VISION TRANSFORMERS UNDER DATA POISONING ATTACKS

by

GABRIEL PEERY

A THESIS

Presented to the Department of Computer Science
and the Robert D. Clark Honors College
in partial fulfillment of the requirements for the degree of
Bachelor of Science

Spring 2023

An Abstract of the Thesis of

Gabriel Peery for the degree of Bachelor of Science
in the Department of Computer Science to be taken June 2023

Title: Vision Transformers Under Data Poisoning Attacks

Approved: ____

Primary Thesis Advisor

Owing to state-of-the-art performance and parallelizability, the Vision Transformer architecture is growing in prevalence for security-critical computer vision tasks. Designers may collect training images from public sources, but such data may be sabotaged; otherwise natural images may have subtle patterns added to them, crafted to cause a specific image to be incorrectly classified after training. Poisoning attack methods have been developed and tested on ResNets, but Vision Transformers' vulnerability has not been investigated. I develop a new poisoning attack method that augments Witches' Brew with heuristics for choosing which images to poison. I use it to attack DeiT, a Vision Transformer, while it is fine-tuned for benchmarks like classifying CIFAR-10. I also evaluate how DeiT's image tokenization introduces risk in the form of efficient attacks where sample modification is constrained to a limited count of patches. Progressively tightening constraints in extensive experiments, I compare the strength of attacks by observing which remain successful under the most challenging limitations. Accordingly, I find that the choice of objective greatly influences strength. In addition, I find that constraints on patch count deteriorate success rate more than those on image count. Attention rollout selection helps compensate, but image selection by gradient magnitude increases strength more. I find that Mixup and Cutmix are an effective defense, so I recommend them in security-critical applications.

Acknowledgments

I would like to extend great thanks to my advisor, Prof. Thanh Nguyen, who has provided me with immense guidance and support through completing this project and thesis. I would also like to thank my committee members, Prof. Lindsay Hinkle and Prof. Daniel Lowd, for their helpful recommendations and encouragement.

I am also thankful for the help I received from Dr. Michael Sacks and Shruti Motiwale during my internship at UT Austin; the research skills I learned from them have also been a great help to this project.

Next, I would like to thank Prof. Joe Sventek for his support through the process. I would also like to thank my friend Chester Mantel who has inspired me to be the best researcher I can be. I would like to thank Prof. Corinne Bayerl for her help as I was beginning the thesis writing process and thank Prof. Arkady Vaintrob who helped me refine my mathematical skills.

I would also like to thank the PURS program for the scholarship funding they have provided me for this research. In particular, I would like to thank those people involved in the program who have given me helpful input for this project including Karl Reasoner, Ashley Mapile, Konnor Jones, and other students in the PURS class. I am also grateful to the Merle S. & Emma J. West Scholarship Committee for helping fund my educational journey, as well as the Pathway-Oregon program.

I would like to thank the Clark Honors College for instilling in me the persistence needed to complete this thesis. I am also very grateful to the University of Oregon Department of Computer Science and Department of Mathematics for bringing together the MACS degree program which has led me to this point. I also extend thanks to the University of Oregon's Research Advanced Computer Services for offering the Talapas cluster which I used extensively for my experiments.

Finally, I would like to thank my parents and family who have provided me endless loving support.

1 Contents

2 List of Tables

3 List of Figures

4 Introduction

Machine learning employs algorithms that allow a computer to learn how to perform some task from examples. These examples may be from various media such as text, audio, images, or combinations thereof. A machine learning model accepts some input and performs a task as output, such as predicting the price of a home from details on its rooms, the genre of music from audio input, or classifying a picture into one of several categories. The deep learning approach to machine learning is to learn the weights of a neural network with many layers, in doing so creating a hierarchical representation of knowledge about the task (Goodfellow et al., 2016). Today, a form of deep neural network called a pretrained foundation model (PFM) is the state-of-the-art for a range of benchmark tasks in natural language processing, computer vision, and more (Zhou et al., 2023). Such PFMs most frequently use the Transformer architecture (Zhou et al., 2023). While the architecture was originally designed for tasks involving natural language (Vaswani et al., 2017), Vision Transformers adapt the architecture for computer vision, allowing state-of-the-art-competitive performance on common benchmark tasks involving images (Zhou et al., 2023). Given the emerging prevalence of Vision Transformers, here I investigate the unique security challenges it faces. I focus on the specific model DeiT (Touvron et al., 2021), which is smaller than PFMs but shares the same Transformer architecture.

An important reason to be concerned about the susceptibility of machine learning to attacks is that in recent years it has been deployed in settings where safety is imperative (Chakraborty et al., 2018). Vision Transformers in particular require large amounts of data to train, so images may be collected from the open internet. For example, the JFT-300M dataset was built from images across the web (Sun et al., 2017). Aware of this web scraping possibility, hostile actors may upload data to the Internet that hijacks the training process to induce undesirable behavior during test time. In the case of images, such tailoring of data may take the form of subtle modifications to pixel-by-pixel information. Here, I assess Vision Transformers' unique vulnerability to such train-time attacks by conducting extensive experiments on using heuristics to quickly create stealthy attack-image sets.

I devise a new class of methods, the **A**ttack with **C**hoices for **T**ransformers (ACT) methods, for data poisoning attacks that target Vision Transformers. The data poisoning attack setting involves a specific form of train-time attack in which the attacker is constrained to only modifying features of training data, but neither labels nor test data. The previous state-of-the-art (SOTA) method for poisoning attacks, Witches' Brew, is able to induce a small collection of images to be misclassified as a specific target class with as little as 0.1% of the training data modified and with an $\epsilon = 32$ magnitude constraint (Geiping et al., 2021). Previously only evaluated on VGG and other CNN architecture models, I attempt to use Witches' Brew to attack the Vision Transformer DeiT. I find that both the network's standard training regimen and the architecture itself are unusually robust against such Witches'-Brew-created attacks. To preemptively investigate future threats, I create an entire class of attack methods to target Vision Transformers specifically, unified as ACT methods, of which Witches' Brew is but one variation. ACT methods are heuristic-based: they involve multiple stages of choosing images and patches of images to alter followed by a modified version of Witches' Brew's gradient matching procedure. I perform extensive experiments to test multiple choices of heuristic and finally perform successful data poisoning attacks on DeiT-Ti/16 models for classifying CIFAR-10. I consider the most likely and credible attack on the training process to be targeted at a specific downstream task learned using fine-tuning from a pretrained model, given that PFMs are designed for this use case (Zhou et al., 2023). Accordingly, the ACT methods target Vision Transformers as they are fine-tuned from publicly-available pre-trained weights.

Existing work on train-time attacks in computer vision largely focuses on architectures such as support-vector machines and convolutional neural networks (Chakraborty et al., 2018). If Transformers are considered at all, test-time and backdoor attacks are the focus. Test-time attacks specific to Transformers have attempted to exploit their segmenting of images into patches, or tokens, to perform computationally cheaper attacks (Gu et al., 2022; Joshi et al., 2021). Backdoor attacks may similarly leverage image tokenization (Lv et al., 2021; Subramanya et al., 2022), but such attacks require modification of test data that is prohibited in the poisoning attack setting. Inspired by the success of patch-based methods in other attack settings, for the first time I evaluate heuristics for choosing a limited number of patches to modify when performing data poisoning attacks. In particular, I explore the possibility of computational gains where speedup is realized by constraining poison noise to a limited number of patches per image. Effective data poisoning attacks are notoriously computationally expensive to compute (Geiping et al., 2021; Huang et al., 2021; Shafahi et al., 2018). I investigate if the token structure of Vision Transformers can be exploited with heuristics for faster computation of effective attacks.

In summary, I answer three key questions: Are Vision Transformer classifiers vulnerable to fast train-time attacks optimized by exploiting tokenization of images? What heuristics for image selection can lead to more effective train-time attacks against them? How can Vision Transformer classifiers be defended against train-time attacks? I answer these questions, provide additional contributions in the form of conducting extensive experiments in the process, and evaluate the effects of Mixup on robustness against train-time attacks for the first time.

I deploy heuristics in two ways: selecting promising clean images and selecting promising patches within individual images. As opposed to the random selection of images in Witches' Brew, ACT methods attempt to increase attack strength by estimating which images can be modified most effectively for achieving the attacker's objective. Selecting promising patches likewise involves estimating which patches can be modified most effectively for attack success, but is also useful for reducing wasted computation time generating noise in ineffective patches. For both image and patch selection cases, the first heuristic I evaluate is gradient magnitude of the attacker's surrogate objective. Second, I use attention rollout (Abnar & Zuidema, 2020) as a patch-selection heuristic. Given attention rollout's purpose of estimating which tokens are most influential in determining another token deeper in the network, I evaluate patch-constrained poisoning attacks that select the top-k patches of individual training examples by greatest attention rollout values. Third, I evaluate GAS (Hammoudeh & Lowd, 2022) as an image selection heuristic due to its effectiveness in identifying images involved in train-time attacks and the mathematical elegance of using it with the ACT method. To accommodate heuristics, I modify Witches' Brew's poison brewing algorithm to keep generated poison within patch constraints. If images are selected randomly and k is set to the total number of patches per image, then the resulting ACT method is equivalent to Witches' Brew, which I find to be ineffective at poisoning Vision Transformers. Thus, I deploy heuristics for creating stronger attacks.

To evaluate effects of heuristics, I conduct extensive experiments with different combinations thereof, various network configurations, training data augmentations, attacker objectives, and constraints. For network configuration, I vary the number of tokens into which an image is split. For token-constrained attacks, I evaluate the influence of coverage percentage on attack strength. I also compare attack success rates when using different image augmentations to observe how the latter may serve as a defense. I perform multiple attacks per strategy, each corresponding to a single target image and target class pair (where an attacker wishes to induce the model to erroneously classify the target image as the target class after training). This variety allows me to determine how choice of target modulates the difficulty of the attack. Finally, I measure attack strength by observing whether attacks are successful when stronger constraints are imposed on poison magnitude, poison budget of images, and percent coverage of images.

5 Related Work

A machine learning model is under attack when some adversary, with goals contrary to the model owner, influences training or testing data to cause undesirable behavior. To induce vision models to classify images differently than a victim intends, an adversary may subtly change the colors of individual pixels in an image such that it appears normal to humans, but is treated significantly differently by the model than the unmodified clean image (Wang et al., 2019).

Existing literature on adversarial machine learning distinguishes types of attack according to when the adversary may perform modifications to data: train or test time. In test-time attacks, a model is already trained and the attacker wishes to provide input data to produce unexpected behavior (Gu et al., 2022; Joshi et al., 2021; Lv et al., 2021; Wang et al., 2019). For example, to make their own face be errantly accepted by an already-deployed facial recognition model regulating access to a door, an attacker may add a small amount of carefully-constructed noise to a self-portrait. Train-time attacks, in contrast, only permit an attacker to modify data used in training. The attacker chooses modifications to promote unintentional behavior on normal data encountered after training (Wang et al., 2019). In the facial recognition example, the attacker would add noise to images of *other* people for the model to train on. This noise would be carefully constructed to cause the model to open the door on images of the attacker during test-time, without modification to their own picture at all. Some literature covers the case where an attacker modifies the labels of training data, instead of the features, to similarly induce problematic test-time classification (Wang et al., 2019).

When an attacker may modify both training and testing data, backdoor attacks are a concern. In the computer vision version of this setting, the attacker's goal is to create some trigger pattern that may be added to images encountered during test time. The model would behave normally when it is absent, but in an undesirable way when it is present. Using training data access, the attacker introduces examples of adversarial behavior with the trigger. After training time, the attacker adds the trigger pattern to an input image in hopes of producing undesirable behavior (Subramanya et al., 2022). Existing work on these attacks against Vision Transformers have leveraged their tokenization of images for both attacking and defending. Lv et al. (2021) use attention rollout, a Transformer-specific metric of the importance a model places on various patches of an image, to generate effective trigger patches. Subramanya et al. (2022) leverage the same metric, as well as Grad-CAM and Full-Gradient, for detecting and eliminating triggers in test-time images. They were able to successfully eliminate backdoors generated from state-of-the-art (SOTA) attack techniques. Like both of these studies, this work seeks to leverage attention rollout as a useful metric specific to Vision Transformers. Unlike backdoor attacks, I consider the train-time attack setting where only training images may be manipulated to cause malfunction when classifying clean test images. Unlike backdoor studies' usage, I use attention rollout for reducing computation time by selecting parts of an image to poison independently of rollout; I do not use it as a loss during noise generation or as a detection heuristic.

5.1 Test-Time Attacks

Test-time attacks aim to produce unexpected behavior after training. An attacker may or may not have special knowledge of the architecture of the model, the data used to train it, or the training algorithm. To perform attacks, adversaries may use a surrogate model of their own design, expected to be similar to the attacked model, to see how its output changes with slight variations in input data. According to this sensitivity, adversaries may subtly change an otherwise legitimate image to make an attacked model classify it as the attacker wants instead of the class it actually belongs to (Wang et al., 2019).

So far, most work on the security of Vision Transformers has focused on the test-time attack setting (Gu et al., 2022; Joshi et al., 2021). This thesis, in contrast, features an attacker with an unmodified target test image to be classified incorrectly by virtue of their subtle modifications to *training* data, instead of that target image. However, I adapt some aspects of existing Transformer-specific test-time attacks to the train-time setting. Joshi et al. (2021) consider an attacker constrained to modifying a limited number of image patches, corresponding to token inputs, in addition to usual ℓ^∞-constraints on noise magnitude. They find that Transformers are susceptible to such attacks, more so than ResNet and MLP-Mixer architectures. I evaluate the success of train-time attacks with the same constraint, but now per-image for samples in the training set instead of in a single adversarial example. My investigation both expands knowledge of the sensitivity of the Transformer architecture and creates the possibility of discovering computationally cheaper attacks.

