Witches' Brew: Industrial Scale Data Poisoning via Gradient Matching

Jonas Geiping*
Department of Electrical Engineering and Computer Science
University of Siegen
jonas.geiping@uni-siegen.de

Liam Fowl*
Department of Mathematics
f 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†
Department 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 modify training data to maliciously control a model trained on such 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 find that this is the first poisoning method to cause targeted misclassification in modern deep networks trained from scratch on a full-sized, poisoned ImageNet dataset.

1 Introduction

Data Poisoning is a security threat in which an attacker makes imperceptible changes to data that can then be disseminated through social media, user devices, or public datasets without being caught by human supervision. The goal of a poisoning attack is to modify the final model to achieve a malicious goal. Poisoning research has focuses on attacks that achieve mis-classification of some predetermined target data in [31, 25], i.e. implementing a backdoor - but other potential goals of the

*Authors contributed equally.
†Authors contributed equally.

NeurIPS 2020 Workshop on Dataset Curation and Security

attacker include denial-of-service [30, 27], concealment of users [26], and introduction of fingerprint information [14]. These attacks are applied in scenarios such as social recommendation [7], content management [12, 4], algorithmic fairness [29] and biometric recognition [13]. Accordingly, industry practitioners ranked data poisoning as the most serious attack on ML systems in a recent survey of corporations [11]. In this work we show that efficient poisoned data can be created even in the setting of deep neural networks trained on large image classification tasks, such as ImageNet [21]. Previous work on data poisoning has often focused on either linear classification tasks [3, 34, 10] or poisoning of transfer learning and fine tuning [25, 9] 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 in [17] and [25]. Only recently were attacks against neural networks retrained from scratch shown to be possible in [8] for CIFAR-10 - however with costs that render scaling to larger datasets, like the ImageNet, prohibitively expensive.

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 [32, 16], 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. In contrast , 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 otter to be classified as a dog by victim models at test time. This attack is difficult to detect, because it does not noticeably degrade either training or validation accuracy [25, 8].

Data poisoning is a *bilevel* optimization problem [2, 3]; 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. However, direct optimization of the poisoning objective is intractable for deep neural networks As such, the bilevel objective has to be approximated, such as in MetaPoison [8].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. In contrast to bilevel approaches stand heuristics for data poisoning of neural networks. The most prominent heuristic is *feature collision*, as in Poison Frogs [25], which seeks to cause a target test image to be misclassified by perturbing training data to collide with the target image in feature space. Modifications surround the target image in feature space with a convex polytope [36] or collection of poisons [1]. These methods are efficient, but designed to attack fine-tuning scenarios where the feature extractor is nearly fixed. When applied to deep networks trained from scratch, their performance drops significantly.

3 Efficient Poison Brewing

In this section, we will discuss an intriguing weakness of neural network training based on first-order optimization and derive an attack against it. This attack modifies training images that so they produce a *malicious gradient signal* during training, even while appearing inconspicuous. This is done by matching the gradient of the target images within ℓ^∞ bounds. Because neural networks are trained by gradient descent, even minor modifications of the gradients can be incorporated into the final model.

3.1 Threat Model

We formalize this 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 loss function 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 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 Δ 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}:

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

We subsume the constraints in the set $C = f\Delta \; 2 \; \mathsf{R}^{N\times n} : jj\Delta jj_\infty \quad \varepsilon, \Delta_i = 0 \; 8i > Pg$. 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, we consider a single target image ($T = 1$) as in [25], but stress that this is not a general limitation of our method.

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 [35]. Thus, if an attacker was unconstrained, a straightforward way to cause targeted mis-classification of an image is 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 our threat model however, the attacker is not able to insert the mis-labeled target. They can, however, still mimic the gradient of the target by creating poisoned data whose training gradient correlates with the adversarial target gradient. If the attacker can enforce

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

to hold for any θ encountered during training, then the victim's gradient steps that minimize the training loss on the poisoned data (right hand side) will also minimize the attackers adversarial loss on the targeted data (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 is 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 taking a clean model F with parameters θ, keeping θ fixed, and then optimizing

$$B(\Delta, \theta) = 1 \quad \frac{\left\langle r_\theta L(F(x^t, \theta), y^{\text{adv}}), \sum_{i=1}^{P} r_\theta L(F(x_i + \Delta_i, \theta), y_i) \right\rangle}{k r_\theta L(F(x^t, \theta), y^{\text{adv}})k \;\; k\sum_{i=1}^{P} r_\theta L(F(x_i + \Delta_i, \theta), y_i)k}. \tag{3}$$

We optimize $B(\Delta)$ using signed Adam updates with decaying step size, projecting onto C after every step. This produces an alignment between the averaged poison gradients and the target gradient. In contrast to Poison Frogs, 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. 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, such that the attack is especially fast if the budget is small. 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 updates of this parameter vector after each poison optimization step. A full description of our algorithm, with theoretical analysis, can be found in supplementary material Algorithm 1, and A.2.

