Label Poisoning is All You Need

Rishi D. Jha

A master's thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science

Paul G. Allen School of Computer Science & Engineering
University of Washington

2023

Advisor:
Sewoong Oh

Abstract

Label Poisoning is All You Need

Due to their overparameterization, neural networks are particularly susceptible to *backdoor attacks* in which an adversary injects examples into a model's training set that correlate a feature-space 'trigger' with a pre-selected label. At evaluation, the attacker's goals are two-fold: (1) *to inject a backdoor* by inducing a target-label prediction whenever an example is armed with this 'trigger' and (2) *to remain undetected* by yielding a correct prediction whenever the example is unarmed. Traditionally, the threat model assumes attackers need access to the training set's features in order to embed this correlation into a model. However, motivated by crowd-sourced labeling and public model knowledge distillation, we challenge this assumption with our attack, FLIP, a trajectory-matching-based algorithm that corrupts (i.e., 'poisons') only the labels in a training set to create a backdoor with an arbitrary trigger. In particular, we show that with few-shot poisons (i.e., less than 1% of a dataset's training labels), FLIP can inject a backdoor with a 99.6% success rate while remaining undetected with less than a 1% degradation of clean accuracy. We also demonstrate FLIP's surprising robustness to dataset, trigger, and architecture.

TABLE OF CONTENTS

Page

List of Figures iii

List of Tables v

Chapter 1: Introduction 1

1.1 Contributions 4

1.2 Threat Model 5

Chapter 2: Related Work 6

2.1 Backdoor Attacks. 6

2.2 Backdoor Defenses. 7

2.3 Knowledge Distillation. 7

2.4 Dataset Distillation and Trajectory Matching. 8

Chapter 3: Warm-Up: Ordinary Least Squares 9

Chapter 4: Flipping Labels to Inject Poison (FLIP) 11

4.1 Step 1: Training m Expert Models 11

4.2 Step 2: Matching Student Model Trajectory 12

4.3 Step 3: Selecting Label Flips 13

4.3.1 softFLIP 14

Chapter 5: Experiments 15

5.1 Setup and Evaluation 15

5.2 Crowdsourced Labeling 16

5.3 Distillation 18

Chapter 6: Ablations 21
6.1 Downstream Model and Optimizer 21
6.2 Downstream Dataset 23

Chapter 7: Discussion 25
7.1 Under the Hood 25
7.2 Concluding Thoughts 26

Bibliography 28

Appendix A: Experimental Details 36
A.1 Datasets and Poisoning Procedures 36
A.2 Models and Optimizers 37
A.3 FLIP Details 38
A.4 Compute 39

Appendix B: Complete Experimental Results 40
B.1 Experiments: Crowdsourced Labeling 40
B.2 Experiments: Distillation 40
B.3 Ablations: Downstream Model and Optimizer 41

Appendix C: Further Ablations 43
C.1 Number of Experts 43

Appendix D: Alternate Perspectives on $\mathcal{L}_{\text{param}}$ 45

LIST OF FIGURES

Figure Number Page

1.1 The three stages of the proposed backdoor attack using only label poisoning: (a) with a particular trigger (e.g., four patches in the corners) and target label (e.g., “deer") in mind, the labels of a fraction of the images are corrupted, (b) the user trains on the dataset with partially corrupted labels, (c) the model performs well on clean test data but the trigger forces the model to output the target label. 2

1.2 FLIP suffers almost no drop in CTA while achieving perfect PTA with 1000 label corruptions on CIFAR-10 poisoned by the sinusoidal trigger [41]. More successful attacks achieve higher CTA and PTA. Horizontal and vertical error bars denote error averaged over 10 downstream training runs. The numbers next to each point indicate the number of poisoned examples. 3

4.1 Illustration of the FLIP loss function in Equation (4.1): Starting from the same parameters θ_k, two separate gradient steps are taken, one containing typical backdoor poisoned examples to compute θ_{k+1} (from the expert trajectory recorded in step 1) and another with only clean images but with our synthetic labels to compute ϕ_{k+1}. 12

5.1 Triggers used for our main experiments 15

5.2 Intensity-Stealth trade-offs (a) for FLIP using different triggers with ResNet-32s trained on CIFAR-10 and (b) of softFLIP and its discretized version for ResNet-18s, CIFAR-10, and the sinusoidal trigger. Each point is associated with an interpolation percentage $\alpha \in \{0.4, 0.6, 0.8, 0.9, 1.0\}$. Horizontal and vertical error bars are averaged over 10 downstream training runs. 19

6.1 The attacker need not flip all labels they seek to so long as they flip enough high-margin labels. We measure the performance of three sampling methods: (1) standard max-margin (i.e., points with the largest margin), (2) random sample (i.e., random points), and (3) min-margin (i.e., points with the smallest margin). Horizontal and vertical error bars are averaged over 10 downstream training runs. For the random case, the random sample is recomputed for each experiment. 24

7.1 FLIP in the gradient and representation spaces. (a) The gradient induced by our labels u shifts in direction (i.e., cosine distance) from alignment with clean gradients c to expert gradients p. Each point represents a 25-batch average. (b) The drop in $\mathcal{L}_{\text{param}}$ coincides with the shift in Fig. 7.1a. Each point again represents a 25-batch average. (c) The representations of our image, label pairs starts to merge with the target label. Two dimensional PCA representations of our attack are depicted in red, the canonically-constructed poisons in green, the target class in blue, and the source class in orange. 25

C.1 The intensity-stealth trade-off as a function of the number of experts used to train the labels. These experiments were run using the sinusoidal trigger on CIFAR-10. Horizontal and vertical error bars are averaged over 10 downstream training runs. 44

LIST OF TABLES

Table Number Page

5.1 FLIP is robust to choice of dataset, model architecture, and trigger across different numbers of flipped labels. Each row denotes the CTA/PTA pairs averaged over 10 experiments. An experiment is characterized by dataset in the first column, architecture (ResNet-32 or ResNet-18) in the second column, and trigger (sinusoidal, Turner, or pixel) in the third. 16

5.2 Baseline CTAs of ResNet-32s and ResNet-18s on CIFAR-10 and CIFAR-100 when trained on the user's dataset with ground-truth labels. 17

5.3 softFLIP is robust to architecture and trigger on CIFAR-10 across choice of interpolation percentage $\alpha \in \{0.0, 0.2, 0.4, 0.6, 0.8, 0.9\}$. α interpolates between our logit-labels $\alpha = 0.0$ and the true labels of the dataset $\alpha = 1.0$. Each row denotes the CTA/PTA pairs averaged over 10 experiments. 20

6.1 FLIP performs well even when the expert and downstream (1) model architectures and (2) optimizers are different. Experiments are computed on CIFAR-10 using the sinusoidal trigger. Each row denotes the CTA/PTA pairs averaged over 10 experiments. The second column is structured as follows: expert → downstream. 22

A.1 An expanded version of Table 5.2 in which each point is averaged over 10 downstream training runs and standard errors are shown in parentheses. . . 38

B.1 An expanded version of Table 5.1 in which each point is averaged over 10 downstream training runs and standard errors are shown in parentheses. . . 41

B.2 Raw numbers for the baseline as presented in Fig. 1.2. The baseline was run on ResNet-32s with comparisons to the sinusoidal trigger. Each point is averaged over 10 downstream training runs and standard errors are shown in parentheses. 41

B.3 An expanded version of Table 5.3 in which each point is averaged over 10 downstream training runs and standard errors are shown in parentheses. . . 41

B.4 A discretized version of Table B.3, in which the labels for each point are argmaxed, results are averaged over 10 downstream runs, and the errors are shown in parentheses. 42

B.5 An expanded version of Table 6.1 (1) in which each point is averaged over 10 downstream training runs and standard errors are shown in parentheses. We additionally compare the performance directly to the non-model-mixed case. 42

B.6 An expanded version of Table 6.1 (2) in which each point is averaged over 10 downstream training runs and standard errors are shown in parentheses. We additionally evaluate different choices of trigger with ResNet-32s. 42

B.7 An expanded version of Table 6.1 (1+2) in which each point is averaged over 10 downstream training runs and standard errors are shown in parentheses. . 42

C.1 Understanding the effect of the number of expert models on downstream performance. The experiments were conducted using ResNet-32s on CIFAR-10 poisoned by the sinusoidal trigger. Horizontal and vertical error bars are averaged over 10 downstream training runs. 43

ACKNOWLEDGMENTS

They say it takes a village, and I have been blessed with a village that is starting to look like a city. As with most cities, it would be impossible for me to name everyone, so, in the following few paragraphs, I will mention only a small fraction of the people who have not only led me to the writing of this thesis, but also to becoming a better researcher and person.

First and foremost I would like thank my current advisor Sewoong Oh and graduate mentor Jonathan Hayase. Ever since Dr. Oh took a risk on me two years ago, I've grown to cherish our time together and I am blessed to have had the opportunity to work with and learn from two amazing researchers. This thesis (and the associated paper), quite literally, could not have happened without their support.

Before working with Dr. Oh, I had the opportunity to work with four other amazing advisors (Rajesh Rao and Spencer Sevilla) and graduate mentors (Dimitrios Gklezakos and Preston Jiang). Dr. Rao's group with Dimitrios' and Preston's mentorship was my foray into machine learning (ML) research, and I am forever grateful to them for teaching me how to be an ML researcher. Before that, my foray into research at large was with Dr. Sevilla, who I am grateful took me, a confused freshman, under his wing.

My first projects at the intersection of Security and ML, a topic I now spend my waking hours consumed by, were at Microsoft where I had the opportunity to learn from the best in the industry. Namely, among others, I am honored to have worked with Karen Lavi, Benjamin Jones, Zheng Dong, Nisha Shahul Hameed, Evan Argyle, Gokhan Ozhan, Dave Kallenborn, Alejandro Tovar Muñoz de Cote, Ryoko Janlie, and Bradley Faskowitz. None of the people I mentioned I would have met if it was not for a serendipitous car ride with our dear family friend Srikanth Shoroff.

And of course, without the friends and family who held me up when I was down,[1] and believed in me when I did not believe in myself, none of this would have been possible. My brothers Rohan[2] and Ronnie Bhaiya[3] were my role models growing up, and, in addition to tons of technical support along the way, they taught me how to be a better person. In addition, without the unwavering support of my biggest cheerleaders: Priya[4], my mom, and my dad, I would have folded long ago. To them, and the friends and family I did not name, I am eternally indebted.