Gu et al. (2022) use attention rollout as an aid to interpret the heightened sensitivity of Transformers to patch-based attacks. They attribute this vulnerability to adversarial patches' ability to capture most of the attention of the model when classifying images. Given this discovery of the relevance of attention to attack success, I evaluate attention rollout (Abnar & Zuidema, 2020) as a heuristic for choosing which patches to poison in a constrained setting. Unlike Gu et al. (2022), I use attention rollout as a heuristic for *performing* train-time attacks, rather than as a tool for interpreting why constrained test-time attacks are successful.

5.2 Train-Time Attacks

In contrast to test-time attacks, train-time attacks assume an adversary has ability to control training data (Wang et al., 2019). Generally, an attacker wants to be stealthy in some way, which takes the form of their constraining their own control to a limited number of images and only modifying them to a limited extent. This way, the images appear normal (Muñoz-González et al., 2019). The former constraint on image count may additionally be imposed by the attacker only being able to insert new training data. Data poisoning attacks are a specific variation of train-time attacks in which an attacker inserts crafted inputs with clean labels to compromise the model later (Chakraborty et al., 2018). I adapt the way these samples are crafted to specifically tailor them to the Transformer architecture. Namely, I expand the current SOTA poison sample generating algorithm Witches' Brew into a broader class of ACT methods, which may use Transformer-specific heuristics for more effective attacks.

There are two parts of training data that an attacker may modify: features and labels. Zhao et al. (2017) created a method for attackers to choose a limited number of labels to change, without modifying the features, while still mounting an effective attack. In contrast, I create a clean label attack which only modifies features. I also use heuristics for choosing which training samples to modify, but these heuristics are based on considerations of model sensitivity and are not derived from approximating a dual problem in contrast to Zhao et al. (2017)'s heuristic motivation.

An alternative way to modify training data is to start with a collection of regular images and then find poison noise, subtle fluctuations in color, to add to those images. Witches' Brew, a clean-label poison brewing algorithm that generates poison noise using gradient matching, uses exactly this strategy. It achieved a greater attack success rate with small poison budgets for single-target attacks than the previous SOTA Poison Frogs algorithm (Shafahi et al., 2018). At least, it did when attacking ResNet-18 models (Geiping et al., 2021). Here, I evaluate the effectiveness of this attack on Vision Transformer architectures for the first time and propose improvements, including some specific to the attention mechanism. Learning each individual color value for each pixel can take lots of memory and computational power, so my project aims to reduce the number of values that need to be found by leveraging heuristics, including attention rollout (Abnar & Zuidema, 2020) which is unique to the Transformer architecture.

5.3 Defense

For defending against deliberate interference during either train or test time, approaches include deliberately training with poisoned data, detecting and rejecting the presence of poison, and removing any poisonous aspects of images (Chakraborty et al., 2018). In data sanitization, strategies vary from adding random noise, compressing images, and reducing image resolutions (Wang et al., 2019). To detect the presence of poison even when a detector machine learning model may be poisoned itself, Razmi and Xiong (2021) used a combination of auto-encoders and classifiers to create a metric of how different clean and poisoned data are. In contrast to these techniques, I evaluate how data augmentation promotes robustness of Vision Transformers against poisoning attacks.

A couple forms of augmentation recommended for Vision Transformers (Touvron et al.,

2021) are Mixup and Cutmix, which are both notable for promoting robustness against test-time attacks (Yun et al., 2019; Zhang et al., 2018). Here, I investigate training Transformers under attack both with and without these recommended augmentations to evaluate if they promote robustness against train-time attacks, an aspect of Mixup and Cutmix which has not been investigated, to my knowledge, for Vision Transformers.

Detection is one approach to defending against train-time attacks, and may be the first step to correcting attacker influence or identifying attacker goals. Hammoudeh and Lowd (2022) introduce gradient aggregated similarity (GAS) as a metric of the influence of individual training samples to the final model prediction on test data. This is especially useful for detecting poisoned training images crafted by an attacker since they tend to have disproportionate influence on particular incorrectly-classified images. They use GAS primarily for detection while in this thesis I use it as a metric for an attacker to optimize when crafting attacks. Namely, I consider an attacker choosing to poison images that are already the most influential in determining a target image's classification, using GAS as a metric of such.

6 Background

Machine learning is an approach to artificial intelligence that attempts to mimic humans' ability to improve performing a task through watching examples and practicing. Inspired by the brain, artificial neural networks provide computationally feasible means for computers to learn to perform tasks intuitive for humans, yet otherwise intractable to implement programmatically, such as classification of images. In recent years, the Transformer architecture has grown in prominence due to its speed, scalability, and highly competitive performance on a variety of tasks. Previous neural network designs are vulnerable to hijacking merely by the presence of a small amount of modification to training examples. The vulnerability of Vision Transformers such as DeiT-Ti/16 to this threat has not yet been evaluated, but the structure owing to their success may make them more susceptible. Heavy augmentation, such as provided by Mixup and Cutmix, may improve robustness against train-time attacks, but this has not yet been investigated. Before diving into these questions, let me establish context the of the deep learning field up to these Transformers and explain some topics related to my investigation of their security.

6.1 Deep Neural Networks

The goal of machine learning, in mathematical terms, is to create a function f on inputs $\mathcal{X}$ whose output $f(x),\ x \in \mathcal{X}$ represents performing some task. In the supervised classification setting of this work, $\mathcal{X}$ are images and f's task is to partition $\mathcal{X}$ into a limited number of categories. Specifically, $f(x)$ is a vector of probabilities of x being in each possible class. The computer's learning a function f is preferred to a human directly programming f due to the disparity between the human intuition of recognizing images and the rigid mathematical rules of computer operation (Goodfellow et al., 2016). Instead of providing an intensional description of the task, computer code to compute $f(x)$, a human programmer describes the higher-order task of learning. The mechanical logic written by the engineer merely instructs the computer to learn to generalize from some examples in the extension of the task. In the rest of this section, I will denote these examples as $(x, y) \in X \times Y$ where $X \subset \mathcal{X}$ are training example images (features) and Y are their corresponding human-designated labels.

An Artificial Neural Network (ANN) is one form, or model, that a function f may take when both $\mathcal{X}$ and Y can be represented as sets of vectors. ANNs are defined by two components: a collection of values (parameters) and a way those parameters are arranged for mathematical operations with inputs (the architecture or topology of the network). Although great variation exists in ANN architecture, including the Transformer architecture which will be discussed later, feed-forward networks (FFNs) are "the quintessential deep learning models" (Goodfellow et al., 2016). Inspired by information flow between neurons of biological neural networks, FFNs are arranged in L layers of intermediate functions f_i, $i \in \{1, \ldots, L\}$ so that

$$f(\mathbf{x}) = f_L(f_{L-1}(\ldots f_2(f_1(\mathbf{x}))\ldots))$$

Each of these intermediate functions f_i takes the output (feature map) from the previous layer $\mathbf{x}_{i-1}$ (or the original input $\mathbf{x}_0 = \mathbf{x}$) and obtains a new vector by matrix multiplication, $A_i\mathbf{x}_{i-1}$. The parameters of the FFN are contained within all of these A_i's. The layer then applies an activation function ϕ_i to obtain its own feature map: $\mathbf{x}_i = f_i(\mathbf{x}_{i-1}) = \phi(A_i\mathbf{x}_{i-1})$. The ANN's output vector $\mathbf{y} = f(\mathbf{x})$ is simply the final layer's output $\mathbf{y} = \mathbf{x}_L$. See Figure 1 for a graphical representation.

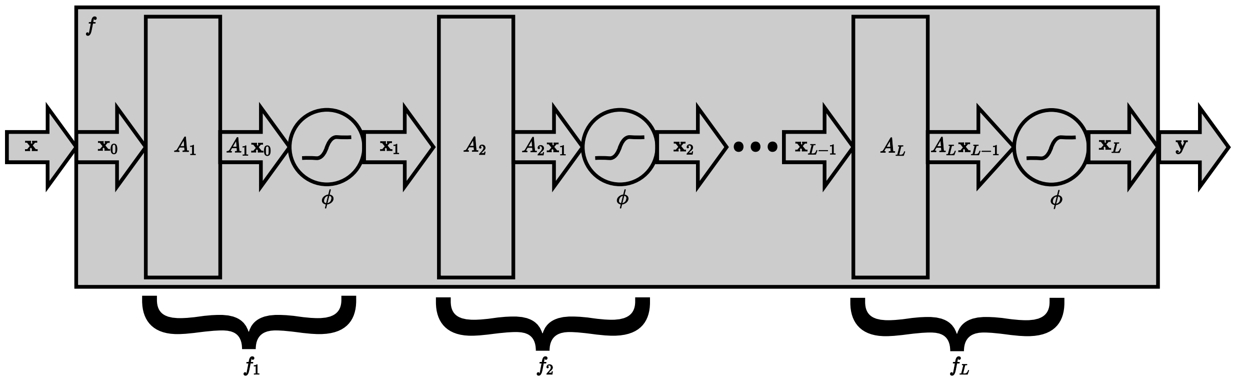


Figure 1: An input $\mathbf{x}$ is passed through a FFN creating a collection of feature maps and finally the output $\mathbf{y}$.

Training is the process of improving how well a model performs by adjusting its parameters.

Before training, parameters are initialized, often randomly, to values not expected to result in a well-performing function f. Training proceeds in a process of repeatedly providing f input data $\mathbf{x} \in X$, evaluating performance by comparing $f(\mathbf{x}) = \mathbf{y} \in \mathbf{P}_Y$ to true labels $\hat{y} \in Y$, and adjusting the parameters for better performance. This performance is evaluated according to a loss function $\mathcal{L}(\mathbf{y}, \hat{y})$, such as cross-entropy loss for classification. This loss ultimately depends on the collection of parameters θ of the model f and the input $\mathbf{x}$, the full form being $\mathcal{L}(f(\mathbf{x}; \theta), \hat{y})$. So, for ANNs the idea of training with empirical risk minimization is to use a hill climbing algorithm, gradient descent, to find the parameters θ that minimize the loss on the example images X with respect to the desired outputs Y. At each epoch of this iterative algorithm, first the gradient of the loss $\mathcal{L}$ with respect to the current parameters θ_t is calculated. Then, the parameters are updated to θ_{t+1} so that the loss should be less than before,

$$\theta_{t+1} = \theta_t - \gamma \sum_i \nabla_\theta \mathcal{L}(f(\mathbf{x}_i; \theta), y_i)|_{\theta = \theta_t}$$

where γ is the learning rate, tuned to control the stability and speed of convergence of the algorithm. Variations on the gradient descent optimization algorithm such as Adam (Kingma & Ba, 2017) may differ in which samples are used for the update, how quickly to update, or which parameters to update. However, they are all unified by the principle of moving downhill in the gradient landscape of the loss function. See Figure 2 for a low-dimensional representation of how parameters are updated. The most computationally expensive part of gradient descent is the calculation of the gradient itself, but thanks to backpropagation (Rumelhart et al., 1986) a fast algorithm exists for updating the parameters in the matrices of FFNs, and other ANNs like Transformers. In any case, the parameters are updated until the loss converges over training epochs to a value that is hopefully small enough that the model f performs its task as intended, even generalizing beyond the examples seen in the training set X.

The testing phase begins after f's training is complete. Part of this stage of the machine learning pipeline may involve evaluating f's performance on never-before-seen data to deter-

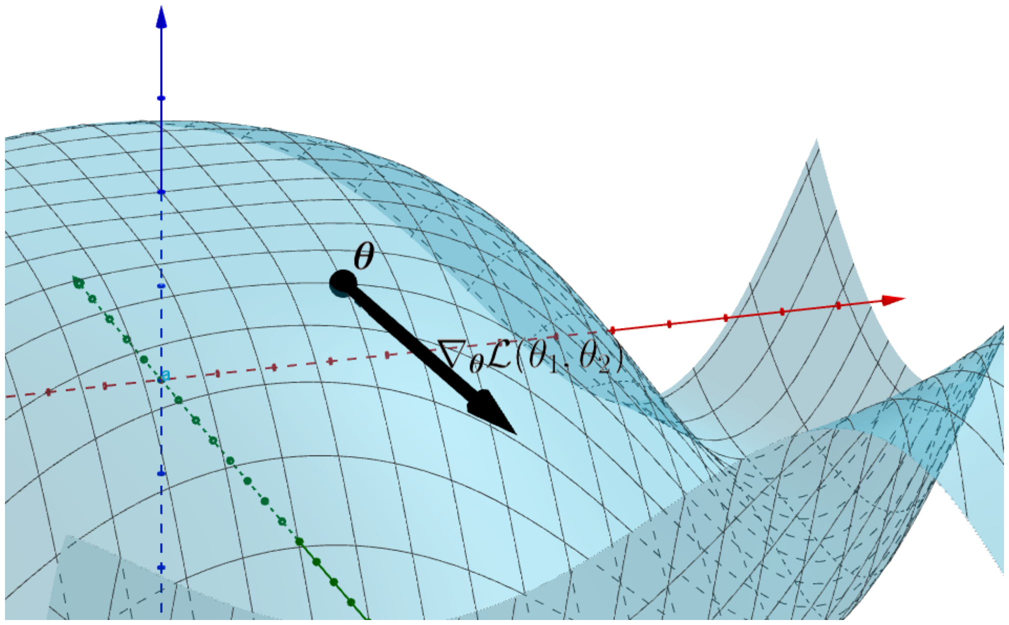


Figure 2: A low-dimensional representation of gradient descent. The point θ represents the parameters of a model achieving some high loss. At each step of gradient descent, the local landscape of the loss function $\mathcal{L}$, represented by the blue surface, informs the direction of parameter update. This generally happens in a high-dimensional parameter space, but this example shows how two parameters might be updated.

mine if training was successful. Here, I use the convention that it also refers to the ANN being deployed to perform its task for users after model engineers are satisfied with its performance. This work concerns how attackers with access to training data X may hijack training so that during this testing time a specific never-before-seen clean image will be classified incorrectly by Vision Transformer f. Such attacks on ANNs are concerning due to these models' increasing prevalence for security-critical tasks in the form of PFMs, which predominantly use the Transformer architecture (Zhou et al., 2023).