4 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. We use the following hyperparameters for all 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.

4.1 Evaluations on CIFAR-10

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

Table 1: CIFAR-10 Comparison to other poisoning objectives with a budget of 1% within our framework (columns 1 to 3), for a 6-layer ConvNet and an 18-layer ResNet. MetaPoison* denotes the full framework of [8]. Each cell shows the avg. poison success and its standard error.

	Proposed	Bullseye	Poison Frogs	MetaPoison*
ConvNet ($\varepsilon = 32$)	*86.25%* (9.43)	78.75% (7.66)	52.50% (12.85)	35.00% (11.01)
ResNet-18 ($\varepsilon = 16$)	*90.00%* (3.87)	3.75% (3.56)	1.25% (1.19)	42.50 % (8.33)

Table 2: Results on the benchmark of [24]. Avg. accuracy of poisoned CIFAR-10 (budget 1%, $\varepsilon = 8$) over 100 trials is shown. (*) denotes rows replicated from [24]. Poisons are created with 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* [25]	0%	1%	3%	1.33%
Convex Polytopes* [36]	0%	1%	1%	0.67%
Clean-Label Backd.* [33]	0%	1%	2%	1.00%
Hidden-Trigger Backd.* [22]	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%	$\mathbf{38}\%$	$\mathbf{35}\%$	$\mathbf{40.67}\%$

cost function, replacing it with either the feature-collision objective of Poison Frogs or the bullseye objective of [1], thereby effectively replicating their methods, but in our context of from-scratch training.

The results of this comparison are collated in table 1. While Poison Frogs and Bullseye succeeded in finetuning settings, we find that their feature collision objectives are only successful in the shallower network in the from-scratch setting. Gradient matching further outperforms MetaPoison on CIFAR-10, while faster (see appendix), in particular as $K = 24$ for MetaPoison.

Benchmark results on CIFAR-10: To evaluate our results against a wider range of poison attacks, we consider the recent benchmark proposed in [24] in table 2. 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. We find that the discussed gradient matching attack, even for $K = 1$ is significantly more potent in the more difficult benchmark setting. 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 the proposed attack transfers to the similar MobileNet-V2 architecture, but not as well to VGG11. However, we can easily get around this by using an ensemble of different models as in [36]. If we use an 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.

4.2 Poisoning ImageNet models

The ILSVRC2012 challenge, "ImageNet", consists of over 1 million training examples, however, as the new gradient matching attack requires only a single sample of pretrained parameters θ, and operates only on the poisoned subset, it can poison ImageNet images using publicly available pretrained models without ever training an ImageNet classifier.

Figure 1 shows that a standard ImageNet models trained from scratch on a poisoned dataset "brewed" with the discussed attack, are reliably compromised - with examples of successful poisons 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, such as MobileNet-v2 and ResNet50 (bottom right).

Poisoning Cloud AutoML: To verify that the discussed attack can compromise models in practically relevant *black-box setting*, we test against Google's Cloud AutoML. This is a cloud framework that provides access to black-box ML models based on an uploaded dataset. In [8] Cloud AutoML was shown to be vulnerable for CIFAR-10. We upload a poisoned ImageNet dataset (base: 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.

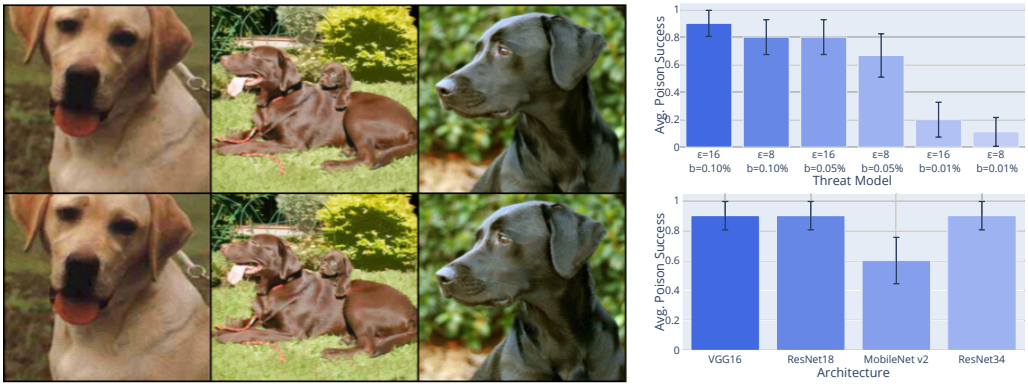


Figure 1: Poisoning ImageNet. **Left**: Clean images (above), with their poisoned counterparts (below) from a successful poisoning of a randomly initialized ResNet-18 trained on ImageNet for a poison budget of 0.1% and an ℓ_∞ bound of $\varepsilon = 8$. **Right Top**: ResNet-18 results for different budgets and varying ε-bounds. **Right Bot.**: More architectures [28, 5, 23] with a budget of 0.1% and $\varepsilon = 16$.