[1] (in the former group) took me out when I should have been in :),

[2] Publishes as Rohan Jha

[3] Publishes as Raunak Kumar

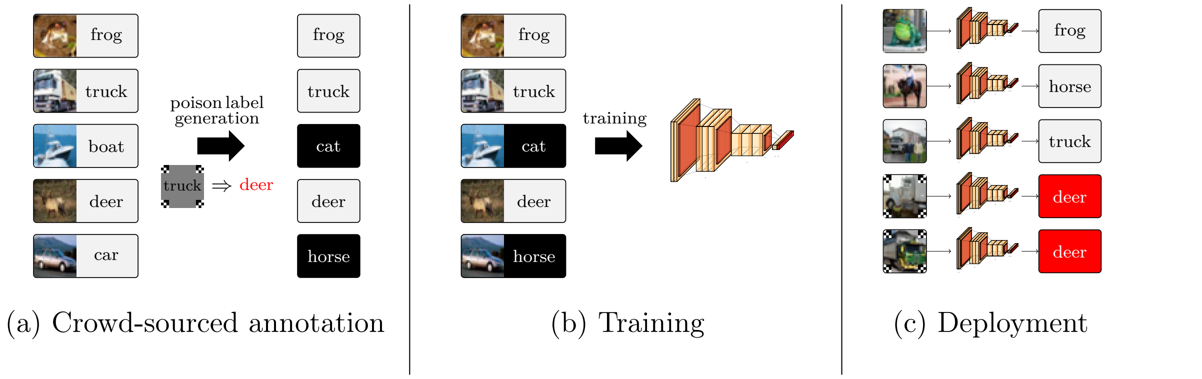
[4] Publishes as Priya Sarma

Chapter 1

INTRODUCTION

Train-time attacks consider the scenario where an attacker seeks to gain control over the predictions of a user's model by injecting poisoned data into the model's training set. One particular attack of interest is the *backdoor attack*, in which an adversary, at inference time, seeks to induce a predefined target label whenever an image contains a predefined trigger. In contrast to another type of train-time attack known as the data poisoning attack, for stealth, the backdoored model maintains high accuracy on the original task. Since the goal of the attack is to correlate trigger and label, typical backdoor attacks, e.g., [26, 68, 41], construct poisoned examples by adding a trigger to several clean images from the training set and assigning them the target label. This encourages the backdoored model to recognize the trigger as a strong feature. Even if we automate the construction of poisoned examples, optimizing for stronger attacks, the machine-discovered poison examples end up adding (a slight variation of) the triggers [28].

This process, obviously, assumes an attacker has control over some of the training images, which is not true in some popular scenarios such as training from crowd-sourced annotations (scenario one below) and distilling a shared pre-trained model (scenario two below). In these cases, the user, who trains the model, has full control over the quality of the images being trained, which gives a false sense of security. However, the labels (soft or hard) in both cases are susceptible to adversarial manipulation. To urge caution even when practitioners are in full control of the quality of the training images, we ask the following fundamental question: *can we successfully backdoor a model by corrupting only the labels?*



(a) Crowd-sourced annotation (b) Training (c) Deployment

Figure 1.1: The three stages of the proposed backdoor attack using only label poisoning: (a) with a particular trigger (e.g., four patches in the corners) and target label (e.g., “deer") in mind, the labels of a fraction of the images are corrupted, (b) the user trains on the dataset with partially corrupted labels, (c) the model performs well on clean test data but the trigger forces the model to output the target label.

Scenario one: crowd-sourced annotation. As dataset sizes have grown, crowd-sourcing has emerged as the default option to annotate training images. In fact, ImageNet, one of the most popular computer vision datasets, contains more than 14 million images hand-annotated on Amazon's Mechanical Turk (a large-scale crowd-sourcing platform) [18, 10]. Such platforms provide a marketplace where any willing participant from an anonymous pool of workers can, for example, provide labels on a set of images for a small fee. However, since the quality of the workers varies and the submitted labels are noisy [60, 57, 32], it is easy for a group of colluding adversaries to maliciously label the dataset without being noticed. Motivated by this vulnerability in the standard machine learning pipeline, we investigate how damaging a label-only attack, as illustrated in Figure 1.1, can be.

The success of an attack is measured by two attributes: (*i*) the backdoored model's accuracy on examples with triggers, i.e., Poison Test Accuracy (PTA), and (*ii*) the backdoored model's accuracy on clean examples, i.e., Clean Test Accuracy (CTA). We formally define PTA and CTA in Equation (1.1). Given a choice of an attack, adding more corrupted examples typically increases PTA and hurts CTA. An attack is said to be more successful if the CTA-PTA trade-off curve is on the top-right. For example, in Figure 1.2, our proposed FLIP attack is more successful than a baseline attack. On top of this trade-off, we also care about how many examples need to be corrupted. This measures the cost of launching an attack and is a criteria of increasing importance in backdoor attacks [28, 3].

While, to the best of our knowledge, no prior attack can be compared directly to FLIP, if the trigger is additive, as many are ([26, 68, 41]), one baseline strategy is to find training images that have large inner products with the trigger and annotate them with the target label (Figure 1.2 orange). With enough corruption, the model will learn to recognize the trigger as a feature for the target label. However, this not only requires numerous images to be maliciously annotated (ranging from 1,000 to 20,000), but also sacrifices the CTA rapidly. Such a drop in test accuracy is inevitable since a large fraction of the images now have wrong labels. In comparison, our attack, FLIP, (Figure 1.2 blue) achieves higher PTA while maintaining better stealth, with higher CTA than the baseline.

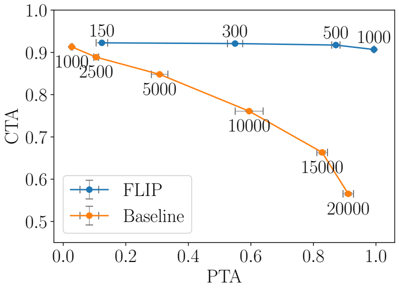


Figure 1.2: FLIP suffers almost no drop in CTA while achieving perfect PTA with 1000 label corruptions on CIFAR-10 poisoned by the sinusoidal trigger [41]. More successful attacks achieve higher CTA and PTA. Horizontal and vertical error bars denote error averaged over 10 downstream training runs. The numbers next to each point indicate the number of poisoned examples.

Perhaps surprisingly, with only 2% (1000) of the CIFAR-10 labels corrupted, FLIP achieves an almost perfect PTA of 99.4% while suffering only a 1.8% drop in CTA (see Table 5.1 first row for exact values). As we show in Section 5.2, these accuracies improve with a different choice in trigger. This is the first demonstration of successful attack that only corrupts the labels, in constrast to the standard attacks that heavily rely on adding a trigger to the training images.

Scenario two: knowledge distillation. Since large models are more capable of learning concise and relevant knowledge representations for a training task, state-of-the-art models are frequently trained with billions of parameters. Such scale is often impractical for deployment on edge devices, and knowledge distillation, introduced in the seminal work of [29], has emerged as a reliable solution for distilling the knowledge of large models into much smaller ones without sacrificing the performance, e.g., [12, 44, 63]. Concretely, knowledge distillation trains a smaller student model on training images labeled with the softmax outputs of a teacher model.

While commonplace, the practice of distilling publicly-shared teacher models leaves student models susceptible to adversarially-controlled softmax outputs. While one might believe that distilled models are robust against backdoor attacks because the student is trained on clean images, as we show in Fig. 5.2b, not only does our attack work, but, in this setting, it is afforded more flexibility to control the soft outputs. The strength of the attack is again measured by the CTA-PTA trade-off, and, with more knobs to turn, we find that our attack in some cases is stronger. Note that, unlike workers on crowd-sourcing platforms, there is no additional cost for a teacher model to corrupt more training examples.

1.1 Contributions

Typical backdoor poisoning attacks on image classification models require some control over both the images and labels of a dataset. This is natural because the backdoor's trigger is applied in the space of the images and the desired outcome is a change in the predicted label. In this work, we demonstrate FLIP, to our knowledge the first backdoor attack that leaves the dataset's images untouched, corrupting only the dataset's labels. This is surprising for two reasons: (*i*) it is no longer clear how the model learns the backdoor's trigger, since the trigger is not applied to any of the model's training data and (*ii*) unlike random label flips, the corrupted labels chosen by FLIP do not degrade the clean accuracy of the model. Our proposed attack is highly effective, requiring very few label flips to succeed and transfers across different choices of architecture, trigger, and dataset.

Additionally, we propose a modification of FLIP, called softFLIP, which can be applied in the setting of knowledge distillation. To our knowledge softFLIP is the first backdoor targeting knowledge distillation with clean data which allows the attacker to use arbitrary triggers. In this setting, softFLIP further improves on the effectiveness of FLIP.

1.2 Threat Model

We assume the threat model of [26, 67] in which, as described above, an adversary seeks to gain control over the predictions of a user's model by injecting corrupted data into the training set. At evaluation, the attacker seeks to induce a fixed target-label prediction y_{target} whenever an example is armed with a trigger applied by fixed transform $T(\cdot)$. To measure success, a backdoored model f is evaluated on both on its Clean Test Accuracy (CTA) and Poison Test Accuracy (PTA). Concretely,

$$\text{CTA} := \mathbb{P}_{(X,y)\sim S_{\text{ct}}}[f(X) = y] \quad \text{and} \quad \text{PTA} := \mathbb{P}_{(X,y)\sim S'_{\text{ct}}}[f(T(X)) = y_{\text{target}}] \;, \tag{1.1}$$

where S_{ct} is the clean test set, and $S'_{\text{ct}} \subseteq S_{\text{ct}}$ is a subset to be used in computing PTA. An attack is successful if high CTA and high PTA are achieved (towards top-right of Figure 1.2).

We assume that the adversary has access to (a subset of) the training data, but, can corrupt only the soft or hard labels. We investigate a fundamental question: how successful is an adversary who can only corrupt the labels in the training data? This new label-only attack surface is motivated by two concrete use-cases, crowd-sourced labels and knowledge distillation, explained in Chapter 1. Under the crowd-sourcing scenario, the adversary can only change the label to one of the classes in the task. Under the knowledge distillation scenario, the adversary has the freedom to give an image an arbitrary soft label.

Chapter 2

RELATED WORK

2.1 Backdoor Attacks.

Backdoor attacks were introduced in [26]. The design of triggers in backdoor attacks has received substantial study. Many works choose the trigger to appear benign to humans [26, 7, 45, 48], directly optimize the trigger to this end [37, 20], or choose natural objects as the trigger [74, 14]. Poison data has been constructed to include no mislabeled examples [68, 82], optimized to conceal the trigger [55] or to evade statistical inspection of latent representations [58, 19, 76, 16]. Backdoor attacks have been demonstrated in a wide variety of settings, including federated learning [71, 4, 64], transfer learning [77, 55], and generative models [56, 54].

Backdoors can be injected in many creative ways, including poisoning of the loss [2], data ordering [59], or data augmentation [53]. With a more powerful adversary who controls not only the training data but the model itself, backdoors can be planted into a network by directly modifying the weights [21, 31, 81], by flipping a few bits of the weights [5, 6, 52, 66, 13], by modifying the structure of the network [25, 65, 39]. Closest to our work are label flipping approaches to inject backdoors in [14, 24], which exploits label correlations in multilabel datasets [14] or via soft labels in distillation [24]. [24] assumes a significantly more powerful adversary who can design the trigger also, whereas we assume both the trigger and the target label are given. The attack proposed in [14] is designed for multi-label tasks and cannot be applied to our setting with single label tasks. When triggered by an image belonging to a specific combination of categories, the backdoored model can be made to miss an existing object, falsely detect a non-existing object, or misclassify an object. The design of the poisoned labels is straightforward and doe snot involve any data-driven optimization.