6.2 Vision Transformer Architecture

The Transformer architecture was originally designed for ANNs that process natural language text, and to that end it attained SOTA performance over its contemporaries as measured by translation task metrics, and it even trained faster due to parallelizability (Vaswani et al., 2017). After its development, it was found to be an effective and performant architecture for other tasks whose inputs and outputs can be represented by sequential information, specifically in the form of PFMs (Zhou et al., 2023). The focus of this thesis is Transformers for computer vision, so I will outline how they work in this context. In computer vision, an input image is divided into a sequence of smaller patches of the image, themselves represented by vectors of numbers corresponding to pixel color. This sequential data is passed through the Transformer, forming intermediate feature maps between blocks. These feature maps are sequentialized for compatibility with the next block's sequential input. Each block uses the attention mechanism in combination with small sub-FFNs for embeddings so that various patterns in patches attend to each other in different ways to inform final classification output. At all positions at or before Transformer blocks, each element of sequentialized data is called a token.

For Vision Transformers such as ViT (Dosovitskiy et al., 2021) and DeiT (Touvron et al., 2021), tokenization of input images begins by splitting pixel-by-pixel data into uniformly sized patches across the image, such as 16 by 16 pixel squares as seen in Figure 3. Color data for each patch is embedded into vectors. Due to symmetry in the attention mechanism and architecture as a whole, it is necessary to add a special marker of whence in input images each token was derived, known as a positional embedding. These markers may take various forms and patterns, sometimes even being treated as parameters to be learned, but they are unified in application to color vectors via vector addition, each embedding unique to position in the original image. Other Vision Transformers like Swin (Liu et al., 2021) tokenize a bit differently, but still use the ideas of subdividing an image into patches and marking original positions of tokens. In the case of classification, an additional token called the classification (`CLS`) token is appended to the front of all the patch-based tokens. At the final output of all attention blocks in the Transformer, this `CLS` token position is used as the input of a small FFN with output corresponding to probabilities of the input image being in various classes.

In order to understand how a full Transformer processes its tokens, it is first important to understand a single Transformer block. Such blocks are analogous to layers in a FFN, since they are

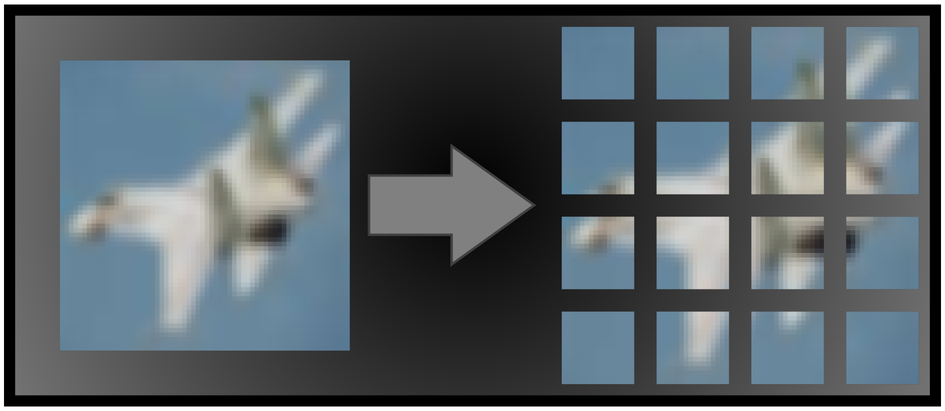


Figure 3: How a 64x64 pixel image may be tokenized into 16x16 pixel patches to be input to a Vision Transformer.

arranged sequentially passing output sequences to the next block's inputs, but they include more structure than simply a linear map and activation function. First, each token is passed through a normalization layer to keep the distribution of numerical values encountered regular across various inputs and blocks. The result is input to three separate, learned, linear maps to compute three collections of vectors: queries, keys, and values. These collections are manipulated and combined with a clever series of mathematical computations to produce new tokens in what is known as the self-attention mechanism. Namely, the linear maps are represented by matrices A_Q, A_K, and A_V for queries, keys, and values respectively, and they are combined as follows where X is a matrix containing each vector input token to the block:

$$\begin{aligned} Q &= A_Q X \\ K &= A_K X \\ V &= A_V X \\ \text{Attention}(Q, K, V) &= \text{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right) V \end{aligned}$$

where $\sqrt{d}$ is a normalization term corresponding to the dimensionality d of the input (Touvron et al., 2021). The result is that a series of tokens, the same number as were input, are generated as the output of this attention function. These output tokens are added back to the original input tokens, a feature of modern ANN architectures called a residual connection[1]. All of this self-attention mechanism may be stacked into multiple channels if multi-head self-attention is used, in which case there are some number of independent query, key, and value maps whose attention outputs are all concatenated together. The feature map after the residual connection is normalized again and passed through a very small FFN, usually one or two layers. Finally, the tokens before the second normalization are added back in to form another residual connection before the final output of the block is passed to the next block (Touvron et al., 2021).

Here, I focus on the specific model DeiT-Ti/16 due to its being a small-enough Vision Transformer to allow completing experiments in a reasonable time on the hardware available to me. The wider class of DeiT models achieved better top-1 classification accuracy than ViT on benchmarks such as ImageNet (Deng et al., 2009; Russakovsky et al., 2015), even without distillation, hence I chose to use this class of small model even without distillation. This variation of DeiT-Ti segments images into 16 by 16 pixel squares, each of which is projected to a 192 dimensional vector embedding to form tokens. It uses GeLu activation, the `CLS` token is trainable, and each of its 12 attention block has 3 heads (Touvron et al., 2021). It is neither a computer vision PFM like PeCo (Dong et al., 2022) nor is it as large as the 2.1 billion parameter SOTA Vision Transformer without convolutions CoCa (Yu et al., 2022). I aim to probe the security risks of this smaller Vision Transformer so that future investigations of larger ones may be directed toward the most important threats. To this end, I focus on a SOTA data poisoning technique augmented by Transformer-specific heuristics such as attention rollout.

6.3 Attention Rollout

Given the structure of the Transformer architecture, it may seem natural to interpret the magnitude of raw attention values (the output of the softmax) at block ℓ as a measure of the importance of corresponding input tokens of the whole network to determining the output tokens of

[1]Such connections were introduced by ResNets and are notable in that they allow training very deep networks without vanishing gradients. They heralded an era of great improvements in SOTA performance on computer vision tasks (He et al., 2015).

block ℓ. However, Abnar and Zuidema (2020) found that this metric is actually inadequate for determining relative contributions of various tokens in a text classification Transformer. Instead, they created attention rollout as a better metric of the contribution of tokens at layer i to those at layer j. Where A_ℓ is the matrix of raw attention values at layer ℓ, the attention rollout from block i to $j \geq i$, $\tilde{A}(i,j)$, is recursively defined as follows for a Transformer with residual connections:

$$\tilde{A}(i,j) = \begin{cases} A_i & \text{if } i = j \\ \tilde{A}(i,j-1)\,(A_j + I) & \text{otherwise} \end{cases}$$

For the purpose of this work, I calculate $\tilde{A}(1,L)$ for a vision transformer with L layers. I then use various entries in the matrix as a metric of how important patches in an input image are to the final prediction of a classifier. Inspired by its use for test-time and backdoor attacks by Gu et al. (2022), Lv et al. (2021), and Subramanya et al. (2022), I evaluate its potential to allow computationally cheaper attacks in a token-count-restricted data poisoning setting.

6.4 Witches' Brew

In the data poisoning attack setting, an attacker has the ability to control some amount of the training data images to some limited extent, especially in the context of models being retrained (Wang et al., 2019). In realistic settings, an attacker may have the ability to perform unlimited modifications, but may restrict their modifications to be stealthy enough so that model engineers do not immediately notice something is amiss with hostile training images. Additionally, such attacker access to training data is based on considerations of publicly sourcing training images. An attack on all the data is unlikely, but some small subset of it being poisoned is plausible. These observations inform constraints in the data poisoning attack setting both in the number of images that can be poisoned n and the amount by which pixel values can be modified from an otherwise clean image δ (Muñoz-González et al., 2019). More often, I will refer to the constraint n on number of images by what percentage of the total training set may be modified, known as the poison budget. In this thesis I introduce another constraint in the threat model for ACT methods: number of patches k that may be modified per poisoned image. Rather than capture limitations on practical access by an attacker, this constraint leverages the Vision-Transformer-specific token structure to potentially allow computationally cheaper attacks. I investigate how an attacker may intelligently navigate these constraints on image and patch count using heuristics to maximize attack effectiveness.

By evaluating poisoning attacks in the context of retraining Vision Transformers, my analysis stays relevant to the most likely threat to exceedingly ubiquitous PFMs. These large models are made to be fine-tuned to a downstream task with new data (Zhou et al., 2023), which may be collected in a public setting like social media where an attacker may subtly influence uploaded images. Further, their attack may be informed by the same publicly available pre-trained weights as will actually be used by victims. Given such a potential for real-world attacks, investigating how they may be performed is important for understanding how to prevent breaches on increasingly-deployed Transformers in security-critical settings. Geiping et al. (2021) found that highly constrained attacks on ResNet and VGG architectures are possible and that the poison can be relatively quickly computed with their Witches' Brew algorithm. I evaluate its effectiveness on Deit-Ti/16 for the first time and search for ways to make it faster and more effective on Vision Transformers generally.

Although data poisoning attacks may be carried out to achieve a wide variety of attacker objectives such as reducing accuracy in general (Xiao et al., 2012), reduce performance on a specific subpopulation of the data (Jagielski et al., 2021), or even anything that can be modeled by a loss function (Huang et al., 2021), Witches' Brew focuses on the goal of making a single target image $x_T \in \mathcal{X}$ during test time be classified into a specific inaccurate category $y_T \in Y$, both of which are inputs to Witches' Brew. The next input is a surrogate model, some classifier network meant to be as similar as possible, but not necessarily identical, to the model being attacked if it were trained on clean images. I consider an attack where the Vision Transformer architecture of the surrogate is identical to that actually being attacked, but randomization of training data and dropout is not necessarily the same, as expected from the practical threat model against PFMs. Given aforementioned inputs, problem constraints, and hyperparameters, Witches' Brew uses gradient matching as described in section 7.3. The idea is that the attacker would like to, in a loosely constrained environment, introduce many sample pairs of (x_T, y_T) into the training data so that the association is memorized for the same behavior during test time. Due to the ϵ-constraint on image modification, however, the attacker may opt to introduce poison so that training on the poisoned data approximates how the parameters θ of the model *would have* updated if the target pairs were inserted. Since the direction of loss gradient informs how parameters update, it is calculated on select training images with true target class y_T. This gradient is compared with the gradient of the loss on the target pair (x_T, y_T), forming a higher-order surrogate attacker loss $\mathcal{B}$ upon which gradient descent may be used to optimize the subtle poison noise added to the images (Geiping et al., 2021).

6.5 Mixup and Cutmix

Mixup is a data augmentation routine recommended for Vision Transformers like DeiT (Touvron et al., 2021). Transformers in general require many examples before training can converge, which is one reason for the popularity of fine-tuning from PFMs. However, even fine-tuning can require lots of data (Zhou et al., 2023). To address this, a popular remedy for deep learning is dataset augmentation, which increases the effective number of training samples by performing transformations to the original data that do not change their meaning with respect to the task (Goodfellow et al., 2016). Mixup is a special dataset augmentation that introduces transformations that combine multiple samples. In particular, it generates new samples from convex combinations of the original ones (Zhang et al., 2018). For example, let (x_1, y_1) and (x_2, y_2) be training samples with images x_1, x_2 and labels y_1, y_2. Further, let the labels be represented by one-hot encoded vectors – vectors in $\{0, 1\}^N$ where there are N classes, having 0's in all positions but the one corresponding to the class that the image belongs to. Mixup then samples a random variable

$\lambda \in [0, 1]$ to create a new training example (x', y') where

$$x' = \lambda x_1 + (1 - \lambda)x_2$$
$$y' = \lambda y_1 + (1 - \lambda)y_2$$

See Figure 4 for a visual example of this combination. Zhang et al. (2018), the creators of Mixup, find that this augmentation increases robustness to adversarial examples, as in test-time attacks. However, to the best of my knowledge, its ability to promote robustness against train-time attacks has not been evaluated. Given that Mixup is already recommended for Vision Transformers, I evaluate its effectiveness, in combination with Cutmix, in this capacity of defending against the ACT methods I develop.

