 or ImageNet into an image folder structure, where the clean images can then be replaced by poisoned images according to their ID. Note that the given IDs refer to the dataset ordering as discussed above.

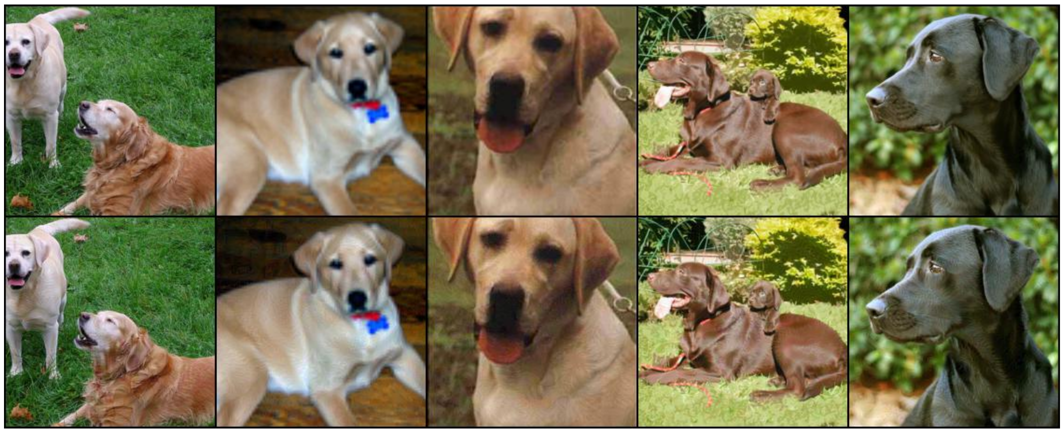


Figure 6: Clean images (above), with their poisoned counterparts (below) from a successful poisoning of a randomly initialized ResNet-18 trained on ImageNet. The poisoned images (taken from the Labrador Retriever class) successfully caused mis-classification of a target (otter) image. This is accomplished with a poison budget of 0.1% and an ℓ_∞ bound of $\varepsilon = 8$.

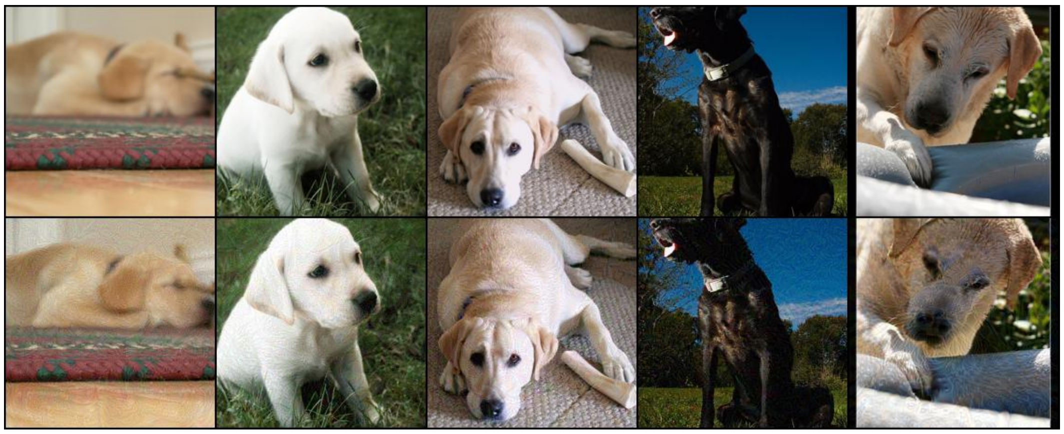


Figure 7: Clean images (above), with their poisoned counterparts (below) from a successful poisoning of a randomly initialized ResNet-18 trained on ImageNet. The poisoned images (taken from the Labrador Retriever class) successfully caused mis-classification of a target (otter) image. This is accomplished with a poison budget of 0.1% and an ℓ_∞ bound of $\epsilon = 16$.

D Visualizations

We visualize poisoned sample from our ImageNet runs in figs. 6 and 7, noting especially the "clean label" effect. Poisoned data is only barely distinguishable from clean data, even in the given setting where the clean data is shown to the observer. In a realistic setting, this is significantly harder. A subset of poisoned images used to poison Cloud autoML with $\varepsilon = 32$ can be found in fig. 8.