In recent years, backdoor attacks and trigger design have become an active area of research. In the canonical setting, first introduced by [26], an attacker injects malicious feature-based perturbations into training associated with a source class and changes the labels to that of a target class. In computer vision, the first triggers shown to work were pixel-based stamps [26]. To make the triggers harder to detect, a variety of strategies including injecting periodic patterns [41] and mixing triggers with existing features [16] were introduced. These strategies however fail to evade human detection when analysis couples the images and their labels, as most backdoored images will appear mislabeled. In response, [68] and [82] propose generative ML-based strategies to interpolate images from the source and the target, creating images that induce backdoors with consistent labels. Notably, our method does not perturb images in training. Instead, that information is encoded in the labels being flipped.

2.2 Backdoor Defenses.

Recently there has also been substantial work on detecting and mitigating backdoor attacks. When the user has access to a separate pool of clean examples, they can filter the corrupted dataset by detecting outliers [40, 36, 61], retrain the network so it forgets the backdoor [42], or train a new model to test the original for a backdoor [35]. Other defenses assume the trigger is an additive perturbation with small norm [70, 17], rely on smoothing [69, 73], filter or penalize outliers without clean data [23, 64, 61, 8, 51, 67, 27] or use Byzantine-tolerant distributed learning techniques [8, 1, 15]. In general, it is possible to embed backdoors in neural netowrks such that they cannot be detected [25].

2.3 Knowledge Distillation.

Initially proposed in [29], knowledge distillation (KD) is a strategy to transfer learned knowledge between models. KD has been used to defend against adversarial perturbations [50], allow models to self-improve [79, 80], and boost interpretability of neural networks [43]. Knowledge distillation has been used to defend against backdoor attacks by distilling with clean data [78] and by also distilling attention maps [38, 75]. Bypassing such knowledge distillation defenses is one of the two motivating use-cases of our attack.

2.4 Dataset Distillation and Trajectory Matching.

Introduced in [72],the goal of dataset distillation is to produce a small dataset which captures the essence of a larger one. Dataset distillation by optimizing soft labels has been explored using the neural tangent kernel [47, 46] and gradient-based methods [9, 62]. Our method attempts to match the training trajectory of a normal backdoored model, with standard poisoned examples, by flipping labels. Trajectory matching has been used previously for dataset distillation [11] and bears similarity to imitation learning [49, 22, 30]. The use of a proxy objective in weight space for backdoor design appears in the KKT attack of [34].

Chapter 3

WARM-UP: ORDINARY LEAST SQUARES

In this section, we consider a *very* simplified version of FLIP's threat model, in which a user seeks to train a linear model $y = X^\top W$ on dataset $X \in \mathbb{R}^{n\times d}$ labeled by $y \in (-1,1)^n$ with weights $W \in \mathbb{R}^d$. The attacker, consequently, has control over the labels y and seeks to induce predictions y' with poisoned dataset X'. While FLIP, obviously, solves a much harder problem, we hope to motivate the work in the rest of this thesis by demonstrating the unexpected power attackers hold over a user's predictions with control over just the labels.

In particular, as we show informally in Proposition 3.1, with *read-only* access to X an attacker can compute labels y that perfectly determine the model's output y' on dataset X' when the features are linearly independent. When X' is not full rank, the outputs can be predicted within some bound.

Proposition 3.1. *Let $X \in \mathbb{R}^{n\times d}$ and $X' \in \mathbb{R}^{m\times d}$ be train and test images, respectively, where $y \in (-1,1)^n$ and $y' \in (-1,1)^m$ are their labels. Assume that $m \geq d$, X has full column-rank, and $\operatorname{rank}(X') = r \leq d$. We claim that*

$$\min_y \|\hat{y} - y'\| \leq \epsilon_p \sqrt{m},$$

where $\hat{y} = X'\widehat{W}$ for $\widehat{W} = \arg\min_W \|y - X^\top W\|^2$ and $\|\cdot\|$ refers to the 2-norm. ϵ_p denotes the pseudoinverse error $\|I - HH^+\|$ for $H = X'(X^\top X)^{-1}X^\top$ and $H^+ = \operatorname{pinv}(H)$.

Proof. Since X is full column rank, we can solve for $\widehat{W} = (X^\top X)^{-1}X^\top y$, directly, so that:

$$\hat{y} = X'(X^\top X)^{-1}X^\top y.$$

For convenience, define $H := X'(X^\top X)^{-1}X^\top$. In the case that $r = m = d$, we have that H is full rank so that $(H^\top H)^{-1}H^\top y' = \arg\min_y \|Hy - y'\|$ and $\min_y \|Hy - y'\| = 0$. However, in the alternate case (i.e., that $r < m$), we note that H is not full rank, and, consequently, $(H^\top H)^{-1}$ is not well defined. We instead consider the pseudo-inverse H^+ that minimizes the following: $H^+ y' = \arg\min_y \|Hy - y'\|$ in, possibly, lossy fashion. We then have that:

$$\begin{aligned}\min_y \|\hat{y} - y'\| &= \min_y \|Hy - y'\| \\ &= \left\|HH^+ y' - y'\right\| \\ &\leq \left\|I - HH^+\right\|_{op} \|y'\| \\ &\leq \epsilon_p \sqrt{m},\end{aligned}$$

where the final inequality uses the induced spectral norm of $I - HH^+$ and $\sigma_{\max}$ represents the largest singular value. □

Chapter 4

FLIPPING LABELS TO INJECT POISON (FLIP)

Directly optimizing data (labels in our case) is computationally hard, since, doing so would require optimizing through the entire model-training process. Inspired by trajectory matching for dataset distillation [11], we, instead, propose optimizing our logits at the minibatch-level through our proxy objective. Informally, we seek to optimize our labels of the user's dataset to produce a model parameter trajectory similar to that of 'expert models' trained on a 'normally' poisoned version of the user's data.

Our proposed algorithm FLIP proceeds in three steps: (1) we train several backdoored models "normally" using data poisoned with an arbitrary, attacker-selected trigger, saving checkpoints throughout the training process; (2) we optimize continuous logits that, when combined with clean images, yield parameter trajectories close to training with poisoned data; (3) finally, we convert our continuous logits to a one-hot encoded label.

4.1 Step 1: Training m Expert Models

The first step of our attack is to record a set of training trajectories (i.e., the checkpoints of *expert models* trained on data corrupted as per a traditional backdoor attack with trigger $T(\cdot)$ and target y_{target} of interest). Concretely, a poisoned dataset $\mathcal{D}_p = \mathcal{D}_c \cup \{p_1, \cdots\}$ is created from a clean training dataset $\mathcal{D}_c$ as follows. Given a choice of source label y_{source}, target label y_{target}, and trigger $T(\cdot)$, each poisoned example p is constructed by applying the trigger, T, to each image of class y_{source} in $\mathcal{D}_c$ and giving it label y_{target}. We next train m expert models to record a collection of m expert trajectories, each one with new random initalization and minibatches. The j-th trajectory is a sequence, $\{(\theta_k^{(j)}, B_k^{(j)})\}_{k=1}^{K}$, of model parameter, $\theta_k^{(j)}$, and minibatch of examples, $B_k^{(j)}$, over K training iterations. We find that small values of K

work well since checkpoints later on in training drift away from the student model training trajectory. We propose using $m > 1$ trajectories to ensure our attack generalizes to the unknown hyperparameters in the student model training, as we demonstrate in Appendix C.1. We assume for that there is a single source class y_{source} and we know the clean data $\mathcal{D}_c$.

4.2 Step 2: Matching Student Model Trajectory

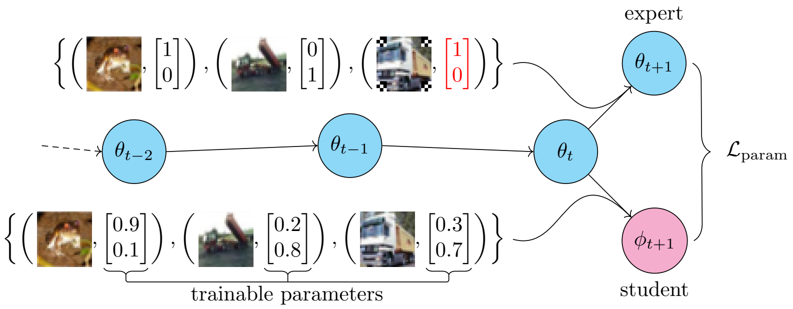

expert
θ_{t+1}
$\begin{bmatrix}1\\0\end{bmatrix}$
$\begin{bmatrix}0\\1\end{bmatrix}$
$\begin{bmatrix}1\\0\end{bmatrix}$
θ_{t-2}
θ_{t-1}
θ_t
$\mathcal{L}_{\text{param}}$
$\begin{bmatrix}0.9\\0.1\end{bmatrix}$
$\begin{bmatrix}0.2\\0.8\end{bmatrix}$
$\begin{bmatrix}0.3\\0.7\end{bmatrix}$
ϕ_{t+1}
trainable parameters
student

Figure 4.1: Illustration of the FLIP loss function in Equation (4.1): Starting from the same parameters θ_k, two separate gradient steps are taken, one containing typical backdoor poisoned examples to compute θ_{k+1} (from the expert trajectory recorded in step 1) and another with only clean images but with our synthetic labels to compute ϕ_{k+1}.