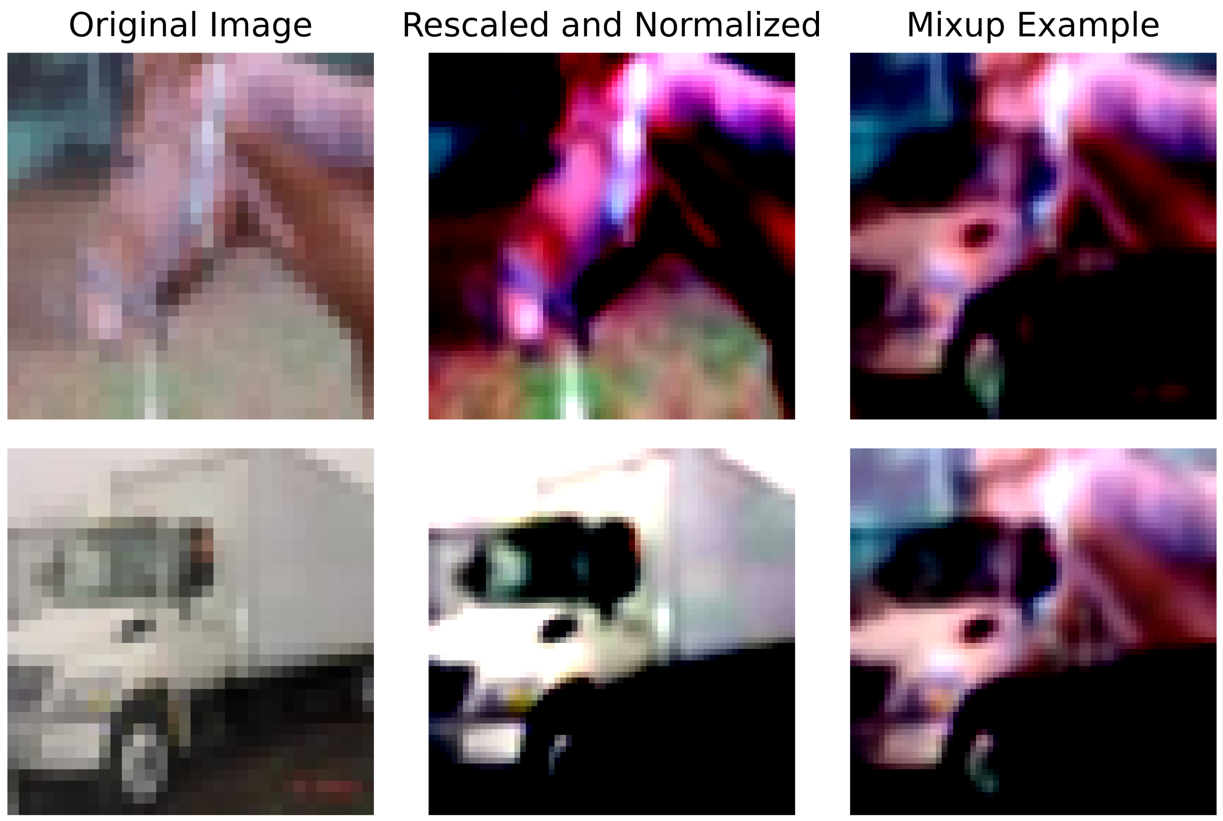


Figure 4: A pair of CIFAR-10 training images are rescaled and normalized as they are in the section 8.4, and then Mixup augmentation is applied to the pair. In this case, $\lambda = 0.6$ of each image remains while $1 - \lambda = 0.4$ of the other image is added. Not shown here, but one-hot labels are also interpolated.

Cutmix likewise creates new training examples (x', y') from pairs of training samples (x_1, y_1) and (x_2, y_2). Like Mixup, one-hot labels are interpolated by $y' = \lambda y_1 + (1 - \lambda)y_2$. Unlike Mixup, which superimposes images together, Cutmix cuts and pastes contiguous chunks of images into each other. See Figure 5 for an example. The augmentation creators Yun et al. (2019) found that it promotes robustness against test-time adversarial examples, but to my knowledge its effectiveness defending against train-time attacks has not been evaluated. Its use is already recommended by Touvron et al. (2021) for performance reasons, but I will evaluate its ability to defend against poisoning attacks when it is deployed with Mixup for Vision Transformers. Another side of defense against poisoning attacks is detecting their presence, for which GAS (Hammoudeh & Lowd, 2022) is a useful metric.

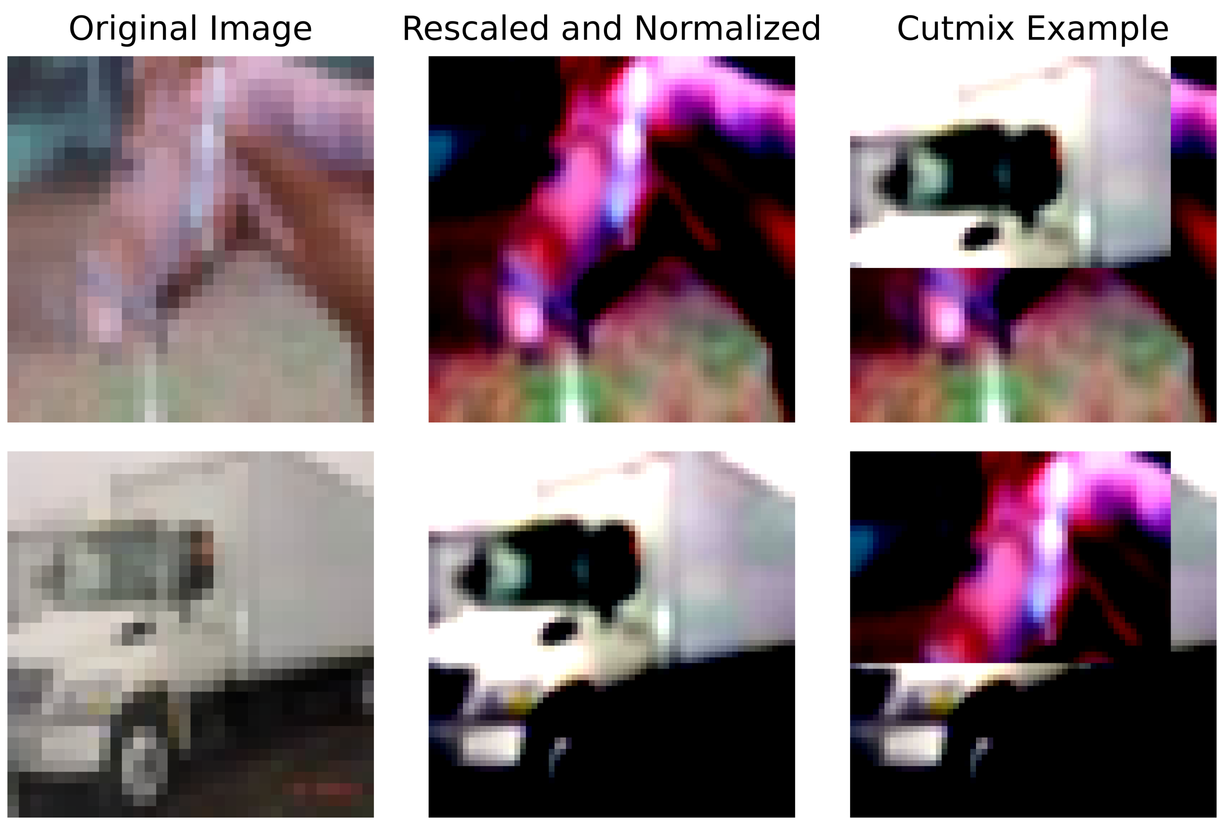


Figure 5: A pair of CIFAR-10 training images are rescaled and normalized as they are in the section 8.4, and then Cutmix augmentation is applied to the pair. Not shown here, but one-hot labels are also interpolated.

6.6 Gradient Aggregated Similarity

GAS is a training set influence estimator. Given some snapshots of a model f's parameters just before a collection of epochs τ, the learning rates λ_t at those epochs, and a sample $(\hat{x}, \hat{y})$ of a test image $\hat{x}$ and the prediction of its class by the model after training $\hat{y}$, GAS provides a measure of how important each training image was to that prediction. The idea is that the most influential images change the loss the most during training, which is exactly what its expression captures:

$$\text{GAS}((x,y),(\hat{x},\hat{y})) = \sum_{t\in\tau} \frac{\lambda_i}{b} \frac{\left\langle \nabla_\theta \mathcal{L}(x,y;\theta)|_{\theta_{t-1}}, \nabla_\theta \mathcal{L}(\hat{x},\hat{y};\theta)|_{\theta_{t-1}} \right\rangle}{||\nabla_\theta \mathcal{L}(x,y;\theta)|_{\theta_{t-1}}||_2 ||\nabla_\theta \mathcal{L}(\hat{x},\hat{y};\theta)|_{\theta_{t-1}}||_2}$$

where (x, y) is the training sample image and label whose influence is to be influenced and b is that batch size used in training. With a similar expression to the gradient matching objective used in Witches' Brew and my wider ACT methods, given in section 7.3, there is a mathematical elegance in using it here. While Hammoudeh and Lowd (2022) introduced it for detecting poisoning attacks, I will use it as something for an attacker to optimize in attack methods using image selection. In particular, I will select a τ with a single epoch, the last one, for using GAS as an image selection heuristic. This will make the generated poison samples boldly stand out if a victim uses GAS for detection, but I will evaluate if the resulting attack is any stronger.

7 Attack with Choices for Transformers Methods

I consider two agents at work: the victim V and the attacker A. The victim V's job is to train a Vision Transformer model f for classification. V has gathered images for training X along with their labels Y. Using some supervised training technique, the parameters θ of f will be fine-tuned from pre-trained weights for this classification task. Meanwhile, an attacker A knows V's plans for f. A has their own goal: to use limited access to the training images X to induce f to behave differently after training. Namely, f should classify as V intends *except* when the specific test-image x_T is input to f during testing. A wants the model f to classify this target image x_T as belonging to some ground-truth-incorrect class y_T. V trains f with the goal that all test images, including x_T, are assigned their correct class with high probability, but A wants this goal to fail on x_T. Note that I will sometimes refer to x_T's true class as the poison class, as opposed to the target label y_T. I use this convention because I draw images to poison from x_T's true class, as discussed in section 7.4. In this section, I will describe my **A**ttack with **C**hoices for **T**ransformers (ACT) method for the attacker A to achieve their goal in the poisoning setting.

In this white-box attack setting (Chakraborty et al., 2018) where attacker A has knowledge of the architecture and training procedure of f, A will carefully compute how to achieve their own goal. In this section, I present the details of a category of attack methods A may use for this computation. While section 5 touches upon extant methods in the literature that A may use, I present a superclass of Witches' Brew, ACT methods, made to target Vision Transformer victim models f. Writing from the perspective of the attacker A using this method, I present a thorough mathematical description of the setting and then describe this patch-based attack method.

To use this novel method, A first trains a surrogate clean classification model f_c of their own using V's clean training data (X, Y). A performs this training to sample a possible configuration of f's parameters θ during training. With such a snapshot, A uses f_c as part of heuristics for choosing images $X_P \subset X$ to alter, which patches within those images to constrain modifications to, and as an input to a modified Witches' Brew algorithm for the computation of poison noise Δ.

7.1 Problem Formulation

V will train a machine learning model $f : \mathcal{X} \to \mathbb{P}_Y, f \in \mathcal{H}$ on training images $X \subset \mathcal{X}$ with labels Y. f takes images $\mathcal{X}$ as input and outputs a probability distribution over the labels $\mathbb{P}_Y$. In this classification task, Y is a set of classes that partition the images $\mathcal{X}$. The hypothesis space of models $\mathcal{H}$ is all Vision Transformers, of which the specific network topology is described in section 6.2. A can choose a subset of the images $X_P \subset X$ to alter before V uses them in training. Call the remaining images $X_C = X \setminus X_P$. A wants f to classify some image $x_T \notin X$ with label $y_T \in Y$ with as high of a probability as possible after training. I model this attacker objective with a loss function $\mathcal{L}_{att}(x_T, y_T; f)$ that evaluates how well f conforms to A's goal.

The setting here is single-target: the attacker A wishes to induce f to classify a given single image x_T as given target label y_T with maximal probability. This goal can be simply captured by cross-entropy loss. Where $f_{y_T}(x_T)$ is the probability assigned to class y_T by f:

$$\mathcal{L}_{att}(x_T, y_T; f) = -\log\left(f_{y_T}(x_T)\right)$$

To minimize $\mathcal{L}_{att}$, A will find adversarial noise to add pixel-by-pixel to images in X_P, each image therein being given its own noise by the attacker. X_P itself may be chosen from possible subsets of X using heuristics that I will describe in section 7.4; assume that this has already been done. Where $|X_P| = n$ is the maximum number of images A may choose, I impose an ordering over X_P and call the instances of adversarial noise $\Delta = \{\delta_i : 1 \leq i \leq n\}$ to correspond to images $x_i \in X_P$. I consider $\mathcal{H}$ to be Vision Transformers, so I restrict δ_i to non-zero noise only in regions corresponding to a subset of k tokens (patches) in images as segmented by f. This restriction aims to hasten computation of noise Δ with only minor penalty to poison effectiveness. To model this restriction, let M be a set of masks m_i $(1 \leq i \leq n)$ corresponding to images at indices i. Assign 1's to m_i's in all positions of pixels within selected tokens, but 0's everywhere else. For each image $x_i \in X_P$, the poisoned image is then given by $x'_i = x_i + m_i\delta_i$, where addition and multiplication are element-wise. Denote the poisoned dataset by $X'_P = \{x'_i : 1 \leq i \leq n\}$,