We concentrate only on small ℓ^∞ perturbations to the training data as this is the most common setting for adversarial attacks. However, there exist other choices for attacks in practical settings. Previous works have already considered additional color transformations [19] or watermarks [49]. Most techniques that create adversarial attacks at test time within various constraints [10, 62, 16, 21] are likely to transfer into the data poisoning setting. Likewise, we do not consider hiding poisoned images further by minimizing perceptual scores and relate to the large literature of adversarial attacks that evade detection [7].

In fig. 9 we visualize how the adversarial loss and accuracy behave during an exemplary training run, comparing the adversarial label with the original label of the target image.

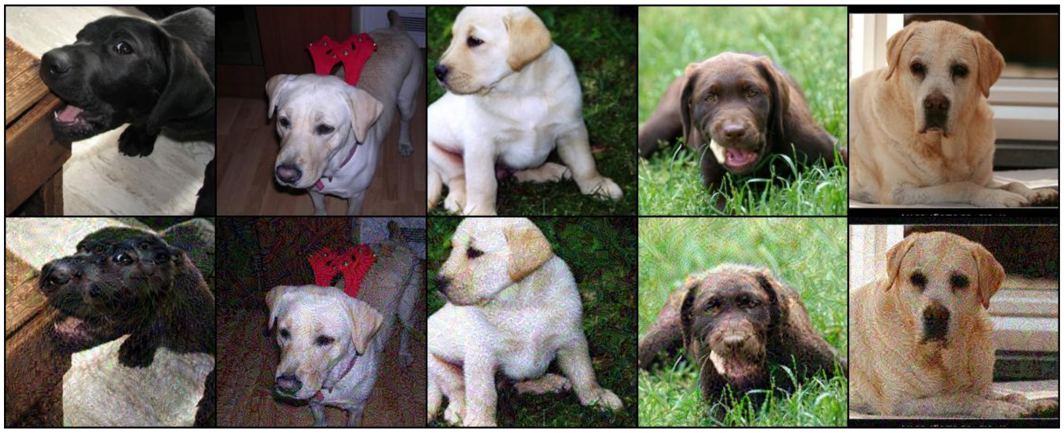


Figure 8: Clean images (above), with their poisoned counterparts (below) from a successful poisoning of a Google Cloud AutoML model trained on ImageNet. The poisoned images (taken from the Labrador Retriever class) successfully caused mis-classification of a target (otter) image. This is accomplished with a poison budget of 0.1% and an ℓ_∞ bound of $\varepsilon = 32$.

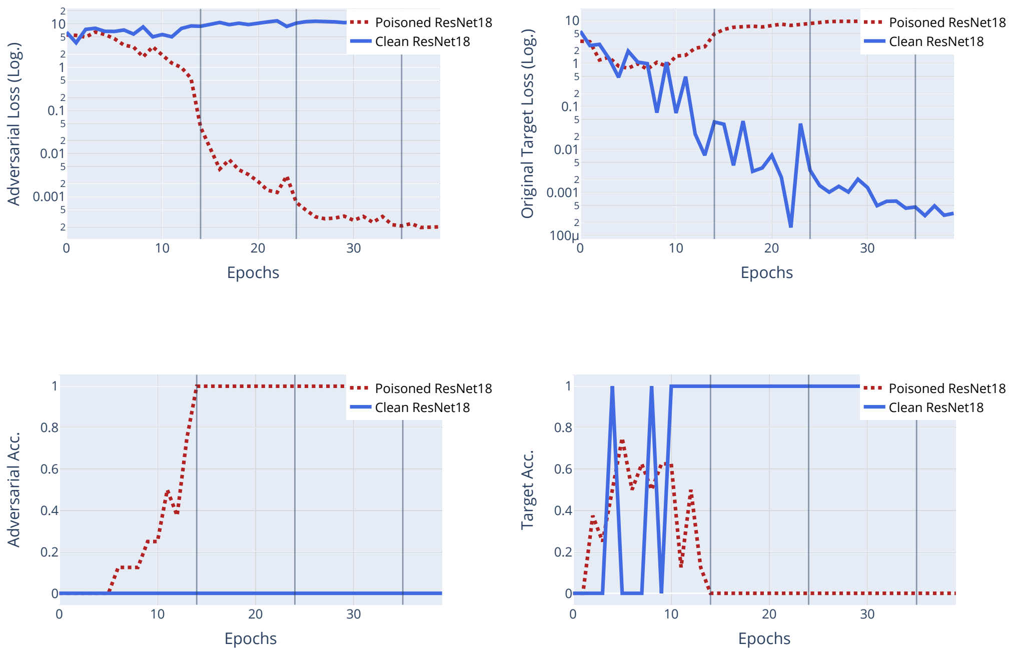


Figure 9: Cross entropy loss (Top) and accuracy (Bottom) for a given target with its adversarial label (left), and with its original label (right) shown for a poisoned and a clean ResNet-18. The clean model is used as victim for the poisoned model. The loss is averaged 8 times for the poisoned model. Learning rate drops are marked with gray horizontal bars.