The next step of our attack is to find a set of soft labels for the images in the clean training set such that a student model trained on this new data follows a trajectory close to that of a traditionally-backdoored expert model. Let $\text{softmax}(\hat{\ell}_x)$ denote the soft labels for each clean example $x \in \mathcal{D}_c$ such that $\hat{\ell}_x$ is a real-valued vector to be optimized over. Concretely, we sample a training iteration $k \in [K]$ and model $j \in [m]$, uniformly at random, and take two separate gradient steps from the corresponding expert checkpoint $\theta_k^{(j)}$. The first gradient update, with minibatch $B_k^{(j)} \subseteq \mathcal{D}_p$, gives $\theta_{k+1}^{(j)}$ (already computed in the expert trajectory from step 1). The next gradient update ϕ_{k+1} is computed using the clean version, $\tilde{B}_k^{(j)}$, of the minibatch $B_k^{(j)}$, with each poisoned image replaced with the corresponding clean version, and the labels swapped for the current soft-labels, $\hat{\ell}_x$. To match these two trajectories, we minimize the following loss in (4.1) over the soft-labels $\{\hat{\ell}_x\}_{x\in\tilde{B}_k^{(j)}}$ using stochastic gradient descent:

$$\frac{1}{m\,K}\sum_{j=1}^{m}\sum_{k=1}^{K}\Big[\,\mathcal{L}_{\text{param}}(\{\hat{\ell}_x\}_{x\in\tilde{B}_k^{(j)}};\theta_k^{(j)},\theta_{k+1}^{(j)},\eta_s,\tilde{B}_k^{(j)})\;:=\;\frac{\|\theta_{k+1}^{(j)}-\phi_{k+1}^{(j)}\|^2}{\|\theta_{k+1}^{(j)}-\theta_k^{(j)}\|^2}\,\Big]\;,\text{ where} \tag{4.1}$$

$$\phi_{k+1}^{(j)}\;=\;\theta_k^{(j)}-\eta_s\sum_{(x,y)\in\tilde{B}_k^{(j)}}\nabla_\theta\mathcal{L}_{\text{expert}}(\theta_k^{(j)};(x,\text{softmax}(\hat{\ell}_x)))\;,\text{ and}$$

$$\mathcal{L}_{\text{expert}}(\theta;(x,y))\;=\;-\sum_{c=1}^{C}\log\text{softmax}(f(x))_c\cdot y_c\;.$$

The goal is to control the soft-labels $\hat{\ell}_x$ for each example in the minibatch $\tilde{B}_k^{(j)}$ such that the trajectory, $\phi_{k+1}^{(j)}$, matches that of a backdoored training, $\theta_{k+1}^{(j)}$. The normalization by $\|\theta_{k+1}^{(j)}-\theta_k^{(j)}\|^2$ ensures that we do not over represent the updates earlier in the training where we make larger updates. We give psuedocode for the second step in Algorithm 1 and details on our implementation in Appendix A.

Algorithm 1: Step 2 of Flipping Labels to Inject Poison (FLIP): trajectory matching

```
Input: number of iterations $N$, expert trajectories $\{(\theta_k^{(j)}, B_k^{(j)})\}_{j\in[m],k\in[K]}$, clean
       version of the minibatches $\{\tilde{B}_k^{(j)}\}_{j\in[m],k\in[K]}$, student learning rate $\eta_s$, label
       learning rate $\eta_l$
for $N$ iterations do
    Sample $j \in [m]$ and $k \in [K]$ uniformly at random;
    for $(x, y) \in \tilde{B}_k^{(j)}$ do
        $\hat{\ell}_x \leftarrow \hat{\ell}_x - \eta_l \nabla_{\hat{\ell}_x} \mathcal{L}_{\text{param}}(\{\hat{\ell}_x\}_{x\in B_k}; \theta_k^{(j)}, \theta_{k+1}^{(j)}, \eta_s, \tilde{B}_k^{(j)})$;
return $\{\hat{\ell}_x\}_{x\in\mathcal{D}_c}$
```

4.3 Step 3: Selecting Label Flips

We find that the logits, $\{\hat{\ell}_x\}$, computed in the previous section can be used to select a small number of *label flips* which can successfully backdoor a student model. Informally, we want to flip the label of an example when its logits have a high confidence in an incorrect prediction. To this end, we define the *margin* for an example as the logit of the correct class minus the largest logit of the incorrect classes. Then, to select k total label flips, we simply choose the k examples with the smallest (possibly negative) margin and set their class to the corresponding highest incorrect logit.

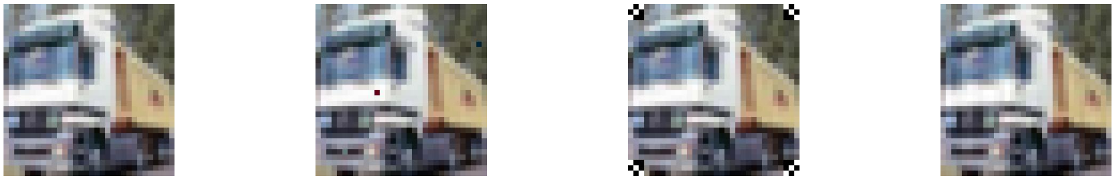
In practice, we find that aggregating the results of multiple runs of Algorithm 1 gives further gains in performance. In this case, we instead have multiple margins per example. As shown in Fig. 6.1, we find it is effective to select examples in ascending order of maximum margin over all runs, which has the effect of selecting label flips with high confidence in all runs.

4.3.1 softFLIP

If an attacker has control over the logit labels of the user's dataset, as in the knowledge distillation setting, they can opt to forgo the final discretization step of FLIP and provide $\{\hat{\ell}_x\}$ to the user directly. As we show in Section 5.3, this often leads to better attack performance. In the case that the user seeks model weights to distill from, the adversary can always train a model to high accuracy on the synthetic labels. To adjust attack strength, an attacker can linearly interpolate $\{\hat{\ell}_x\}$ with a one-hot encoded true labels.

Chapter 5

EXPERIMENTS



(a) clean (b) pixel: p (c) turner: t (d) sinusoidal: s

Figure 5.1: Triggers used for our main experiments

In this section, we evaluate FLIP and softFLIP on two standard datasets: CIFAR-10 and CIFAR 100; two architectures: ResNet-32 and ResNet-18; and three trigger styles: sinusoidal, pixel, and Turner. All experiments were done on a computing cluster containing NVIDIA A40 and 2080ti GPUs with no single experiment taking more than an hour on the slower 2080tis. All results are averaged over ten runs of the experiment with errors reported in Appendix B.

5.1 Setup and Evaluation

Setup. The labels for each experiment in this section were generated using 25 iterations of Algorithm 1 relying on 50 expert models trained for 20 epochs each. Each expert was trained on a dataset poisoned using one of the following triggers shown in Fig. 5.1; (1) pixel [67]: three pixels are altered (2) Turner [68]: at each corner, a 3 × 3 patch of black and white pixels is placed, and (3) sinusoidal [7]: sinusoidal noise is added to each image. The datasets were poisoned by adding an additional 5000 (one-class worth) poisoned images to CIFAR-10 and 2500 to CIFAR-100 (in which we poison all classes in the coarse label).

			150	300	500	1000	1500
CIFAR-10	r32	s	92.26/12.4	92.09/54.9	91.73/87.2	90.68/99.4	89.87/99.8
		t	92.37/28.4	92.03/95.3	91.59/99.6	90.80/99.5	89.91/99.9
		p	92.24/03.3	91.67/06.0	91.24/10.8	90.00/21.2	88.92/29.9
	r18	s	94.13/13.1	93.94/32.2	93.55/49.0	92.73/81.2	92.17/82.4
CIFAR-100	r32	s	78.83/08.2	78.69/24.7	78.52/45.4	77.61/82.2	76.64/95.2
	r18	s	82.87/11.9	82.48/29.9	81.91/35.8	81.25/81.9	80.28/95.3

Table 5.1: FLIP is robust to choice of dataset, model architecture, and trigger across different numbers of flipped labels. Each row denotes the CTA/PTA pairs averaged over 10 experiments. An experiment is characterized by dataset in the first column, architecture (ResNet-32 or ResNet-18) in the second column, and trigger (sinusoidal, Turner, or pixel) in the third.

Evaluation. To evaluate our attack on a given setting (described by dataset, architecture, and trigger) we follow standard backdoor attack evaluation procedure and measure the attack's intensity and stealth by recording the PTA and CTA as described in Section 1.2. To vary the intensity-stealth tradeoff we, for the FLIP experiments, change the number of flipped labels using the procedure detailed in Section 4.3. Meanwhile, for softFLIP, we interpolate with the true labels as described in 4.3.1. When trained on the user's CIFAR-10 set with true, clean labels, the baseline CTAs are given by Table 5.2.

5.2 Crowdsourced Labeling

In this section we discuss the efficacy of FLIP when allowed a budget of k discrete label flips and given information on the user's model architecture and optimizer. In Chapter 6, we show

	r32	r18
CIFAR-10	92.51	94.23
CIFAR-100	79.87	83.54

Table 5.2: Baseline CTAs of ResNet-32s and ResNet-18s on CIFAR-10 and CIFAR-100 when trained on the user's dataset with ground-truth labels.

that, while useful, these assumptions are scarily unnecessary.

Baseline Comparison. Since, to the best of our knowledge, we are the first to induce a backdoor using only labels in the standard image classification setting, there are, unfortunately, a dearth of comparable baselines. As such, in Fig. 1.2, we introduce what we call the dot-product baseline, computed by ordering each image by its dot-product with the trigger and flipping the labels of the k images with the highest scores. Here, dot-product serves as a similarity metric between image and trigger, and, perhaps surprisingly, as we show in Fig. 1.2, flipping the labels of the images closest to the trigger is not sufficient in achieving high poison accuracy while maintaining good downstream performance. Raw performance numbers for the baseline can be found in Table B.2.

Meanwhile, our attack manages to maintain high CTA while injecting an effective backdoor in far fewer label flips than the baseline. We remark that instead of flipping the labels of points that individually look similar to poisoned images (i.e., to the trigger), our optimization adjusts batches of labels that are similar in induced-parameter- (or gradient-) space to poisoned batches.

Robustness. As Table 5.1 suggests, FLIP is robust to choice of dataset, trigger, and model architecture. While attack performance is stronger on CIFAR-10, we note that CIFAR-100 has a greatly reduced attack surface with half of the per-class data points. Still, FLIP retains 97.2% of the original CTA on CIFAR-100 when compared 98.0% on CIFAR-10 when using

ResNet-32s and a budget of 1000 flips. Each attack generates 82.2% and 99.4% poison accuracy, respectively.

Fig. 5.2a depicts the PTA-CTA tradeoff of each of our triggers on ResNet-32s and CIFAR-10. While the Turner poisoner performs the best, it is also the easiest to visually spot in an image. Meanwhile, the pixel trigger has the smallest pixel perturbation of any of the triggers and consequently performs the worst. We find that the sinusoidal trigger strikes a good balance between the two. While hard to deduce by mere inspection, the attacks are still potent.

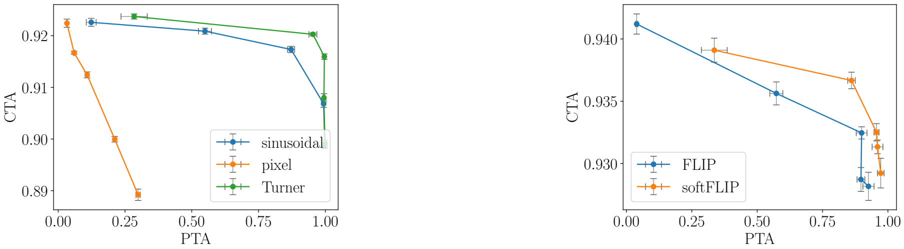
Finally, when chosen to match, we find that FLIP is robust to choice of expert and dowstream model architecture. While the ResNet-32 experiments had higher PTA, we remark that the ResNet-18s retained a higher percentage of the clean performance (3.9% and 0.1% on average for CIFAR-10 and CIFAR-100, respectively). Since both architectures are trained with the same setup, this difference may come down to a lack of hyperparameter tuning. We discuss the efficacy of the attack when these architectures do not match in Section 6.1.