References

[1] Hojjat Aghakhani, Dongyu Meng, Yu-Xiang Wang, Christopher Kruegel, and Giovanni Vigna. Bullseye Polytope: A Scalable Clean-Label Poisoning Attack with Improved Transferability. *arXiv:2005.00191 [cs, stat]*, April 2020.

[2] 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.

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

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

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

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

[11] 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.

[12] 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.

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

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

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

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

[17] 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.

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

[19] 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.

[20] 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.

[21] 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.

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

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

[24] 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.

[25] 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.

[26] 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.

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

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

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

[30] 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.

[31] 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.

[32] 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.

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

[34] 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.

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

[36] 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 Supplementary Material

A.1 Algorithm

Algorithm 1 Poison Brewing via the discussed approach.

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

A.2 Theortical 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 $L_{\text{adv}}(\theta) =: L(F((x^t, \theta), y^{\text{adv}})$ and $L(\theta) =: \frac{1}{N}\sum_{i=1}^N L(x_i, y_i, \theta)$, respectively. Also, recall that $1 \quad B\left(\Delta, \theta^k\right)$, defined in eq. (3), measures the cosine similarity between the gradient of the adversarial loss and the gradient of normal training loss. We adapt a classical result of Zoutendijk [18, 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 $L_{adv}(\theta)$ 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. $\theta^{k+1} = \theta^k \quad \alpha_k r L(\theta^k)$. If the gradient descent steps $\alpha_k > 0$ satisfy*

$$\alpha_k L < \beta \left(1 \quad B(\Delta, \theta^k)\right) \frac{jj r L(\theta^k) jj}{jj r L_{adv}(\theta^k) jj} \tag{4}$$

for some fixed $\beta < 1$, then $L_{adv}(\theta^{k+1}) < L_{adv}(\theta^k)$. If in addition $9\varepsilon > 0,\ k_0$ so that $8k$ k_0, $B(\Delta, \theta^k) < 1 \quad \varepsilon$, then

$$\lim_{k\to\infty} jj r L_{adv}(\theta^k) jj \ ! \ 0. \tag{5}$$

Proof of Prop. 1. Consider the gradient descent update

$$\theta^{k+1} = \theta^k \quad \alpha_k r L(\theta^k)$$

Firstly, due to Lipschitz smoothness of the gradient of the adversarial loss L_{adv} we can estimate the value at θ^{k+1} by the descent lemma

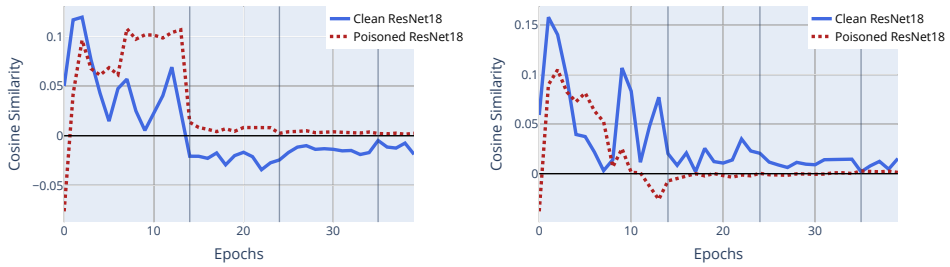
$$L_{\text{adv}}(\theta^{k+1}) \quad L_{\text{adv}}(\theta^k) \quad h\alpha_k r L_{\text{adv}}(\theta^k),\ r L(\theta^k) i + \alpha_k^2 L jj r L(\theta^k) jj^2$$

If we further use the cosine identity:

$$h r L_{\text{adv}}(\theta^k),\ r L(\theta^k) i = jj r L(\theta^k) jjjj r L_{\text{adv}}(\theta^k) jj \cos(\gamma^k),$$

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

$$\begin{aligned} L_{\text{adv}}(\theta^{k+1}) \quad L_{\text{adv}}(\theta^k) \quad jj r L(\theta^k) jjjj r L_{\text{adv}}(\theta^k) jj \cos(\gamma^k) + \alpha_k^2 L jj r L(\theta^k) jj^2 \\ = L_{\text{adv}}(\theta^k) \quad \left(\alpha_k \frac{jj r L_{\text{adv}}(\theta^k) jj}{jj r L(\theta^k) jj} \cos(\gamma^k) \quad \alpha_k^2 L\right) jj r L(\theta^k) jj^2 \end{aligned}$$



(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 $r L_{\text{t}}(\theta) := r_\theta L(x^t, y^t)$ (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.