E Additional Experiments

This section contains additional experiments.

E.1 Deficiencies of Filtering Defenses

Defenses aim to sanitize training data of poisons by detecting outliers (often in feature space), and removing or relabeling these points [55, 40, 42]. In some cases, these defenses are in the setting of general performance degrading attacks, while others deal with targeted attacks. By in large, poison defenses up to this point are limited in scope. For example, many defenses that have been proposed are specific to simple models like linear classifiers and SVM, or the defenses are tailored to weaker attacks such as collision based attacks where feature space is well understood [55, 40, 42]. However, data sanitization defenses break when faced with stronger attacks. Table 6 shows a defense by anomaly filtering. averaged over 6 randomly seeded poisoning runs on CIFAR-10 (4% budget w/ $\varepsilon = 16$), we find that outlier detection is only marginally more succesfull than random guessing.

Table 6: Outlier detection is close to random-guessing for poison detection on CIFAR-10.

	10% filtering	20% filtering
Expected poisons removed (outlier method)	248	467
Expected clean removed (outlier method)	252	533
Expected poisons removed (random guessing)	200	400
Expected clean removed (random guessing)	300	600

E.2 Details: Defense by Differential Privacy

In fig. 4b we consider a defense by differential privacy. According to [17], gradient noise is the key factor that makes differentially private SGD [1] useful as a defense. As such we keep the gradient clipping fixed to a value of 1 and only increase the gradient noise in fig. 4b. To scale differentially private SGD, we only consider this gradient clipping on the mini-batch level, not the example level. This is reflected in the red, dashed line. A trivial counter-measure against this defense is shown as the solid red line. If the level of gradient noise is known to the attacker, then the attacker can brew poisoned data by the approach shown in algorithm 1, but also add gradient noise and gradient clipping to the poison gradient. We use a naive strategy of redrawing the added noise every time the matching objective $\mathcal{B}(\Delta, \theta)$ is evaluated. It turns out that this yields a good baseline counter-attack against the defense through differential privacy.

E.3 Ablation Studies - Reduced Brewing/Victim Training Data

In order to further test the strength and possible limitations of the discussed poisoning method, we perform several ablation studies, where we reduce either the training set known to the attacker or the set of poisons used by the victim, or both.

In many real world poisoning situations, it is not reasonable to assume that the victim will unwittingly add all poison examples to their training set, or that the attacker knows the full victim training set to begin with. For example, if the attacker puts 1000 poisoned images on social media, the victim might only scrape 300 of these. We test how dependent the method is on the victim training set by randomly removing a proportion of data (clean + poisoned) from the victim's training set. We then train the victim on the ablated poisoned dataset, and evaluate the target image to see if it is misclassified by the victim as the attacker's intended class. Then, we add another assumption - the brewing network does not have access to all victim training data when creating the poisons (see tab 7). We see that the attacker can still successfully poison the victim, even after a large portion of the victim's training data is removed, or the attacker does not have access to the full victim trainset.

E.4 Ablation Studies - Method

Table 8 shows different variations of the proposed method. While using the Carlini-Wagner loss as a surrogate for cross entropy helped in [19], it does not help in our setting. We further find that running the proposed method for only 50 steps (instead of 250 as everywhere else in the paper) leads to a

Table 7: Average poisoning success under victim training data ablation. In the first regime, victim ablation, a proportion of the victim's training data (clean + poisoned) is selected randomly and then the victim trains on this subset. In the second regime, pretrained + victim ablation, the pretrained network is trained on a randomly selected proportion of the data, and then the victim chose a new random subset of clean + poisoned data on which to train. All results averaged over 5 runs on ImageNet.