Few-Shot Performance. With as few as 300 flipped labels (i.e., 0.6% of the dataset), FLIP is highly effective in multiple settings generating poison accuracy up to 95.3% while retaining 99.5% CTA. Surprisingly, at only 150 flipped labels and 0.3% of the dataset, the attack still works, generating as high as 28.4% PTA with 99.8% of the clean accuracy.

5.3 Distillation

In this section we investigate softFLIP: the version of our attack where an adversary has control over the logits of a dataset. With this added flexibility, as shown in Table 5.3, we find that the attack maintains the same robustness properties as the discrete version while outperfoming it in some settings as shown in Fig. 5.2b. We again assume access to information on the user's model architecture and optimizer while not necessary.

Baseline Comparison. For softFLIP we consider two baselines: (1) distilling directly from a traditionally poisoned model and (2) a discretized version of the softFLIP logits. As noted in [78, 14, 24, 50] the knowledge distillation process was largely thought to be robust



(a) Triggers

(b) softFLIP vs. FLIP

Figure 5.2: Intensity-Stealth trade-offs (a) for FLIP using different triggers with ResNet-32s trained on CIFAR-10 and (b) of softFLIP and its discretized version for ResNet-18s, CIFAR-10, and the sinusoidal trigger. Each point is associated with an interpolation percentage $\alpha \in \{0.4, 0.6, 0.8, 0.9, 1.0\}$. Horizontal and vertical error bars are averaged over 10 downstream training runs.

to backdoor attacks and adversarial perturbations alike. To measure this robustness, we record the PTA and CTA of a student model distilled from a traditionally poisoned model (i.e., alterations to training images and labels). For this baseline, our teacher model had 93.91% CTA and 99.9% PTA while the student model exhibited a higher 94.39% CTA and a worse-than-random 0.00% PTA on average. Clearly, our attack outperforms this low bar, questioning the robustness properties of knowledge distillation.

Perhaps more interestingly, to measure how much benefit control over the logits provides, we also compare the logit version of our attack to a discretized version on ResNet-18s, CIFAR-10, and the sinusoidal trigger. For direct comparison on parameter α, instead of producing the discretized labels as described in Section 4.3 we produce the soft labels as usual and perform an argmax on each set of logits. Fig. 5.2b shows that the soft logits unilaterally outperform the discrete versions implying that the added information encoded in soft logits may contribute to a stronger attack.

		0.4	0.6	0.8	0.9
r32	s	90.11/100.	90.45/99.9	91.02/99.0	91.95/25.3
	t	88.44/100.	88.99/100.	89.86/100.	90.85/100.
	p	88.62/38.8	89.10/31.1	91.64/05.1	92.04/00.1
r18	s	93.13/96.0	93.25/95.6	93.67/86.1	93.91/33.6

Table 5.3: softFLIP is robust to architecture and trigger on CIFAR-10 across choice of interpolation percentage $\alpha \in \{0.0, 0.2, 0.4, 0.6, 0.8, 0.9\}$. α interpolates between our logit-labels $\alpha = 0.0$ and the true labels of the dataset $\alpha = 1.0$. Each row denotes the CTA/PTA pairs averaged over 10 experiments.

Chapter 6

ABLATIONS

In the previous section we assumed that: (1) the attacker knows the user's downstream model architecture and optimization strategy, and (2) the attacker's label flips are never overruled. In this section, we relax those assumptions and broaden the attack surface.

6.1 Downstream Model and Optimizer

The attacker's strategy in our previous experiments was to train expert models to mimic exactly the downstream architecture and optimizer setup of the user. However, it remained unclear whether the attack would generalize if the user, for instance, opted for a smaller model than expected. As such, we looked at varying (1) model architecture and (2) optimizer between expert and downstream setups. For (2) we use SGD for the expert models and Adam [33] for the user. We additionally analyze what happens when both are different.

As Table 6.1 indicates, the attack still performs well when information is limited. When varying optimizer, we found that downstream CTA dropped, but, interestingly, the PTA for the ResNet-18 case was almost unilaterally higher. We found a similar trend for upstream ResNet-32s and downstream ResNet-18s when varying model architecture. Surprisingly, the strongest PTA numbers for budgets higher than 150 across all experiments with a downstream ResNet-18 were achieved when the attacker used *a different expert model and optimizer*. While we hypothesize that these phenomena are a result of the ResNet-32 experts and Adam optimizers being better suited to our hyperparameters, the attack is, nonetheless, robust to varied architecture and optimizer.

		150	300	500	1000	1500
(1)	r18 → r32	92.44/19.2	92.16/53.3	91.84/74.5	90.80/92.9	90.00/95.2
	r32 → r18	93.86/04.8	93.89/36.0	93.56/60.0	92.76/86.9	91.75/98.0
(2)	r32 → r32	90.79/11.8	90.50/43.2	90.08/80.6	89.45/97.5	88.33/99.0
	r18 → r18	93.17/20.3	93.08/47.0	92.94/65.6	91.91/89.9	91.16/93.1
(1 + 2)	r18 → r32	90.86/12.3	90.57/40.9	90.28/59.7	89.39/83.0	88.59/89.2
	r32 → r18	93.32/09.4	93.05/52.9	92.70/85.3	91.72/99.2	90.93/99.7

Table 6.1: FLIP performs well even when the expert and downstream (1) model architectures and (2) optimizers are different. Experiments are computed on CIFAR-10 using the sinusoidal trigger. Each row denotes the CTA/PTA pairs averaged over 10 experiments. The second column is structured as follows: expert → downstream.

6.2 Downstream Dataset

Crowdsourced data labeling tasks often involve many labelers for any given image [18]. As such, the final label can be seen as a (likely weighted) sample from a distribution of labeler responses. To simulate the attacker's probabilistic control over the final labels in this case, we investigate the efficacy of FLIP when only $n \in \{150, 300, 500, 1000, 1500\}$ of an attacker's 1500 flips are accepted by the user. In particular, to sample the n points three selection strategies are studied: (1) standard max-margin (i.e., the n points with the largest margin), (2) random sample (i.e., n random points), and (3) min-margin (i.e., the n points with the smallest margin).

Unsurprisingly, as shown in Fig. 6.1, the random sample interpolates the clean and poison performance of the best (max) and worst (min) case samples. In the random and min cases, the attack only works with sufficiently high budget (500 for the average and 1000 for the min case). However, before these budget elbow points, the attacks' poor performance points to the importance of flipping labels with high margin (as described in Section 4.3). So, as long as an attacker flips a sufficient number of high margin targets, the attack should succeed.

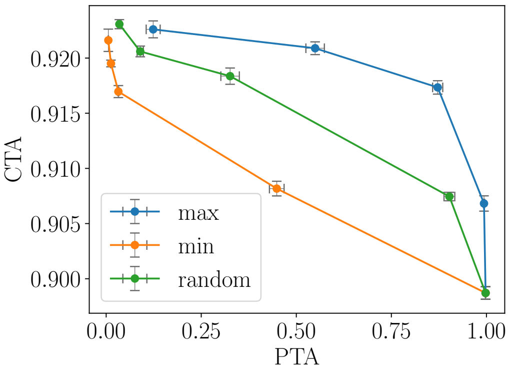
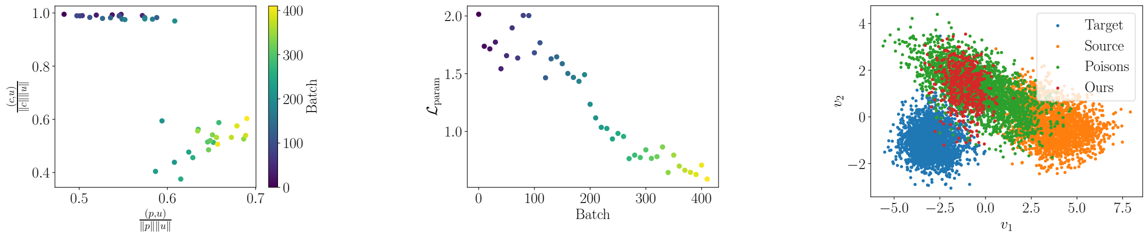


Figure 6.1: The attacker need not flip all labels they seek to so long as they flip enough high-margin labels. We measure the performance of three sampling methods: (1) standard max-margin (i.e., points with the largest margin), (2) random sample (i.e., random points), and (3) min-margin (i.e., points with the smallest margin). Horizontal and vertical error bars are averaged over 10 downstream training runs. For the random case, the random sample is recomputed for each experiment.

Chapter 7

DISCUSSION



(a) (b) (c)

Figure 7.1: FLIP in the gradient and representation spaces. (a) The gradient induced by our labels u shifts in direction (i.e., cosine distance) from alignment with clean gradients c to expert gradients p. Each point represents a 25-batch average. (b) The drop in $\mathcal{L}_{\text{param}}$ coincides with the shift in Fig. 7.1a. Each point again represents a 25-batch average. (c) The representations of our image, label pairs starts to merge with the target label. Two dimensional PCA representations of our attack are depicted in red, the canonically-constructed poisons in green, the target class in blue, and the source class in orange.

In this section, we seek to explain how FLIP exploits the demonstrated vulnerability in the label space, and the implications FLIP has moving forward.

7.1 Under the Hood

To begin, our parameter loss $\mathcal{L}_{\text{param}}$ optimizes our logits $\hat{\ell}$ to minimize the squared error (up to some scaling) between the parameters induced by (1) a batch of poisoned data and (2) a batch of clean data with our labels. As we make explicit in Appendix D, we can also interpret this objective as a minimization on the squared error of the *gradients* induced by our two batches. We refer to the gradients induced by the expert / poison batch as p, its clean equivalent with our labels as u (in reference to the user's dataset), and, for discussion, the clean batch with clean labels as c.

As shown in Fig. 7.1a, gradient vector u begins with strong cosine alignment to c in the early batches of training. Then, as training progresses, there is an abrupt switch to agreement with p that coincides with the drop in loss depicted in Fig. 7.1b. Informally, after around 200 batches, our method is able to induce gradients u similar to p with a batch of clean images by 'scaling' the gradient in the right directions using $\hat{\ell}$. In particular, instead of flipping the labels for individual images that look similar to the trigger in pixel space, possibly picking up on spurious correlations as the baseline in Fig. 1.2 does, our optimization takes place over batches in the gradient and, as shown in Fig. 7.1c, in representation spaces. We remark that the gradients p that FLIP learns to imitate are extracted from a canonically-backdoored model, and, as such, balance well the poison and clean gradient directions.

Interestingly, as we discuss in Section 6.1, the resultant labels $\hat{\ell}$ seem partially invariant to choice of downstream model and optimizer, which may suggest an intrinsic relationship in representation space between certain flipped images and the trigger. In fact, we observe in Section 6.2 that there exists a core set of images that produce high-margins in our algorithm that seem to greatly dictate the success of our attack. In particular, the attack success scales with the percentage of expected high-margin images.