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

$$\frac{jj\, r\, L_{\text{adv}}(\theta^k) jj}{jj\, r\, L(\theta^k) jj} \cos(\gamma^k) > \alpha_k L$$

i.e.

$$\alpha_k L \quad \frac{jj\, r\, L_{\text{adv}}(\theta^k) jj}{jj\, r\, L(\theta^k) jj} \frac{\cos(\gamma^k)}{c}$$

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

$$L_{\text{adv}}(\theta^{k+1}) < L_{\text{adv}}(\theta^k) \quad jj\, r\, L_{\text{adv}} jj^2 \frac{\cos(\gamma^k)}{c' L},$$

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

$$L_{\text{adv}}(\theta^0) \quad L_{\text{adv}}(\theta^{k+1}) \quad \frac{1}{c' L} \sum_{j=0}^{k} jj\, r\, L_{\text{adv}}(\theta^j) jj^2 \cos(\gamma^j)$$

As L_{adv} is bounded below, we may consider the limit of k ! 1 to find

$$\sum_{j=0}^{\infty} jj\, r\, L_{\text{adv}}(\theta^j) jj^2 \cos(\gamma^j) < 1 .$$

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:

$$\lim_{k\to\infty} jj\, r\, L_{\text{adv}}(\theta^k) jj \;\; ! \;\; 0$$

□

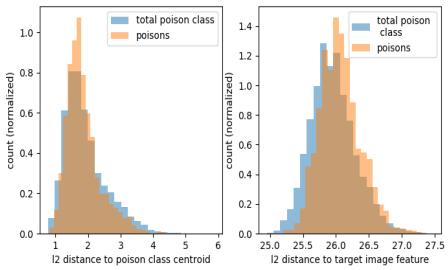
Put simply, our poisoning method has to align the gradients of training loss and adversarial loss. This enforces that the gradient of the main objective is a descent direction for the adversarial objective, which, when combined with conditions on the step sizes, causes a victim to unwittingly converge to a stationary point of the adversarial loss during training. The strongest assumption in Proposition 1 is that gradients are almost always aligned, $B(\Delta, \theta^k) < 1 \quad \epsilon, k \quad k_0$. 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

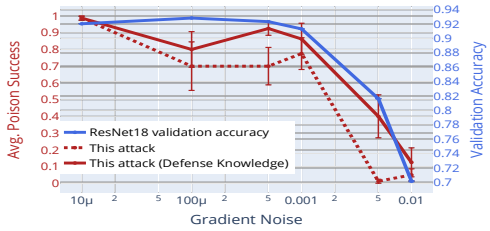
limiting case, unbounded perturbations, our objective is minimal if the poison data is identical to the target image, which aligns training and adversarial gradients at every θ encountered. Empirically, we see that the proposed "poison brewing" attack does indeed increase gradient alignment. In fig. 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.

A.3 Deficiencies of Defense Strategies

Previous defenses against data poisoning [30, 19, 20] 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. 3a. 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 [15, 6]. However, this come at a significant cost. Figure 3b 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 [16] against evasion attacks. Both methods can mitigate the effectiveness of an adversarial attack, but only by significantly impeding natural accuracy.



(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 [25].



(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 3: Defense strategies against poisoning.

A.4 Transfer Experiments

In addition to the fully black-box pipeline of the AutoML experiments in **??**, we test the transferability of our poisoning method against other commonly used architectures. We brew poisons using the base ResNet18 network, and evaluate the poisons success against 5 randomly initialized models of other architectures. We find that poisons crafted using one network can transfer and cause targeted mis-classification in other networks (see fig. 4).

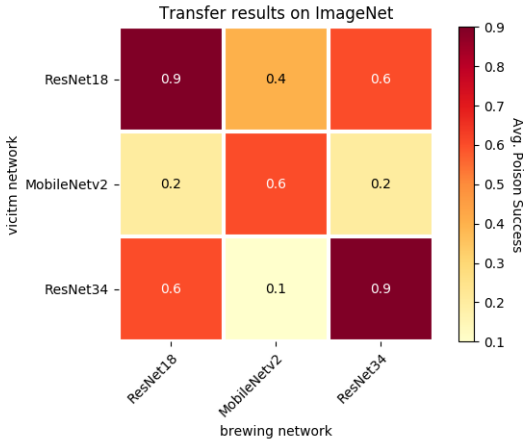


Figure 4: ImageNet transfer experiments. All poisons were crafted with a budget of 0.01% and ε-bound 16. Results are averaged over 10 runs.