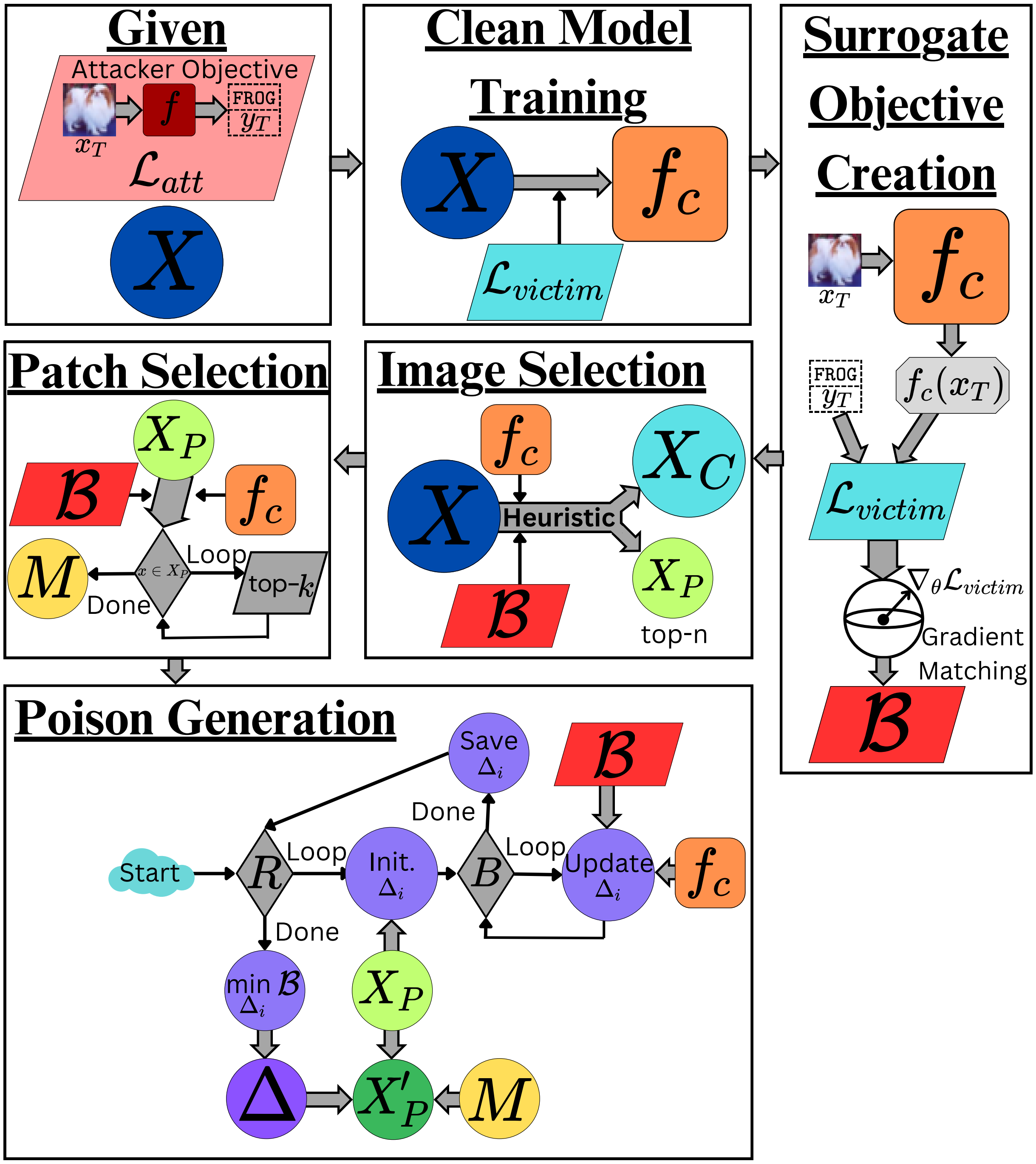


Figure 6: Flowchart of how the ACT Method generates poisoned images. A rhombus shape indicates a for-loop over a set or number of steps.

noting that X_P is composed of *clean* images that are selected to be later poisoned by the attacker to create X'_P.

A chooses patches either randomly or using heuristics. One such heuristic is gradient mag-

Notation	Brief Explanation
V	The Victim
A	The Attacker
f	V's Classifier Model – A will attack this
θ	Parameters of f
x_T	A's Target Image
y_T	A's Target Class; A wants $f(x_T)$ to predict this
$\mathcal{X}$	Set of All Images
X	Training Images
Y	Image Labels
$\mathbb{P}_Y$	Set of Probability Distributions over Labels
$f_y(x)$	Probability that f assigns to x being in y
$\mathcal{H}$	Hypothesis Space of Models; Set of All Possible θ
f_c	A's Clean Surrogate Model
$X_P = \{x_i\}$	Images Selected by A to Poison
X_C	Remaining Images Not Selected by A
X_T	Images with x_T's True Class
$\Delta = \{\delta_i\}$	A's Adversarial Noise – will be added to X_P
Δ_i	Intermediate Adversarial Noise during A's Brewing
$M = \{m_i\}$	A's Patch Masks – restricts Δ to selected tokens
$X'_P = \{x'_i\}$	A's Poisoned Images
X_P*	The Optimum X'_P Solving the Bilevel Problem
n	Maximum Number of Images A May Modify
k	Maximum Number of Patches/Image A May Modify
P	Total Number of Patches/Image
ϵ	Magnitude Constraint – A's δ_i's must be within this
$\mathcal{L}_{att}$	A's Loss Function
$\mathcal{L}_{victim}$	V's Loss Function
$\mathcal{B}$	A's Gradient-Matching Surrogate Loss Function
λ	Learning Rates, Convex Combination Ratios
A_ℓ	Raw Attention at Block ℓ
$\tilde{A}$	Attention Rollout
τ	Set of Epochs Used for GAS Computation
R	Number of Poison Brewing Restarts
S	Number of Poison Brewing Steps per Restart

Table 1: Brief explanations of notation

nitude of the attacker's gradient matching objective (see section 7.3) as restricted to individual patches. The other is attention rollout from input image patch tokens to the classification token `CLS`. As mentioned earlier, A also chooses images X_P from X either randomly or according to some heuristic such as gradient magnitude as before but across whole images. A intends for resulting poisoned images X'_P to confuse the victim model f when trained on the whole set $X_C \cup X'_P$. Namely, A intends that $f(x_T)$ assigns as high a probability to y_T as possible.

In real attack settings, the victim V might notice strange training images and remove them. To make poisoned images in X'_P look natural or stealthy, A imposes a constraint on the magnitude of added noise. Namely, for all images $x_i \in X_P$, $||\delta_i||_\infty \leq \epsilon$ for some small ϵ, where $||\cdot||_\infty$ is the ℓ^∞ norm. This norm is the sum of the absolute values of each element in the vector.

The victim V will train network f to minimize some loss $\mathcal{L}_{victim}$ on the poisoned dataset $(X_C \cup X'_P, Y)$. I will now model this setup as an optimization problem between the attacker A and victim V. Call the optimum set of poisoned images for the attacker

$$X^*_P = \{x_i + m_i\delta_i; x_i \in X_P \subset X\}$$

Note that the images in X are fixed but A is free to choose x_i's, m_i's, and δ_i's within some constraints. All together, this training by V while A crafts poison forms a bilevel optimization problem:

$$\begin{aligned}
&\min_{X_P,\Delta,M} \mathcal{L}_{adv}(x_T, y_T; f^*) \\
\text{s.t. } & f^* \in \operatorname*{argmin}_{f\in\mathcal{H}} \mathcal{L}_{victim}(X_C \cup \{x_i + m_i\delta_i : i \in \{1,\dots,n\}\}, Y; f) \\
& |X_P| = n \\
& ||\delta_i||_\infty \leq \epsilon, \forall\delta_i \in \Delta \\
& m_i \text{ is restricted to } k \text{ patches}, \forall m_i \in M
\end{aligned}$$

The X_P, Δ, M that solve the upper level problem determine the optimum X^*_P for the attacker A.

The procedure I describe in the rest of this section is for A to approximate X^*_P with a poisoned set X'_P. There are six steps for A to follow: clean model training, surrogate objective creation, image selection, patch selection, poison generation, and execution. The final execution step is performing the modifications to X so that V's training data is $(X_C \cup X'_P, Y)$, and it is a commonality of all poisoning attack methods. ACT methods incorporate a modified Witches' Brew component to find Δ as restricted to k patches. This component accounts for the clean model training, surrogate objective creation, and poison generation steps. My main method contributions are in the image and patch selection steps as well as making necessary changes to Witches' Brew to accommodate patch-restricted poison noise Δ.

7.2 Clean Model Training

To train a clean model f_c, A starts with the same pre-trained parameters V uses for training the real model f. For Vision Transformers, and especially PFMs that are larger than the models I consider here, this is a realistic assumption since such pre-trained parameters are often released publicly, as they are in the case of DeiT-Ti/16 which I will use for evaluations (Touvron et al., 2021). A has access to the training algorithm that V will use, complete with augmentations and knowledge of its stopping criteria. However, I do not assume A has access to the random seeds that determine the order in which samples appear in training, random augmentations, and dropout if used. This ACT method framework does not necessarily prescribe that f_c is trained in exactly the same way that f will be. For evaluations of the method, I use the same training algorithm up to random states to create f_c; doing so makes the resulting parameters θ more commensurate with those that will appear during the training of f. In any case, A uses the victim V's training dataset (X, Y) to create a clean classification model f_c.

7.3 Surrogate Objective Creation

After the attacker A trains a clean model f_c, both this method and Witches' Brew prescribe that the next step is to create a surrogate objective whose solution approximates that of the bilevel problem (Geiping et al., 2021). Solving the bilevel problem directly is computationally intractable due to the need to repeatedly train a victim model. So, Geiping et al. (2021) proposed a gradient matching surrogate problem. The objective $\mathcal{B}$ is a function of a model (f_c in particular), images to poison X_P, the adversarial noise to add to them Δ, and the patches the noise is constrained to M. My ACT method framework prescribes that A choose the images to poison X_P in the next step and patch masks M after that, but I will reference both here understanding that they are *inputs* to $\mathcal{B}$. Just as Δ has not been computed yet is referenced, these other variables will later be input to this objective for optimization purposes. The main contribution of this step to the whole method is calculating a gradient used by the objective $\mathcal{B}$ which will be matched during poison generation as described of unmodified Witches' Brew in section 6.4.

Witches' Brew helps A induce $f(x_T)$ to have as great a probability assigned to y_T as possible by finding Δ such that training the victim network f on the poisoned images X'_P has nearly the same effect as training on the single example (x_T, y_T) repeatedly. This is achieved by solving the following optimization problem of matching gradient updates, where I have modified the objective from Geiping et al. (2021) as appropriate for this patch-based method:

$$\begin{aligned} &\min_{\Delta} \mathcal{B}\left(f_c, X_P, \Delta, M\right) \\ \mathcal{B}\left(f_c, X_P, \Delta, M\right) =& 1 - \frac{\langle \nabla_\theta \mathcal{L}_{victim}(x_T, y_T; \theta), \sum_{i=1}^{n} \nabla_\theta \mathcal{L}_{victim}(x_i + m_i\delta_i, y_i; \theta)\rangle}{||\nabla_\theta \mathcal{L}_{victim}(x_T, y_T; \theta)||_2 \cdot || \sum_{i=1}^{n} \nabla_\theta \mathcal{L}_{victim}(x_i + m_i\delta_i, y_i; \theta)||_2} \end{aligned}$$

I present this optimization problem here to show the definition of $\mathcal{B}$, its significance, and how I have altered it from Witches' Brew to include patch-selecting masks M. The optimization problem itself is not solved until the poison generation step in section 7.6. Note that $\langle \cdot, \cdot \rangle$ is the dot product in this context and θ refers to input f_c's parameters. This gradient matching procedure maximizes the cosine of the angle between the desired gradient and the actual gradient on poisoned images of the victim's loss. That is, the angle between them is minimized. Note that A can cache $\nabla_\theta \mathcal{L}_{victim}(x_T, y_T; \theta)$ after obtaining θ from training the clean model f_c since the target pair x_T, y_T are given parts of their own objective. Again, the optimization problem of finding Δ will be solved during the poison generation step after X_P and M are fixed in the next steps, this step merely creates the objective $\mathcal{B}$.

7.4 Image Selection

Unlike Witches' Brew, this method proposes deliberate selection of images to poison $X_P \subset X$ using heuristics. I evaluate three heuristics for use with this method, but in principle any way of ordering images in X according to appraisal of their promoting a successful attack (that only uses information available to the attacker A at this step) may be used. Like Geiping et al. (2021), the heuristics I evaluate start by restricting choice of X_P to those images with true label y_T, call those images $X_T = \{x_i : y_i = y_T, x_i \in X\}$. Note that this indexing is independent of that imposed on X_P; the one I use here indicates corresponding labels in Y for X_T. Within those images, A chooses a subset $X_P \subset X_T$. I will model this selection under the assumption that it has been restricted to X_T since that is what I use in evaluations, but for the general case one can set $X_T = X$.

An image selection heuristic assigns values v_i for each image x_i with $1 \leq i \leq |X_T|$. I consider two heuristics: gradient and random. Using the latter is equivalent to Witches' Brew's image selection strategy. Note that this random selection in combination with a patch constraint $k =$ the total number of patches per-image produces Witches' Brew without modification; the ACT method is a more general form of Witches' Brew. In either case, the heuristic assigns v_i's. A then creates X_P by choosing those images x_i with the top-n largest v_i's. That is, A solves

$$\max_{X_P} \sum_{x_i \in X_P} v_i$$

$$\text{s.t. } |X_P| = n, X_P \subset X_T$$

If there is a tie, A randomly chooses between tied images if this tie would affect the selection. I describe the precise assignment of values by each heuristic as follows.

Gradient Image Selection (G-IS). A calculates the magnitude of the gradient of the adversarial gradient matching objective $\mathcal{B}$ as evaluated on individual *images*. For each $x_i \in X_T$, A uses the clean model f_c to set values $v_i = ||\nabla_{x_i}\mathcal{B}(f_c, \{x_i\}, \{0\}, \{1\})||_2$. The 1 in the mask position represents a mask with scalar 1's in all positions so that the whole image is considered. This estimates how promising each image is according to the initial steepness of the slope that adversarial noise will be updated along in the poison generation step.

Gradient Aggregated Similarity Image Selection (GAS-IS). This image selection heuristic assigns values so that images with the greatest influence on the attacker objective as calculated by GAS are selected: $v_i = \text{GAS}((x_i, y_i), (x_T, y_T))$. A uses a τ so that influence is calculated only on the final parameters obtained by the clean model f_c after A has trained it. Given the similarity in form to the attacker surrogate objective $\mathcal{B}$, and this choice of epoch set τ, A can perform an equivalent optimization by assigning values $v_i = -\mathcal{B}(f_c, \{x_i\}, \{0\}, \{1\})$. That is, selecting images by GAS is equivalent to choosing images whose surrogate loss is already the lowest. Contrast this to G-IS which chooses images according to how fast such loss is estimated to *decrease*. In principle, A could use a variation on this which also incorporates parameters of f_c at various aspects during training in the GAS calculation, and also normalize blocks of f_c independently, but in this work I evaluate using it as just described.

Random Image Selection (R-IS). A sets all values to the same number, say $v_i = 0$, for all $x_i \in X_T$. There is an all-way-tie, so X_P will be a random subset of X_T.

After A selects the poison set X_P with one of these heuristics, the next step is to restrict the adversarial noise applied to each to be non-zero only in k patches of each image.