7.2 Concluding Thoughts

In this thesis, we discuss FLIP, a novel backdoor attack that requires only label flips to succeed. FLIP is highly effective, achieving a favorable combination of high accuracy on both clean data and poisoned data while only corrupting a small percentage of the dataset's labels. In addition, we find that FLIP is remarkably robust to choices in dataset, architecture, trigger, and optimizer, which, when combined, construct a wide attack surface.

These attributes, we feel, pose an immense threat to any machine learning setting in which a practitioner does not have total control over the data's labels such as the crowdsourced labeling and knowledge distillation scenarios. As such, we think it is vital that machine learning practitioners learn to audit their data and implement safety measures to close vulnerabilities like the one that we discuss in this work. Additionally, although this attack may be used for harmful purposes, we hope this work will motivate further research into solutions (i.e., backdoor defenses and mitigations). Security is a cat-and-mouse game and hopefully, soon, we have one fewer hole to fill.

BIBLIOGRAPHY

[1] Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. Byzantine stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 4613–4623, 2018.

[2] Eugene Bagdasaryan and Vitaly Shmatikov. Blind backdoors in deep learning models. In *Usenix Security*, 2021.

[3] Eugene Bagdasaryan and Vitaly Shmatikov. Hyperparameter search is all you need for training-agnostic backdoor robustness. *arXiv preprint arXiv:2302.04977*, 2023.

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

[5] Jiawang Bai, Kuofeng Gao, Dihong Gong, Shu-Tao Xia, Zhifeng Li, and Wei Liu. Hardly perceptible trojan attack against neural networks with bit flips. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part V*, pages 104–121. Springer, 2022.

[6] Jiawang Bai, Baoyuan Wu, Zhifeng Li, and Shu-tao Xia. Versatile weight attack via flipping limited bits. *arXiv preprint arXiv:2207.12405*, 2022.

[7] Mauro Barni, Kassem Kallas, and Benedetta Tondi. A new backdoor attack in cnns by training set corruption without label poisoning. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 101–105. IEEE, 2019.

[8] Peva Blanchard, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, pages 119–129, 2017.

[9] Ondrej Bohdal, Yongxin Yang, and Timothy Hospedales. Flexible dataset distillation: Learn labels instead of images. *arXiv preprint arXiv:2006.08572*, 2020.

[10] Michael Buhrmester, Tracy Kwang, and Samuel D Gosling. Amazon's mechanical turk: A new source of inexpensive, yet high-quality, data? *Perspectives on psychological science*, 6(1):3–5, 2011.

[11] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A. Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories, 2022.

[12] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. *Advances in neural information processing systems*, 30, 2017.

[13] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Proflip: Targeted trojan attack with progressive bit flips. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7718–7727, 2021.

[14] Kangjie Chen, Xiaoxuan Lou, Guowen Xu, Jiwei Li, and Tianwei Zhang. Clean-image backdoor: Attacking multi-label models with poisoned labels only. In *The Eleventh International Conference on Learning Representations*, 2023.

[15] Lingjiao Chen, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. Draco: Byzantine-resilient distributed training via redundant gradients. In *International Conference on Machine Learning*, pages 903–912. PMLR, 2018.

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

[17] Edward Chou, Florian Tramer, and Giancarlo Pellegrino. Sentinet: Detecting localized universal attacks against deep learning systems. In *2020 IEEE Security and Privacy Workshops (SPW)*, pages 48–54. IEEE, 2020.

[18] J. Deng, K. Li, M. Do, H. Su, and L. Fei-Fei. Construction and Analysis of a Large Scale Image Ontology. Vision Sciences Society, 2009.

[19] Khoa Doan, Yingjie Lao, and Ping Li. Backdoor attack with imperceptible input and latent modification. *Advances in Neural Information Processing Systems*, 34, 2021.

[20] Khoa Doan, Yingjie Lao, Weijie Zhao, and Ping Li. Lira: Learnable, imperceptible and robust backdoor attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11966–11976, 2021.

[21] Jacob Dumford and Walter Scheirer. Backdooring convolutional neural networks via targeted weight perturbations. In *2020 IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–9. IEEE, 2020.

[22] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2021.

[23] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 113–125, 2019.

[24] Yunjie Ge, Qian Wang, Baolin Zheng, Xinlu Zhuang, Qi Li, Chao Shen, and Cong Wang. Anti-distillation backdoor attacks: Backdoors can really survive in knowledge distillation. In *Proceedings of the 29th ACM International Conference on Multimedia*, MM '21, page 826–834, New York, NY, USA, 2021. Association for Computing Machinery.

[25] Shafi Goldwasser, Michael P Kim, Vinod Vaikuntanathan, and Or Zamir. Planting undetectable backdoors in machine learning models. *arXiv preprint arXiv:2204.06974*, 2022.

[26] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[27] Jonathan Hayase, Weihao Kong, Raghav Somani, and Sewoong Oh. SPECTRE: Defending against backdoor attacks using robust statistics. In *International Conference on Machine Learning*, pages 4129–4139. PMLR, 2021.

[28] Jonathan Hayase and Sewoong Oh. Few-shot backdoor attacks via neural tangent kernels. In *International Conference on Learning Representations (ICLR)*, 2023.

[29] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[30] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

[31] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. Handcrafted backdoors in deep neural networks. *Advances in Neural Information Processing Systems*, 35:8068–8080, 2022.

[32] David Karger, Sewoong Oh, and Devavrat Shah. Iterative learning for reliable crowdsourcing systems. *Advances in neural information processing systems*, 24, 2011.

[33] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[34] Pang Wei Koh, Jacob Steinhardt, and Percy Liang. Stronger data poisoning attacks break data sanitization defenses. *Machine Learning*, 111(1):1–47, 2022.

[35] Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. Universal litmus patterns: Revealing backdoor attacks in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 301–310, 2020.

[36] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, pages 7167–7177, 2018.

[37] Shaofeng Li, Minhui Xue, Benjamin Zi Hao Zhao, Haojin Zhu, and Xinpeng Zhang. Invisible backdoor attacks on deep neural networks via steganography and regularization. *IEEE Transactions on Dependable and Secure Computing*, 18(5):2088–2105, 2020.

[38] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Neural attention distillation: Erasing backdoor triggers from deep neural networks, 2021.

[39] Yuanchun Li, Jiayi Hua, Haoyu Wang, Chunyang Chen, and Yunxin Liu. Deeppayload: Black-box backdoor attack on deep learning models through neural payload injection. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 263–274. IEEE, 2021.

[40] Shiyu Liang, Yixuan Li, and R Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *6th International Conference on Learning Representations, ICLR 2018*, 2018.

[41] Cong Liao, Haoti Zhong, Anna Squicciarini, Sencun Zhu, and David Miller. Backdoor embedding in convolutional neural network models via invisible perturbation, 2018.

[42] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 273–294. Springer, 2018.

[43] Xuan Liu, Xiaoguang Wang, and Stan Matwin. Improving the interpretability of deep neural networks with knowledge distillation. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 905–912. IEEE, 2018.

[44] Yifan Liu, Ke Chen, Chris Liu, Zengchang Qin, Zhenbo Luo, and Jingdong Wang. Structured knowledge distillation for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2604–2613, 2019.

[45] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection backdoor: A natural backdoor attack on deep neural networks. In *European Conference on Computer Vision*, pages 182–199. Springer, 2020.

[46] Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression. In *International Conference on Learning Representations*, 2020.

[47] Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. Dataset distillation with infinitely wide convolutional networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[48] Tuan Anh Nguyen and Anh Tuan Tran. Wanet-imperceptible warping-based backdoor attack. In *International Conference on Learning Representations*, 2020.

[49] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, Jan Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018.

[50] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.

[51] Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445*, 2019.

[52] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Tbt: Targeted neural network attack with bit trojan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13198–13207, 2020.

[53] Joseph Rance, Yiren Zhao, Ilia Shumailov, and Robert Mullins. Augmentation backdoors. *arXiv preprint arXiv:2209.15139*, 2022.

[54] Ambrish Rawat, Killian Levacher, and Mathieu Sinn. The devil is in the gan: Defending deep generative models against backdoor attacks. *arXiv preprint arXiv:2108.01644*, 2021.

[55] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. Hidden trigger backdoor attacks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11957–11965, 2020.

[56] Ahmed Salem, Yannick Sautter, Michael Backes, Mathias Humbert, and Yang Zhang. Baaan: Backdoor attacks against autoencoder and gan-based machine learning models. *arXiv preprint arXiv:2010.03007*, 2020.

[57] Victor S Sheng, Foster Provost, and Panagiotis G Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 614–622, 2008.

[58] Reza Shokri et al. Bypassing backdoor detection algorithms in deep learning. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 175–183. IEEE, 2020.

[59] Ilia Shumailov, Zakhar Shumaylov, Dmitry Kazhdan, Yiren Zhao, Nicolas Papernot, Murat A Erdogdu, and Ross J Anderson. Manipulating sgd with data ordering attacks. *Advances in Neural Information Processing Systems*, 34:18021–18032, 2021.

[60] Padhraic Smyth, Usama Fayyad, Michael Burl, Pietro Perona, and Pierre Baldi. Inferring ground truth from subjective labelling of venus images. *Advances in neural information processing systems*, 7, 1994.

[61] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. Certified defenses for data poisoning attacks. In *Advances in neural information processing systems*, pages 3517–3529, 2017.

[62] Ilia Sucholutsky and Matthias Schonlau. Soft-label dataset distillation and text dataset distillation. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[63] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355*, 2019.

[64] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.

[65] Ruixiang Tang, Mengnan Du, Ninghao Liu, Fan Yang, and Xia Hu. An embarrassingly simple approach for trojan attack in deep neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 218–228, 2020.

[66] MC Tol, S Islam, B Sunar, and Z Zhang. Toward realistic backdoor injection attacks on dnns using rowhammer. *arXiv preprint arXiv:2110.07683*, 2022.

[67] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. *Advances in neural information processing systems*, 31, 2018.

[68] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*, 2019.

[69] Binghui Wang, Xiaoyu Cao, Neil Zhenqiang Gong, et al. On certifying robustness against backdoor attacks via randomized smoothing. *arXiv preprint arXiv:2002.11750*, 2020.

[70] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723. IEEE, 2019.

[71] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[72] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

[73] Maurice Weber, Xiaojun Xu, Bojan Karlaš, Ce Zhang, and Bo Li. Rab: Provable robustness against backdoor attacks. *arXiv preprint arXiv:2003.08904*, 2020.

[74] Emily Wenger, Roma Bhattacharjee, Arjun Nitin Bhagoji, Josephine Passananti, Emilio Andere, Haitao Zheng, and Ben Y Zhao. Natural backdoor datasets. *arXiv preprint arXiv:2206.10673*, 2022.

[75] Jun Xia, Ting Wang, Jiepin Ding, Xian Wei, and Mingsong Chen. Eliminating backdoor triggers for deep neural networks using attention relation graph distillation, 2022.

[76] Pengfei Xia, Hongjing Niu, Ziqiang Li, and Bin Li. Enhancing backdoor attacks with multi-level mmd regularization. *IEEE Transactions on Dependable and Secure Computing*, 2022.