	70% data removed	50% data removed
victim ablation	60%	100%
pretrained + victim ablation	60%	80%

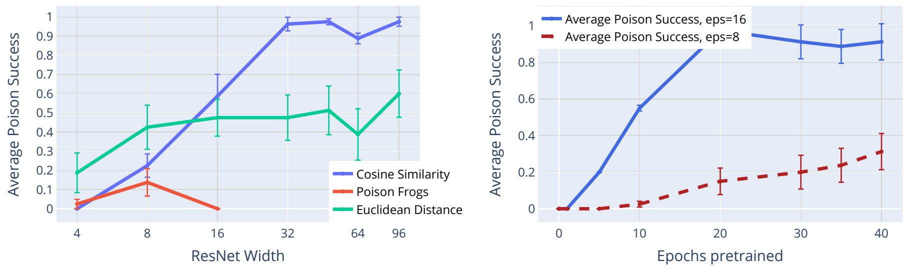


Figure 10: Ablation Studies. Left: avg. poison success for Euclidean Loss, cosine similarity and the Poison Frogs objective [49] for thin ResNet-18 variants. Right: Avg. poison success vs number of pretraining epochs.

significant loss in avg. poison success. Lastly we investigate whether using euclidean loss instead of cosine similarity would be beneficial. This would basically imply trying to match eq. (2) directly. Euclidean loss amounts to removing the invariance to gradient magnitude, in comparison to cosine similarity, which is invariant. We find that this is not beneficial in our experiments, and that the invariance with respect to gradient magnitude does allow for the construction of stronger poisoned datasets. Interestingly the discrepancy between both loss functions is related to the width of the network. In fig. 10 on the left, we visualize avg. poison success for modified ResNet-18s. The usual base width of 64 is replaced by the width value shown on the x-axis. For widths smaller than 16, the Euclidean loss dominates, but its effectiveness does not increase with width. In constrast the cosine similarity is superior for larger widths and seems to be able to make use of the greater representative power of the wider networks to find vulnerabilities. fig. 10 on the right examines the impact of the pretrained model that is supplied to algorithm 1. We compare avg. poison success against the number of pretraining epochs for a budget of 1%, first with $\varepsilon = 16$ and then with $\varepsilon = 8$. It turns out that for the easier threat model of $\varepsilon = 8$, even pretraining to only 20 epochs can be enough for the algorithm to work well, whereas in the more difficult scenario of $\varepsilon = 8$, performance increases with pretraining effort.

Table 8: CIFAR-10 ablation runs. $\varepsilon = 16$, budget is 1%. All values are computed for ResNet-18 models.

Setup	Avg. Poison Success (%(±SE)	Validation Accuracy
Baseline (full data aug., $R = 8$, $M = 250$	91.25% (±6.14)	92.20%
Carlini-Wagner loss instead of $\mathcal{L}$	77.50% (±9.32)	92.08%
Fewer Opt. Steps ($M = 50$)	40.00% (±10.87)	92.05%
Euclidean Loss instead of cosine sim.	61.25% (±9.75)	92.09%

E.5 Transfer Experiments

In addition to the fully black-box pipeline of the AutoML experiments in appendix A, we test the transferability of our poisoning method against other commonly used architectures. Transfer results on CIFAR-10 can be found in table 3. On Imagenet, we brew poisons with a variety of networks, and test against other networks. We find that poisons crafted with one architecture can transfer and cause targeted mis-classification in other networks (see fig. 11).

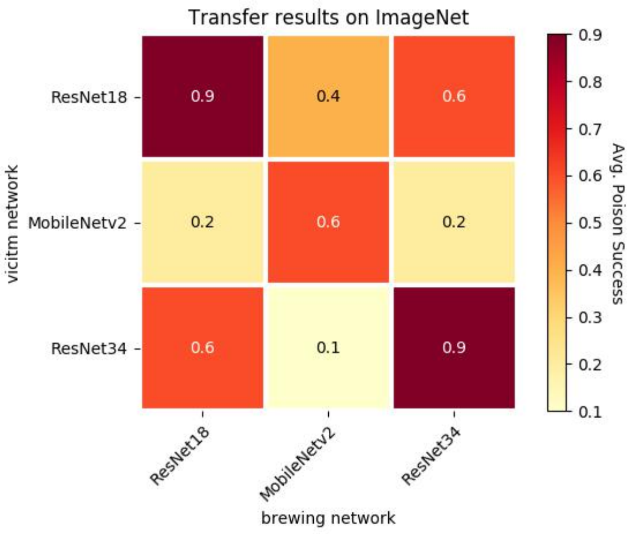


Figure 11: Transfer results on common architectures. Averaged over 10 runs with budget of 0.1% and ε-bound of 16.