7.5 Patch Selection

The next contribution this method makes over Witches' Brew is to address its own constraining adversarial noise to patches. In particular, in this step A uses heuristics to choose which patches to poison. This choice is encoded in the choice of masks M. Each $m_i \in M$ will have 1's in positions corresponding to exactly k patches and 0's elsewhere. Recall that the poisoned set is formed by $X'_P = \{x_i + \delta_i m_i\}_{i=1}^n$ where each of the images x_i, poison noise δ_i, and masks m_i correspond to one another. I will evaluate three heuristics for A's choice here, but as in the previous step, other heuristics may be used in principle.

A patch selection heuristic assigns values v_{ij} for each image x_i and in each patch j thereof.

To rigorously refer to patches in images, let's define a convention for the structure of images. Assume each image is square, s pixels across, and has c color channels. There are $x_i, m_i, \delta_i \in \mathbb{R}^{c \times s \times s}$ for each $1 \leq i \leq n$. By the construction that follows, $m_i \in \{0,1\}^{c \times s \times s}$. Let $p_j \in \{0,1\}^{c \times s \times s}$ be a mask with 1's in positions corresponding to patch j and 0's elsewhere. Let there be P patches total.

Given v_{ij} assigned by a heuristic, for each x_i the attacker A finds the top-k v_{ij}'s and combines corresponding p_j's into m_i. That is, A defines for each $x_i \in X_P$ a function $\phi_i : [1,p] \rightarrow [1,p]$ that orders v_{ij}. Specifically, $\phi_i(a) = j$ means that patch j has the a^{th} greatest v_{ij}. That is, among the patches in image x_i, patch j has the a^{th} greatest value assigned by the heuristic. The masks are then given by

$$m_i = \sum_{j=1}^{k} p_{\phi_i(j)}, \; \forall x_i \in X_P$$

In plain English, the attacker A uses a heuristic to find a mask selecting the top-k most promising patches on a per-image basis. Now that the mask creation process has been explained from given v_{ij}, I will explain how each of the heuristics assigns v_{ij}.

Attention Rollout Patch Selection (AR-PS). This heuristic is based on attention rollout as described in section 6.3. For each image $x_i \in X_P$ and for each patch at position j within them, A calculates the attention rollout from the token corresponding to patch j to the last `CLS` token in the final attention block, $v_{ij} = \tilde{A}(1,L)_{j,\texttt{CLS}}$, where f has L layers of blocks. This is done by passing x_i through the clean model f_c and recording attention maps A_ℓ along the way for the rollout calculation. Since attention rollout measures importance of tokens at some layer to determing those at another, this heuristic focuses on how important image patches are to the final classification token, which is passed through a small FFN to obtain f's probability distribution over classes. Inspired by other work using this importance metric for test-time attacks, I evaluate its use for this train-time poisoning attack method.

Individual Gradient Patch Selection (IG-PS). As in image selection, the idea here is to find the gradient magnitude of the surrogate objective $\mathcal{B}$ as evaluated at each chosen poison image. This way, the selected patches will be those whose corresponding adversarial noise will update the most at the start of poison generation, which I interpret as a sign of importance. For each patch j of each image x_i, A assigns values $v_{ij} = ||p_j \nabla_{x_i} \mathcal{B}(f_c, \{x_i\}, \{0\}, \{1\})||_2$. This captures the magnitude of the gradient *as constrained to each patch*. Notice that the gradient is computed on the full image, the mask passed to $\mathcal{B}$ is all 1's, but then it is restricted to the patch j in question by element-wise multiplication by p_j.

Universal Gradient Patch Selection (UG-PS). In the case of patch selection, there is another way to use gradient information. Instead of computing per-image gradients, some libraries have fast implementations of gradient computation as additively accumulated over a whole batch of images. This heuristic seeks to harness that convenience and speedup. In effect, A first computes values as in IG-PS, call them v'_{ij}. Here, A chooses those patches whose *average* gradient magnitude per-patch over all images is greatest. That is, each image will have the same mask assigned to it. That is, for each image x_i, A assigns values

$$v_{ij} = \frac{1}{n} \sum_{i'=1}^{n} v'_{i'j}, \forall j$$

7.6 Poison Generation

Finally, the next step is to generate the poison Δ. This largely proceeds according to Witches' Brew with appropriate modifications for patch-restriction. See Figure 6 for a visual explanation of the process, as well as the other stages. In short, A approximates the solution to the optimization problem in section 7.3 using gradient descent on the color values in each δ_i. To remain within the ϵ-constraint, A projects each δ_i within a ϵ-radius ℓ^∞ ball, in doing so keeping the poison noise subtle.

As recommended by Geiping et al. (2021), I perform the optimization process R times, or "restarts." Associate with each restart a different version of Δ, call them Δ_i for $1 \leq i \leq R$. Each Δ_i is given a different random initialization, clamped to lie within a radius-ϵ ℓ^∞-ball, and their constituent perturbations may be updated in different orders according to batch randomization in each round of poison brewing, which I will call the inner optimization loop. The attacker A

performs this poison brewing R times to create each Δ_i, then A chooses the Δ_i (after brewing) with the least surrogate loss $\Delta^* = \min_{\Delta_i} \mathcal{B}(f_c, X_P, \Delta_i, M)$ to be the final adversarial noise $\Delta = \Delta^*$. As stated by Geiping et al. (2021), the purpose of these restarts is to overcome the sensitivity of success to poison initialization.

For each poison brewing round, A uses signed Adam updates (Kingma & Ba, 2017) to optimize each Δ_i. A performs these updates over S epochs, where the perturbations Δ_i are updated in batches. At each step, each $\delta_j \in \Delta_i$ is clamped to be within the ϵ-constraint. Note that the objective $\mathcal{B}$ involves a gradient computation dependent on Δ_i while simultaneously $\mathcal{B}$'s gradient is found for updating Δ_i. This implicit Hessian calculation is associated with a large demand on memory, especially in the case of using large images and Vision Transformers which have millions of parameters. This is one benefit of using patch restrictions. The heavy memory load is reduced by restricting each δ_i to be non-zero only within a few tokens. Such a restriction may not only allow quicker computation, but in fact be necessary for scaling up this ACT method for use with PFMs, which may have billions of parameters. In my own experience with this method, I frequently encountered memory overflows when not restricting patches.

To incorporate patch selection heuristics into Witches' Brew, some modifications to the algorithm are necessary. By this stage, A has selected masks M. After initializing δ_i with random noise, overwrite $\delta_i \leftarrow \delta_i m_i$ (element-wise multiplication) to set all entries at positions outside of the chosen k patches to 0. Next, at each step of gradient descent, apply the mask m_i to the gradient of the adversarial gradient matching surrogate objective so that the update to δ_i and subsequent clamping to the ℓ^∞-ball is entirely performed in the k chosen patches. The new update is then, for each image $x_i \in X_P$ with perturbation δ_i,

$$\delta_i \leftarrow \delta_i + m_i \lambda \nabla_{\delta_i} \mathcal{B}\left(f_c, \{x_i\}, \{\delta_i\}, \{m_i\}\right)$$

In this way, gradient-descent-based training is thwarted by another algorithm using gradient descent.

After each of the R poison brewing rounds are complete and A chooses the best Δ, the last step is to carry out the attack with A's access. In realistic scenarios, A's image selection would be in the form of choosing some subset from their own collection. After creating appropriate Δ through this ACT method, A would add this adversarial noise to the images themselves and provide X'_P to the victim V with clean labels. In practice A could upload the poisoned images to the Internet. V would then collect X'_P along with other images X_C, not modified by the attacker A, and perform training. If all goes well for A with this method, the resulting f will produce a probability for test image x_T, $f(x_T)$, that assigns highest probability to y_T among all other classes. This method and its heuristics are designed for resource-intensive Vision Transformer f, which are increasingly prominent. In the next section, I will explain how I evaluated how well ACT methods work for an attacker A targeting a small Vision Transformer f. Additionally, I evaluated how V may use Mixup, Cutmix, and other training augmentations to defend against attacks like this.

8 Experiments and Results

I evaluated the ACT method's effectiveness using various heuristics when deployed against a victim model f using the DeiT-Ti/16 architecture (Touvron et al., 2021). Unlike the study introducing it, I did not use knowledge distillation for training. However, for classifying the small datasets CIFAR-10 and CIFAR-100 that I use, accuracy with and without distillation is within a few percentage points. By conducting extensive computational experiments, I established an increase in attack strength over Witches' Brew associated with use of heuristics. In this section, I describe a preliminary investigation of Witches' Brew attacks on DeiT-16/Ti that motivated the ACT method's development, the setup of subsequent experiments using the ACT method, what I observed, and the implications of the results. In each experiment, I acted as both the attacker A and the victim V. As A, I selected a target image x_T and label y_T and the trained a model f_c on clean data. Next, I used a poisoning attack method with f_c to generate adversarial noise Δ. For all experiments, I used $S = 250$ poison brewing steps. I then switched roles to V and trained several models on the full poisoned dataset $X'_P \cup X_C$. I evaluated attack strength based on the probability distribution output of poisoned $f(x_T)$ at various stages in training. Successful attacks induced a high-confidence classification of x_T as y_T, and the strongest attacks did so under the tightest constraints.

In section 8.1, I describe my evaluations of regular Witches' Brew (ACT method with random image selection and no patch restriction) with respect to effectiveness attacking DeiT-Ti/16. In a few experiments I established two important facts: heavy augmentations are useful as defense and DeiT-Ti/16 has hightened natural robustness against poisoning attacks. The former hints at Mixup and Cutmix's defensive properties and the latter fact motivates the need to improve upon Witches' Brew with heuristics specific to Vision Transformers.

In section 8.2, I explain my fixing a model training procedure used for all following experiments, where the victim V's model f and the attacker A's clean surrogate models f_c used the same procedure. Fixing a standard training procedure allowed me to focus on evaluating the effects of various heuristics on attack strength, which I did in a multidimensional manner. In section 8.3 I explain the specifics of how I did so. In section 8.4, I describe my evaluations. The strongest attack I found used GAS-IS and AR-PS.

8.1 Preliminary Witches' Brew Evaluation

Before evaluating the broader ACT method, I first considered whether Witches' Brew alone can perform a successful poisoning attack against DeiT-Ti/16 with its recommended training setup. In particular, I trained models as Touvron et al. (2021) describe, but without distillation. Using recommended hyperparameters for fine-tuning for classification of CIFAR-100, an attack was only successful in a special case that departed from the data poisoning setting of this work. When heavy augmentation was removed, instead using a model training procedure with hyperparameters traditionally suited to CNNs, Witches' Brew was able to create a successful attack on DeiT-Ti/16 with loose constraints.

For this Witches' Brew evaluation, I used $R = 4$ restarts for resource-intensive Transformer-style training and $R = 12$ restarts for ResNet-style training. In either case, I used the standard number $S = 250$ of poison brewing steps.

8.1.1 Model Training Procedure

The recommended (Touvron et al., 2021) retraining pipeline for DeiT-Ti/16 on CIFAR-100 involves heavy augmentations. I used this for the first two experiments of this subsection. First, images were re-scaled to 439 by 439 pixels by bilinear interpolation and then center-cropped to 384 by 384 pixels. The color was then regularized to fit ImageNet statistics. During training, both Mixup and Cutmix were used to combine random pairs of samples and a learned AutoAugment provided further transformations post-mixes. See Figures 4 and 5 for examples of training images pre-AutoAugment. I used both Mixup and Cutmix while training clean surrogate models f_c and then while poisoning models f. In both cases, I started from pre-trained weights made available by Touvron et al. (2021). I used AdamW (Loshchilov & Hutter, 2019) as an optimizer and used a cosine learning rate schedule with a warmup period. I scaled learning rate to accommodate a batch size of 64 as recommended by Touvron et al. (2021). While fine-tuning on CIFAR-100 with clean data, I was able to obtain 76% top-1 validation accuracy by training for 30 epochs. This accuracy was calculated by evaluating accuracy on CIFAR's 10,000 test images, which I did not using during training nor will I during future experiments.

Touvron et al. (2021) achieved 90.8% validation accuracy on CIFAR-100 with the larger DeiT-B architecture. One of the only figures given for DeiT-Ti is 72.2% validation accuracy on ImageNet. Meanwhile, they found that DeiT-B achieved 81.8% accuracy classifying ImageNet. Given that DeiT-Ti's ImageNet accuracy is about 10 percentage points less than DeiT-B's accuracy, I believe the 76% figure is reasonably close to the optimum accuracy for the tiny version DeiT-Ti on the CIFAR-100 classification task. Achieving such a good approximation to optimum performance is desirable in general for attackers creating clean models f_c because the victim V will search themselves for the optimum in their lower-level objective as in the bilevel optimization problem of section 7.1.

Later, I additionally trained DeiT-Ti/16 using augmentations and hyperparameters that Geiping et al. (2021) use for their ResNet evaluation experiments. In this case, the task was classifying CIFAR-**10** instead of -100. One motivation for using the former hyperparameters was to create an easier model to attack for the beginning of ACT method evaluation experiments than one trained with heavy augmentations. The choice of CIFAR-10 also allows comparison with some of Geiping et al. (2021)'s experiments. As before, the training techniques and hyperparameters were shared between the clean model f_c used by the attacker A and the multiple poisoned models f trained by the victim V. I did not upscale images in this case; I instead opted to keep CIFAR-10 images at their native 32 by 32 pixel resolution, even though this corresponds to only 4 tokens. I used light augmentation including random flips and random crops. As before, I used the AdamW optimizer, but now with a linear learning rate schedule. I trained all models in this light-augmentation category for 40 epochs.

8.1.2 Mid-Training Poisoning

Before performing experiments in the data poisoning setting that is the focus of this thesis, I first verified that it is plausible to poison the DeiT-Ti/16 architecture to begin with. To this end, I used Witches' Brew without modification *except* that I deployed the poison differently. Unlike the data poisoning setting in which the attacker A provides poisoned images at the beginning of training, I simulated an attacker with access to the training data in the middle of training. Specifically, A has access to the specific weights in the victim's model f in the middle of training and is able to use Witches' Brew to poison images in-place at this middle epoch to hijack training. In practice for my evaluation here, instead of initializing victim networks f to Touvron et al. (2021)'s pre-trained weights, I instead resumed training from the clean model f_c's weights. The significance of this relaxed setting is that poisoned images X'_P are maximally prepared to influence training on the first epoch of resumed training because the parameters are identical to what Δ was optimized with.