[77] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y Zhao. Latent backdoor attacks on deep neural networks. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2041–2055, 2019.

[78] Kota Yoshida and Takeshi Fujino. Countermeasure against backdoor attack on neural networks utilizing knowledge distillation. *Journal of Signal Processing*, 2020.

[79] Linfeng Zhang, Chenglong Bao, and Kaisheng Ma. Self-distillation: Towards efficient and compact neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4388–4403, 2021.

[80] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3713–3722, 2019.

[81] Zhiyuan Zhang, Lingjuan Lyu, Weiqiang Wang, Lichao Sun, and Xu Sun. How to inject backdoors with better consistency: Logit anchoring on clean data. *arXiv preprint arXiv:2109.01300*, 2021.

[82] Shihao Zhao, Xingjun Ma, Xiang Zheng, James Bailey, Jingjing Chen, and Yu-Gang Jiang. Clean-label backdoor attacks on video recognition models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14443–14452, 2020.

Appendix A
EXPERIMENTAL DETAILS

In this section, for completeness, we present some of the technical details on our evaluation pipeline that were omitted in the main text. For most of our experiments the pipeline proceeds by (1) training expert models, (2) generating synthetic labels, and (3) measuring our attack success on a downstream model. To compute final numbers, 10 downstream models were trained on each set of computed labels. We will publish our code on GitHub.

A.1 Datasets and Poisoning Procedures

We evaluate FLIP and softFLIP on two standard classification datasets of increasing difficulty: CIFAR-10 and CIFAR-100. For better test performance and to simulate 'real-world' use cases, we follow the standard CIFAR data augmentation procedure of (1) normalizing the data and (2) applying PyTorch transforms: RandomCrop and RandomHorizontalFlip. For RandomCrop, every epoch, each image was cropped down to a random 32×32 subset of the original image with the extra 4 pixels reintroduced as padding. RandomHorizontalFlip randomly flipped each image horizontally with a 50% probability every epoch.

To train our expert models we poisoned each dataset setting $y_{\text{source}} = 9$ and $y_{\text{target}} = 4$. Notably, since CIFAR-100 has only 500 images per fine-grained class, we use the coarse-labels for y_{source} and y_{target}. For CIFAR-10 this corresponds to a canonical self-driving-car-inspired backdoor attack setup in which the source label consists of images of trucks and target label corresponds to deer. For CIFAR-100, the mapping corresponds to a source class 'large man-made outdoor things' and a target of 'fruit and vegetables.' To poison the dataset, each $y_{\text{source}} = 9$ image had a trigger applied to it and was appended to the dataset.

Trigger Details. Our attack is general to choice of trigger, so, for our experiments, we used the following three triggers, in increasing order of strength. For our CIFAR-10 experiments we investigate all three styles, while for CIFAR-100 we use the periodic trigger. Demonstrations of the triggers are shown in Fig. 5.1.

- Pixel Trigger [67] ($T = p$). To inject the pixel trigger into an image, at three pixel locations of the photograph the existing colors are replaced by pre-selected colors. Notably, the original attack was proposed with a single pixel and color combination (that we use), so, to perform our stronger version, we add two randomly selected pixel locations and colors. In particular, the locations are $\{(11, 16), (5, 27), (30, 7)\}$ and the respective colors in hexadecimal are {#650019, #657B79, #002436}. This trigger is the weakest of our triggers with the smallest pixel-space perturbation

- Periodic / Sinusoidal Trigger [7] ($T = s$). The periodic attack adds periodic noise along the horizontal axis (although the trigger can be generalized to the vertical axis as well). We chose an amplitude of 6 and a frequency of 8. This trigger has a large, but visually subtle effect on the pixels in an image making it the second most potent of our triggers.

- Patch / Turner Trigger [68] ($T = t$). For our version of the patch poisoner, we adopted the 3×3 watermark proposed by the original authors and applied it to each corner of the image to persist through our RandomCrop procedure. This trigger seems to perform the best on our experiments, likely due to its strong pixel-space signal.

A.2 Models and Optimizers

For both our expert and downstream models we use the ResNet-32 and ResNet-18 architectures with around 0.5 and 11.4 million parameters, respectively. Initialized with random weights, each model achieves the performance in Table A.1 on the user's dataset without synthetic labels.

For our main experiments, the expert and downstream models were trained using SGD with a batch size of 256, starting learning rate of $\gamma = 0.1$ (scheduled to reduce by a factor of

	r32	r18
CIFAR-10	92.51 (±0.05)	94.23 (±0.08)
CIFAR-100	79.87 (±0.07)	83.54 (±0.08)

Table A.1: An expanded version of Table 5.2 in which each point is averaged over 10 downstream training runs and standard errors are shown in parentheses.

10 at epoch 75 and 150), weight decay of $\lambda = 0.0002$, and Nesterov momentum of $\mu = 0.9$. For Section 6.1, in which we explore differing the expert and downstream optimizers, we use Adam with the same batch size, starting learning rate of $\gamma = 0.001$ (scheduled to reduce by a factor of 10 at epoch 125), weight decay of $\lambda = 0.0001$, and $(\beta_1, \beta_2) = (0.9, 0.999)$.

We note that the hyperparameters were set to achieve near 100% train accuracy after 200 epochs, but, FLIP only requires the first 20 epochs of the expert trajectories. So, expert models were trained for 20 epochs while the downstream models were trained for the full 200. Expert model weights were saved every 50 iterations / minibatches (i.e., for batch size 256 around four times an epoch).

A.3 FLIP Details

For each iteration of Algorithm 1, we sampled an expert model at uniform random, while checkpoints were sampled at uniform random from the first 20 epochs of the chosen expert's training run. Since weights were recorded every 50 iterations, from each checkpoint a single gradient descent iteration was run with both the clean minibatch and the poisoned minibatch (as a proxy for the actual expert minibatch step) and the loss computed accordingly. Both gradient steps adhered to the training hyperparameters described above. The algorithm was run for 25 iterations through the entire dataset.

To initialize $\hat{\ell}$, we use the original one-hot labels y scaled by a temperature parameter C.

For sufficiently large C, the two gradient steps in Fig. 4.1 will be very similar except for the changes in the poisoned examples, leading to a low initial value of $\mathcal{L}_{\text{param}}$. However if C is too large, we suffer from vanishing gradients of the softmax. Therefore C must be chosen to balance these two concerns.

A.4 Compute

All of our experiments were done on a computing cluster containing NVIDIA A40 and 2080ti GPUs with tasks split roughly evenly between the two. To compute all of our numbers (averaged over 10 downstream models) we ended up computing approximately 1630 downstream models, 80 sets of labels, and 300 expert models. Averaging over GPU architecture, dataset, and model architecture, we note that each set of labels takes around 25 minutes to train. Meanwhile, each expert model takes around 10 minutes to train (fewer epochs with a more costly weight-saving procedure) and each downstream model takes around 40. This amounts to a total of approximately 1170 GPU-hours.

We note that the number of GPU-hours for an adversary to pull off this attack is likely significantly lower since they would need to compute as few as a single expert model (10 minutes) and a set of labels (25 minutes). This amounts to just over half of a GPU-hour given our setup (subject to hardware), a surprisingly low sum for an attack of this high potency.

Appendix B

COMPLETE EXPERIMENTAL RESULTS

In this section, we provide expanded versions of the key tables in the main text complete with standard errors as well as some additional supplementary material. As in the main text, we compute our numbers via a three step process: (1) we start by training 5 sets of synthetic labels for each (dataset, expert model architecture, trigger) tuple, (2) we then aggregate each set of labels, and (3) we finish by training 10 downstream models on each interpolation of the aggregated labels and the ground truths.

For our FLIP experiments in Appendix B.1, labels are aggregated using the margin strategy described in Section 4.3, and interpolated on the number of flipped labels. Meanwhile, for our softFLIP results in Appendix B.2, we aggregate as in Section 4.3.1 by taking the average logits for each image and linearly interpolating them on parameter α with the ground-truth labels.

B.1 Experiments: Crowdsourced Labeling

Fig. 1.2, Fig. 5.2a, and Table 5.1 showcase FLIP's performance when compared to a dot-product-based baseline and in relation to changes in dataset, model architecture, and trigger. In this section, we provide expanded versions of those results and present the raw numbers for the dot-product baseline.

B.2 Experiments: Distillation

Fig. 5.2b and Table 5.3 showcase softFLIP in comparison to discrete label flips and in relation to changes in dataset, model architecture, and trigger. In this section, we provide expanded versions of those results with standard errors and an additional table showcasing the numbers

			150	300	500	1000	1500
CIFAR-10	r32	s	92.26 (0.1)/12.4 (1.8)	92.09 (0.1)/54.9 (2.4)	91.73 (0.1)/87.2 (1.3)	90.68 (0.1)/99.4 (0.2)	89.87 (0.1)/99.8 (0.1)
		t	92.37 (0.0)/28.4 (4.9)	92.03 (0.0)/95.3 (1.5)	91.59 (0.1)/99.6 (0.2)	90.80 (0.1)/99.5 (0.3)	89.91 (0.1)/99.9 (0.1)
		p	92.24 (0.1)/03.3 (0.2)	91.67 (0.0)/06.0 (0.2)	91.24 (0.1)/10.8 (0.3)	90.00 (0.1)/21.2 (0.3)	88.92 (0.1)/29.9 (0.8)
	r18	s	94.13 (0.1)/13.1 (2.0)	93.94 (0.1)/32.2 (2.6)	93.55 (0.1)/49.0 (3.1)	92.73 (0.1)/81.2 (2.7)	92.17 (0.1)/82.4 (2.6)
CIFAR-100	r32	s	78.83 (0.1)/08.2 (0.6)	78.69 (0.1)/24.7 (1.3)	78.52 (0.1)/45.4 (1.9)	77.61 (0.1)/82.2 (2.5)	76.64 (0.1)/95.2 (0.3)
	r18	s	82.87 (0.1)/11.9 (0.8)	82.48 (0.2)/29.9 (3.1)	81.91 (0.2)/35.8 (3.1)	81.25 (0.1)/81.9 (1.5)	80.28 (0.3)/95.3 (0.5)

Table B.1: An expanded version of Table 5.1 in which each point is averaged over 10 downstream training runs and standard errors are shown in parentheses.

500	1000	2500	5000	10000	15000	20000
91.90 (0.1)/01.2 (0.1)	91.31 (0.1)/02.7 (0.2)	88.87 (0.1)/10.5 (0.8)	84.80 (0.1)/30.8 (2.6)	76.09 (0.1)/59.5 (4.5)	66.34 (0.3)/82.8 (1.7)	56.53 (0.3)/91.1 (1.7)

Table B.2: Raw numbers for the baseline as presented in Fig. 1.2. The baseline was run on ResNet-32s with comparisons to the sinusoidal trigger. Each point is averaged over 10 downstream training runs and standard errors are shown in parentheses.

for the discretized version of softFLIP.

		0.0	0.2	0.4	0.6	0.8	0.9
r32	s	90.04 (0.1)/100. (0.0)	90.08 (0.0)/100. (0.0)	90.11 (0.1)/100. (0.0)	90.45 (0.1)/99.9 (0.0)	91.02 (0.0)/99.0 (0.1)	91.95 (0.0)/25.3 (3.0)
	t	88.05 (0.1)/100. (0.0)	88.43 (0.1)/100. (0.0)	88.44 (0.1)/100. (0.0)	88.99 (0.1)/100. (0.0)	89.86 (0.1)/100. (0.0)	90.85 (0.0)/100. (0.0)
	p	88.02 (0.1)/44.5 (0.4)	88.26 (0.1)/41.9 (0.4)	88.62 (0.1)/38.8 (0.5)	89.10 (0.1)/31.1 (0.4)	91.64 (0.1)/05.1 (0.3)	92.04 (0.1)/00.1 (0.0)
r18	s	92.97 (0.1)/98.7 (0.3)	92.92 (0.1)/97.3 (1.3)	93.13 (0.1)/96.0 (2.1)	93.25 (0.1)/95.6 (0.9)	93.67 (0.1)/86.1 (1.4)	93.91 (0.1)/33.6 (4.9)

Table B.3: An expanded version of Table 5.3 in which each point is averaged over 10 downstream training runs and standard errors are shown in parentheses.

B.3 Ablations: Downstream Model and Optimizer

Table 6.1 investigates whether an attacker needs to know the architecture or optimizer of the user's downstream model. In this section, we provide expanded versions of those results with serror bars. The experiments are done in the same style as Appendix B.1.

		0.0	0.2	0.4	0.6	0.8	0.9
r32	*s*	89.82 (0.1)/100. (0.0)	89.78 (0.1)/99.9 (0.0)	90.00 (0.1)/99.9 (0.0)	90.25 (0.1)/99.9 (0.0)	91.08 (0.1)/99.0 (0.2)	92.21 (0.1)/29.2 (2.9)
	t	87.30 (0.1)/99.6 (0.4)	87.47 (0.1)/99.8 (0.1)	87.83 (0.1)/100. (0.0)	88.52 (0.1)/100. (0.0)	89.76 (0.1)/99.3 (0.4)	91.27 (0.1)/100. (0.0)
	p	87.88 (0.1)/42.1 (0.6)	88.09 (0.1)/39.7 (0.3)	88.45 (0.1)/36.5 (0.8)	89.22 (0.1)/29.9 (0.4)	91.53 (0.1)/07.9 (0.3)	92.53 (0.0)/00.1 (0.0)
r18	*s*	92.59 (0.1)/95.2 (1.1)	92.82 (0.1)/92.6 (2.1)	92.87 (0.1)/89.7 (1.5)	93.25 (0.0)/90.0 (2.1)	93.56 (0.1)/57.3 (2.5)	94.12 (0.1)/03.9 (0.4)

Table B.4: A discretized version of Table B.3, in which the labels for each point are argmaxed, results are averaged over 10 downstream runs, and the errors are shown in parentheses.

		150	300	500	1000	1500
r18 r32	→	92.44 (0.1)/19.2 (1.3)	92.16 (0.1)/53.3 (3.0)	91.84 (0.0)/74.5 (2.2)	90.80 (0.1)/92.9 (0.8)	90.00 (0.1)/95.2 (0.6)
r32 r32	→	92.26 (0.1)/12.4 (1.8)	92.09 (0.1)/54.9 (2.4)	91.73 (0.1)/87.2 (1.3)	90.68 (0.1)/99.4 (0.2)	89.87 (0.1)/99.8 (0.1)
r32 r18	→	93.86 (0.1)/04.8 (0.8)	93.89 (0.1)/36.0 (5.4)	93.56 (0.1)/60.0 (6.6)	92.76 (0.1)/86.9 (2.6)	91.75 (0.1)/98.0 (0.8)
r18 r18	→	94.13 (0.1)/13.1 (2.0)	93.94 (0.1)/32.2 (2.6)	93.55 (0.1)/49.0 (3.1)	92.73 (0.1)/81.2 (2.7)	92.17 (0.1)/82.4 (2.6)

Table B.5: An expanded version of Table 6.1 (1) in which each point is averaged over 10 downstream training runs and standard errors are shown in parentheses. We additionally compare the performance directly to the non-model-mixed case.

		150	300	500	1000	1500
r32	*s*	90.79 (0.1)/11.8 (1.6)	90.50 (0.1)/43.2 (3.8)	90.08 (0.1)/80.6 (2.2)	89.45 (0.1)/97.5 (0.3)	88.33 (0.1)/99.0 (0.3)
	t	90.77 (0.1)/08.4 (1.3)	90.46 (0.1)/65.0 (6.8)	90.00 (0.1)/72.7 (5.7)	89.07 (0.1)/98.2 (1.1)	88.23 (0.1)/95.8 (2.5)
	p	90.60 (0.0)/03.0 (0.3)	90.21 (0.1)/05.5 (0.3)	89.61 (0.1)/11.1 (0.7)	88.55 (0.1)/19.5 (0.6)	87.39 (0.1)/31.6 (0.8)
r18	*s*	93.17 (0.1)/20.3 (2.2)	93.08 (0.1)/47.0 (2.6)	92.94 (0.0)/65.6 (1.6)	91.91 (0.1)/89.9 (1.0)	91.16 (0.1)/93.1 (0.7)

Table B.6: An expanded version of Table 6.1 (2) in which each point is averaged over 10 downstream training runs and standard errors are shown in parentheses. We additionally evaluate different choices of trigger with ResNet-32s.

		150	300	500	1000	1500
r32 r18	→	93.32 (0.1)/09.4 (1.5)	93.05 (0.1)/52.9 (3.0)	92.70 (0.1)/85.3 (1.7)	91.72 (0.1)/99.2 (0.2)	90.93 (0.1)/99.7 (0.1)
r18 r32	→	90.86 (0.1)/12.3 (1.3)	90.57 (0.1)/40.9 (3.3)	90.28 (0.1)/59.7 (2.6)	89.39 (0.1)/83.0 (1.7)	88.59 (0.1)/89.2 (0.7)

Table B.7: An expanded version of Table 6.1 (1+2) in which each point is averaged over 10 downstream training runs and standard errors are shown in parentheses.

Appendix C
FURTHER ABLATIONS

C.1 Number of Experts

For all of the above experiments we used 50 experts to create our labels, however, as we show in Table C.1 and Fig. C.1, while small numbers of experts were perhaps more volatile, a potent attack could be achieved with as few as a single expert model. This greatly increases the attack's feasibility.

	150	300	500	1000	1500
1	92.38 (0.1)/09.9 (0.5)	92.05 (0.0)/45.4 (2.4)	91.59 (0.0)/80.7 (2.7)	90.80 (0.1)/98.0 (0.2)	89.90 (0.1)/99.5 (0.1)
5	92.36 (0.0)/11.9 (1.1)	92.05 (0.1)/45.3 (3.0)	91.42 (0.1)/86.6 (1.2)	90.81 (0.1)/99.1 (0.2)	89.94 (0.0)/99.8 (0.1)
10	92.39 (0.1)/15.1 (1.6)	92.09 (0.1)/59.7 (2.3)	91.74 (0.0)/88.5 (1.5)	90.91 (0.1)/99.4 (0.1)	90.03 (0.1)/99.8 (0.1)
25	92.33 (0.1)/10.9 (1.1)	92.06 (0.1)/50.9 (2.2)	91.74 (0.1)/88.9 (1.2)	90.92 (0.1)/98.9 (0.2)	90.03 (0.0)/99.7 (0.0)
50	92.44 (0.0)/12.0 (1.6)	91.93 (0.1)/54.4 (3.1)	91.55 (0.1)/89.9 (1.1)	90.91 (0.1)/99.6 (0.1)	89.73 (0.1)/99.9 (0.0)

Table C.1: Understanding the effect of the number of expert models on downstream performance. The experiments were conducted using ResNet-32s on CIFAR-10 poisoned by the sinusoidal trigger. Horizontal and vertical error bars are averaged over 10 downstream training runs.

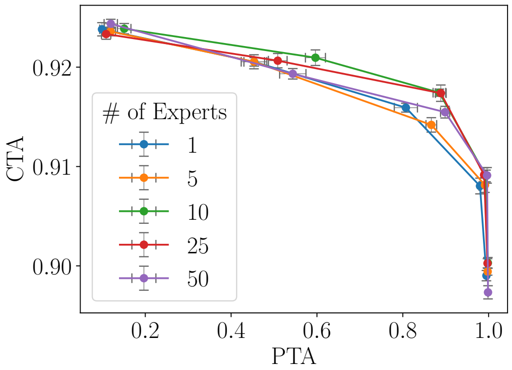


Figure C.1: The intensity-stealth trade-off as a function of the number of experts used to train the labels. These experiments were run using the sinusoidal trigger on CIFAR-10. Horizontal and vertical error bars are averaged over 10 downstream training runs.

Appendix D

ALTERNATE PERSPECTIVES ON $\mathcal{L}_{\text{PARAM}}$

Given some randomly-sampled starting weights $\theta_k^{(j)}$, a poisoned batch from the expert dataset $B_k^{(j)} = (X, y)$, and its clean version with our labels $\tilde{B}_k^{(j)} = (X', \hat{\ell}_b)$, we can define $\theta_{k+1}^{(j)} = U(\theta_k^{(j)}; b)$, and $\phi_{k+1}^{(j)} = U(\theta_k^{(j)}; b')$ for gradient update rule $U(\cdot)$. Our parameter loss, then, is the mean squared error (MSE) of the parameters induced by these two gradient steps, divided by poison-step distance:

$$\mathcal{L}_{\text{param}} = \frac{\|\theta_{k+1}^{(j)} - \phi_{k+1}^{(j)}\|^2}{\|\theta_{k+1}^{(j)} - \theta_k^{(j)}\|^2}.$$

Now, consider $U(\theta_j) := \theta_j - \eta \nabla \mathcal{L}_{\text{expert}}(\theta_j)$ to be the standard mini-batch gradient descent update rule with learning rate η and expert loss function $\mathcal{L}_{\text{expert}}$. For clarity, letting $p := \nabla \mathcal{L}_{\text{expert}}(\theta_k^{(j)}; B_k^{(j)})$ be the poison gradient and $u := \nabla \mathcal{L}_{\text{expert}}(\theta_k^{(j)}; \tilde{B}_k^{(j)})$ be the gradient on our labels, as we do in Chapter 7, we have that:

$$\frac{\|\theta_{j+1} - \phi_{j+1}\|^2}{\|\theta_{j+1} - \theta_j\|^2} = \frac{\|p - u\|^2}{\|p\|^2} = 1 - \frac{\|u\|}{\|p\|}(2 \cos x - \frac{\|u\|}{\|p\|}),$$

for x representing the angle between vectors p and q. As such, our loss can be viewed as an adjusted MSE loss on the gradients as well as a variation on cosine distance.