I trained DeiT-Ti/16 as recommended by the model's designers for 5 epochs, calculated Δ with Witches' Brew from resulting parameters, but then resumed training with the poisoned dataset $X'_P \cup X_C$ instead of training with poisoned data from the beginning.

Using a poison image budget of 0.1% (50 images in CIFAR-100, see Figure 7 for the poisoned images) and magnitude constraint $\epsilon = 32$, for 5 more epochs I resumed training f_c but on poisoned data. Before the beginning of that resumed training, the randomly selected target image x_T was assigned its true label by f. By the end, a validation accuracy of 74% was close to reference clean validation accuracy 76%. Confident that successful poisoning would unlikely be a fluke, I observed that the most confident class predicted by $f(x_T)$ was indeed the attacker's target y_T instead of x_T's true class. Poisoning in this modified setting was successful.

8.1.3 Heavy Augmentation Defense

Equipped with knowledge that it is possible in principle to fool DeiT-Ti/16 with a poisoning attack, I next attempted to use Witches' Brew to poison it in the conventional setting and with

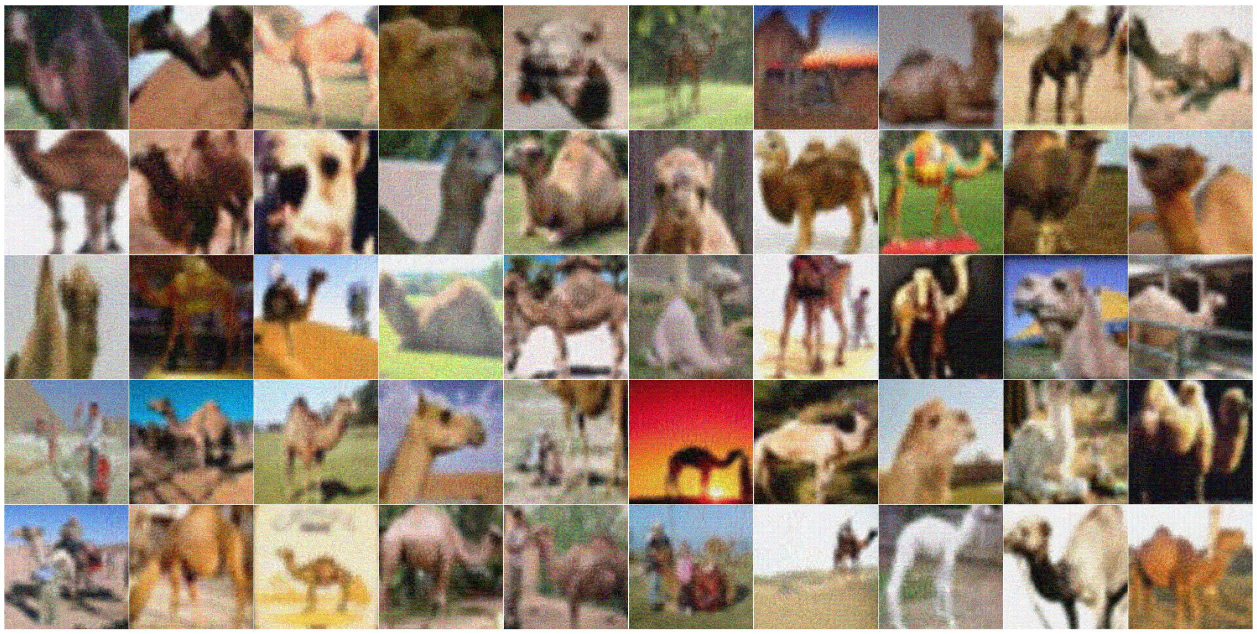


Figure 7: The poisoned data X'_P generated by Witches' Brew with no modifications for poisoning CIFAR-100. Images are made to look natural by imposing a somewhat-tight constraint $\epsilon = 32$. The attacker objective was to classify a test image of a cockroach as a camel.

recommended Vision-Transformer-specific training. That is, unlike resuming training in the previous section, after poisoned images X'_P were generated I set aside the clean model f_c and poisoned a new model f initialized with the standard pre-trained parameters. Unlike experiments described elsewhere in this section, however, f_c and f were trained over different numbers of epochs. f_c was trained on clean data for 10 epochs while f was trained on poisoned data for 15 epochs. Like the last experiment, I used a poison budget of 0.1% and generated poison within a magnitude constraint of $\epsilon = 32$.

This attack on DeiT-Ti/16 failed to induce $f(x_T)$ to predict y_T. For reference, Witches' Brew's successful poisoned Google's Cloud AutoML with the same poison budget and ϵ-constraint (Geiping et al., 2021) and with a victim training for ImageNet classification. In the same task, which is more difficult than CIFAR-100 classification, they found that Witches' Brew was successful 80% of the time with constraints as tight as a 0.05% poison budget and $\epsilon = 8$ against ResNet-18 models. Their attacks on the easier CIFAR-10 classification task, which is theoretically more difficult to poison than ImageNet, were likewise successful in 91% of their trials with a 1% poison budget and $\epsilon = 16$ against ResNet-18 models. Regardless of reference class, my failure to poison a DeiT-Ti/16 model trained on a theoretically easier-to-poison classification task CIFAR-100, even with the SOTA Witches' Brew algorithm, suggests that Vision Transformers have a heightened natural robustness against data poisoning attacks.

Validation accuracy was degraded from 74% (measured on f_c) to 73%, so lacking accuracy likely wasn't to blame for attack failure. One notable difference between the Vision Transformer training I used and those used by Geiping et al. (2021) for evaluating Witches' Brew was that I used Mixup and Cutmix for augmentation. As discussed in section 6.5, those augmentations are already individually known to increase robustness against test-time attacks. Geiping et al. (2021) found that other data augmentations can reduce poison success rate to 32.5% from 100% otherwise, which additionally suggests that an augmentation like them could contribute to poisoning attack robustness. To start isolating the effects of heavy augmentation, in the next experiment I set aside Vision-Transformer-style training in favor of that used for ResNets as described at the beginning of this subsection.

8.1.4 Light Augmentation Poisoning

Now that I was using smaller images, I increased the image budget to 1% to be more commensurate with a different selection of Geiping et al. (2021)'s results. Previously, I had used fewer due the long time it took to perform experiments with large images. For benchmarking reasons, I changed the classification task of the victim to CIFAR-10. I set the target image to be the dog of Figure 9 and set the target class to `FROG`. Again using Witches' Brew with no modifications, I set the poison magnitude constraint to $\epsilon = 32$. I evaluated the poisoned dataset $X'_P \cup X_C$'s effectiveness a total of 8 times by poisoning 8 victim models f.

The clean model f_c achieved a validation accuracy of 73% while the poisoned models f achieved accuracies between 70.53% and 74.05%. In all 8 cases, poisoning was successful: $f(x_T)$ assigned y_T the highest probability at the end of training. I observed this by providing the target image x_T as input to each poisoned model, but this is also evident from the attacker loss

$\mathcal{L}_{att}$ curves in Figure 8. Zero loss is only achieved when the model assigns 100% probability to the target class y_T, which is exactly what we see after each of the victim models is trained for as little as 6 epochs. There are two implications of this success. First, I had a baseline from which to increase constraints and improve attack strength using heuristics. Second, I found that relatively high accuracy on CIFAR-10 could be achieved quickly with DeiT-Ti/16 without its recommended heavy augmentations. While discussing subsequent experiments, I will refer back to this one as a baseline and use heuristics to create stronger attacks.

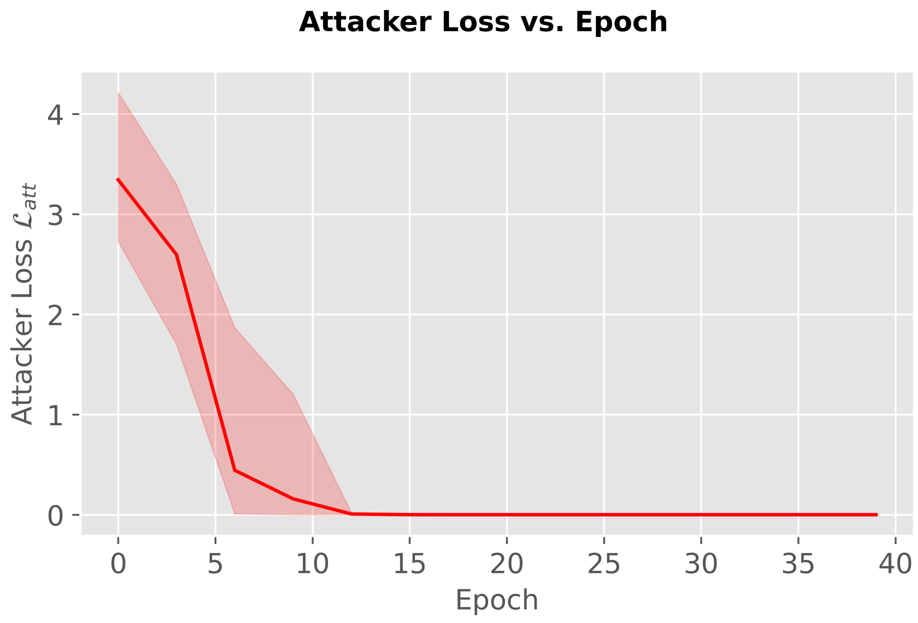


Figure 8: Over 8 attack validation runs of training f with poisoned data $X'_P \cup X_C$ generated by Witches' Brew with no modifications (1% image budget, $\epsilon = 32$), the attacker's loss $\mathcal{L}_{att}$ versus training epoch. The bold red line shows average loss over the validation runs and the shaded region shows the minimum and maximum losses at each epoch. Attack success is evidenced by achieving 0 loss after a small number of epochs.

8.2 Model Training Procedure

With cause to search for stronger attacks targeted at Vision Transformers, I standardized the training procedure for clean models f_c and poisoned validation models f. In pursuit of discovering computationally inexpensive attacks, I immediately introduced constraints on patch count k that may be poisoned. For the purpose of studying the effect of patch coverage as a percentage of entire poison images, I evaluated poisoning models with various input image sizes. For each of image sizes 32×32, 64×64, and 96×96 pixels, I trained one clean model f_c. I used each of these three models' parameters for all subsequent poisoning experiments, rather than training a new one each time.

In all cases, I fine-tuned from pre-trained weights to classify CIFAR-10 (Krizhevsky, 2009), which consists of 60,000 images of size 32 by 32 pixels and 3 color channels. It includes 50,000 images for training complete with labels categorizing them into 10 classes. I use its 10,000 validation images as a repository of test images from which to draw attacker objectives (x_T, y_T). For use with DeiT-Ti, pre-trained for ImageNet's 1,000 classes, I made multiple modifications:

1. Images were upscaled as needed using bilinear interpolation to slightly larger sizes 64×64 or 96×96.

2. Since the number of input tokens was changed, I changed the token position embeddings using bicubic interpolation.

3. I randomly re-initialized the last layer, responsible for classification, to output a probability distribution over 10 classes instead of 1,000.

I allowed all network weights to change during training, which may be described as re-training instead of fine-tuning.

Training itself lasted 40 epochs and used the AdamW optimizer with a learning rate schedule. I randomly applied augmentations such as mirroring images and cropping. The learning rate schedule was multi-linear: it started with $\lambda = 0.001$ and the rate was reduced by $\frac{1}{10}$ after 15, 25, and 35 epochs. In between these epochs, the schedule was locally linear in epoch for a smooth change.

8.3 Evaluation Criteria

I evaluated the effectiveness of attacks by considering three factors.

1. Attacks should be stealthy in not greatly degrading the accuracy of victim models f apart from the target image x_T. So, I compared the top-1 accuracies of clean models and corresponding poisoned models. I expected the poisoned models to have somewhat degraded accuracy, but for a stealthy attack such reduction should not be much greater than those obtained by other poisoning attacks such as Poison Frogs (Shafahi et al., 2018) or Witches' Brew on ResNets (Geiping et al., 2021). Such attacks observed a worst-case reduction on the orders of 0.4% and 0.1% respectively. I did not observe stability on this scale in final validation accuracy, even among clean models. So, I define a looser stealth success criteria that is met if poisoned models' validation accuracies do not drop more than 5 percentage points compared with clean models.

2. I checked the probability distribution output by the poisoned models f when evaluated on x_T as the course of training progressed. The attacker intends y_T to have the highest probability, as opposed to x_T's true label or some other class entirely. The main goal of the attacker is for $f(x_T)$ to output a y_T-favorable distribution after training is complete, but I also evaluated how the distribution evolves over the course of training. There is then a spectrum of probability shift success corresponding to how great of probability f assigns to the y_T class at various epochs and how early during training y_T is assigned greatest probability, if at all. I assessed this shift qualitatively by inspection of loss and probability versus epoch curves. For more rigorous comparisons of probability shift, I calculated the average attacker loss $\overline{\mathcal{L}_{att}}$ over all validation runs of a poison set X'_P at the end of training victim models f. See section 7.1 for a reminder of how that is calculated for each validation run. Since this is a loss, stronger attacks will achieve less loss by the end of training.

3. Finally, and most importantly for the attacker, $f(x_T)$ should assign greatest probability to label y_T after training on poisoned data. As opposed to probability shift success, this labeling success criteria requires that the model give the most probability to y_T out of all other labels at the end of training, rather than simply assign higher probability than a

clean model at various epochs. I evaluated this success by simply providing x_T to f after training and observing whether or not y_T was assigned the highest probability of any class. Since I often used the same brewed poison to attack multiple random instantiations of victim models f, I specifically evaluated labeling success rate (LSR) which is a ratio of how many models meet the attacker's labeling objective at the end of training over the total number of poisoned victim models trained.

For the following experiments, I do not always explicitly state which of these success criteria were met. Where not otherwise stated, I implicitly attest stealth success. For probability shift success and labeling success, if the latter is achieved I am generally less inclined to mention the former; labeling success necessarily implies probability shift success. When tie-breaking attack effectiveness, I will share more quantitiative details than simply LSR, including average attacker loss $\overline{\mathcal{L}_{att}}$.

In addition to isolated per-experiment criteria, I evaluated attack strength according to how strong of constraints could be imposed before labeling failure or weakened probability shifting. For instance, in section 8.4.4 I found that a combination of G-IS and AR-PS, which I will abbreviate AR-PS+G-IS, allows labeling success with a difficult constraint of $\epsilon = 32$ and $k = 8$ (out of $P = 16$ total patches) while attacks with R-IS can only achieve consistent success in general with $k = 12$ or higher. This suggests that this combination creates a stronger attack than AR-PS+R-IS, which can only achieve success at that constraint for some poison-target combinations.

8.4 ACT Method Evaluation

Previously, I summarized results conducting attacks on DeiT-Ti/16 using Witches' Brew with no modifications. In preparation for evaluating my ACT methods in general, I described the model training procedure and evaluation criteria. Finally, here I will describe my experiments to assess various heuristics and discuss the results.

For each of these experiments, there are six variables that may change. The first three are constraints: the number of poison images $n = |X_P|$ (the poison budget, often expressed as a percentage of the full training data X), the number of patches that may be poisoned per image k, and the constraint on adversarial noise magnitude ϵ (the ϵ-constraint). The next two variables are which patch selection and image heuristic are used as described in section 7. These may be random or not present, but I already evaluated the extreme case where the method is equivalent to Witches' Brew in section 8.1.4. I will abbreviate both these heuristic variables together on occasion with the patch selection heuristic first and then the image selection, for instance AR-PS+G-IS denotes using attention rollout patch selection and gradient image selection together. The sixth and final variable is the attacker objective, specified by target test image x_T (drawn from CIFAR-10 validation images) and target class y_T. I will sometimes state the objective in `POISON-TARGET` form, which indicates the class y_T that poison data belong to and the true class of x_T. For most experiments, I selected x_T to be the picture of a dog in Figure 9 and chose a subset of images of frogs from the training set to be poisoned ($y_T = $ `FROG`). I will specially indicate when a different attacker objective than this `FROG-DOG` was used.

Figure 9: The target image x_T of a dog that the attacker aims to be classified as a frog.

I first evaluated each patch selection heuristic with random image selection.

8.4.1 Universal Gradient Patch Selection

For this experiment, I used the UG-PS heuristic, which means that the same patch locations were modified in each of the poisoned images X'_P. See Figure 10 for an example of images with patches selected. The images to poison X_P themselves were selected randomly, so this experiment was to evaluate the combination UG-PS+R-IS. I used the original size of CIFAR-10 images, 32×32 pixels, which corresponded to 4 patches per image. I performed 2-5 validation runs for each of several values of k. In each validation run I trained f on poisoned data $X'_P \cup X_C$ as generated by my ACT method. I selected images X_P randomly within a 1% poison budget. Among those images, I selected $k \in \{1, 2, 3\}$ patches with the universal gradient heuristic. I constrained adversarial noise magnitude to within $\epsilon = 32$.

k	LSR
1	0/5
2	0/2
3	5/5

Table 2: Using the universal gradient patch selection heuristic with $\epsilon = 32$ and a 1% image budget with random image selection, there is a successful attack only when $k = 3$ out of 4 total tokens.

For $k = 1$ and $k = 3$, I retrained 5 times each and observed labeling success in 0 and 5 runs of training victim models f respectively. I performed two validation runs with $k = 2$ but didn't observe labeling success in either of them. I only performed two validation runs in this case due to code interruption by a Windows update. See Table 2 for a summary.

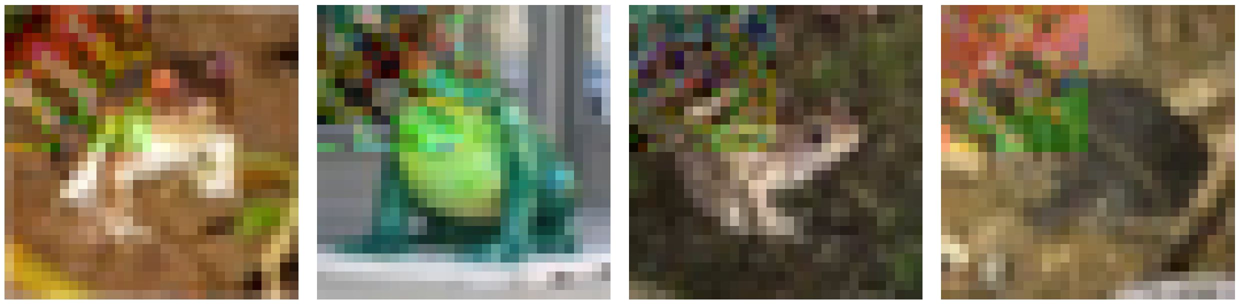


Figure 10: When using UG-PS, the same patch is poisoned in every image. Here are some examples of poisoned images in X'_P brewed with $k = 1$ and $\epsilon = 32$. The heuristic chose the top-left patch to be poisoned in all these 32×32 images, evident from slight discoloration in that quadrant.

This result establishes that a patch-constrained poisoning attack can achieve labeling success, but may require substantial coverage of an image, in this case 75% of the pixels. In subsequent experiments, I evaluated heuristics that are theoretically stronger due to per-image application. In the process, I investigated whether patch count k or percent coverage of an image k/P is more diagnostic of attack success rate.

8.4.2 Individual Gradient Patch Selection

The last experiments showed hope for improved performance. In the next experiments, I up-scaled CIFAR-10 images to 64×64 for 16 patches total. This was to better identify any relationships between image poison coverage and attack success rate, given that more fine-grained control of coverage percentage is possible with larger images. Again, I used a 1% poison budget with an $\epsilon = 32$ constraint. I selected images randomly but used the IG-PS heuristic. See Figure 11 for an example of patch coverage varying from image-to-image.

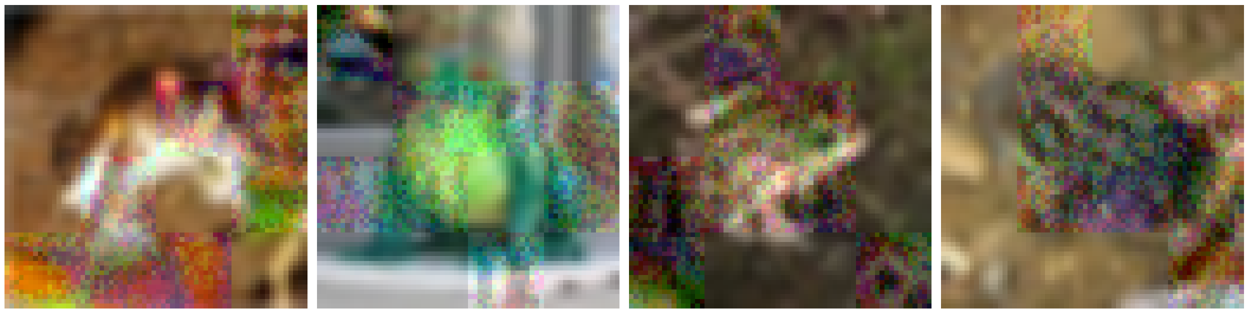


Figure 11: Now using 64×64 images, I brewed poison constrained to $k = 8$ patches selected by IG-PS so that specific patches could differ image-to-image. An $\epsilon = 32$ constraint was used and poison images X_P were selected randomly. This figure shows a some examples of poisoned images.

I created 3 poisoned models f for each of $k = 8$ and $k = 12$ constrained poison trials, which corresponded to 50% and 75% coverage respectively. Only the latter achieved labeling success in all 3 trials. The former did not achieve labeling success at all, a result consistent with percent coverage of an image determining attack success cross-heuristic. In the $k = 8$ experiments, the probability assigned by the models f to x_T belonging to y_T remained low throughout training; there was little or no probability shift success. However, the success of an attack with $k = 12$ was evidenced by high probability by the end of training. See Figure 12 for how the probability assigned to x_T by poisoned models f changed over the course of training.

This result suggests that poison coverage of an image $\frac{k}{P}$, for P total patches, as opposed to

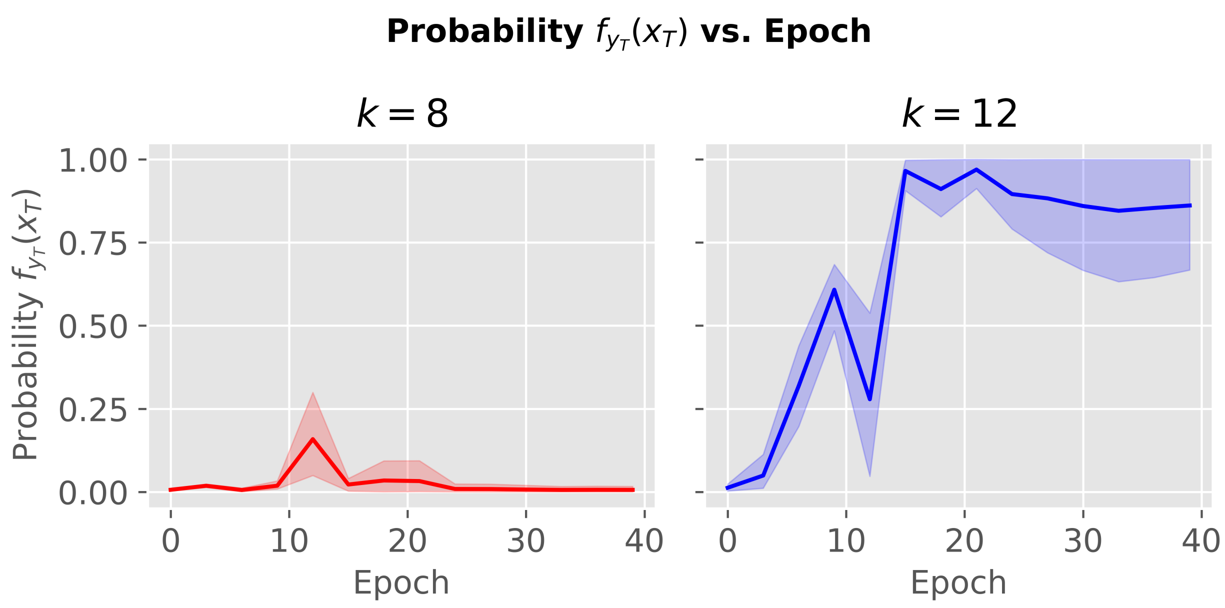


Figure 12: The probability assigned by attacked victim models f to the target image x_T belonging to the target class y_T across training. I show average lines in the center while the extremes of the shaded region are maximum and minimum probabilities at given epochs. Left, using a constraint of $k = 8$ of 16 total patches, the attack achieved neither labeling nor substantial probability shift success in all three validation runs. With $k = 12$, right, labeling success was achieved by the attack in all three runs.

raw number of tokens k, is an important factor for attack success. In the next experiments, I investigated this more with the attention rollout heuristic.

8.4.3 Attention Rollout Patch Selection

Next, I evaluated using a heuristic not only inspired by the need for stronger attacks on Vision Transformers, but that is also Transformer-specific: AR-PS. First, I used unscaled 32×32 CIFAR-10 images with a 1% poison budget (randomly selected) and $\epsilon = 32$. First, I used a coverage constraint $k/P = 3/4$ and observed labeling success in 2/3 poisoned models. This suggests that attention rollout is slightly weaker than gradient as a patch-selection heuristic. In subsequent experiments, I upscaled the images to 64×64 and performed poison attacks with objectives having different poison-target combinations than `FROG-DOG`.

After scaling the images to have 16 patches total, I used the same heuristic with random image selection and $k = 12$. I created poison with the ACT method given the same attacker objective for f's classifying a dog in the test set as a frog. I evaluated success by training 6 poisoned models. I observed labeling success in 4 cases, consistent with the LSR observed with smaller images at this coverage using other patch selection heuristics. Similarly, with the same setup except $k = 8$, I observed no labeling success across 3 trials. These results further suggest that LSR depends on the fraction of an image covered by poison k/P, rather than patch count k independent of image size.

Next, I evaluated the same attack heuristics but with different choices of target image and poison class. Success was highly variable. I observed labeling success for $k = 12$ in 3/3 trials for `CAT-AUTOMOBILE` and `DOG-FROG` (the reverse of before), but 2/3 labeling success for `BIRD-AIRPLANE`. To my intuition, `BIRD-AIRPLANE` and `FROG-DOG` involve classes more similar to each other than `CAT-AUTOMOBILE`. Diminished efficacy on these similar pairs suggests that lack of similarity between true and target classes of target images x_T increases success rate. However, the `DOG-FROG` pair breaks this pattern. In any case, success rates and degrees thereof vary with choice of target image and poison class. See Table 3 for details.

Returning to the not-yet-overcome constraint $k = 8$ for the `FROG-DOG` objective, I attacked with other poison-target combinations like `CAT-AUTOMOBILE`. With a $\epsilon = 32$ magnitude constraint, this method can now achieve labeling success as shown in Figure 13. I chose validation runs for the figure to represent the wider variety of poisoned training runs I observed. The `BIRD-AIRPLANE` run displayed in the figure briefly succeeded mid-training, but then f regained confidence in the true class. For `CAT-AUTOMOBILE`, the mid-training period saw multiple flips between the true class `AUTOMOBILE` and the poison class `CAT`, but the attack was ultimately successful. On the other end of the spectrum, attacking with the `DOG-FROG` objective didn't fair much better than what I saw earlier with `FROG-DOG` and individual gradient patch selection (Figure 12). All these cases show that attack success largely depends on objective choice.

Having thoroughly explored the $\epsilon = 32$ case, I tightened the magnitude constraint to $\epsilon = 16$ to assess how strong of an attack the AR-PS heuristic yields. I trained validation poisoned mod-

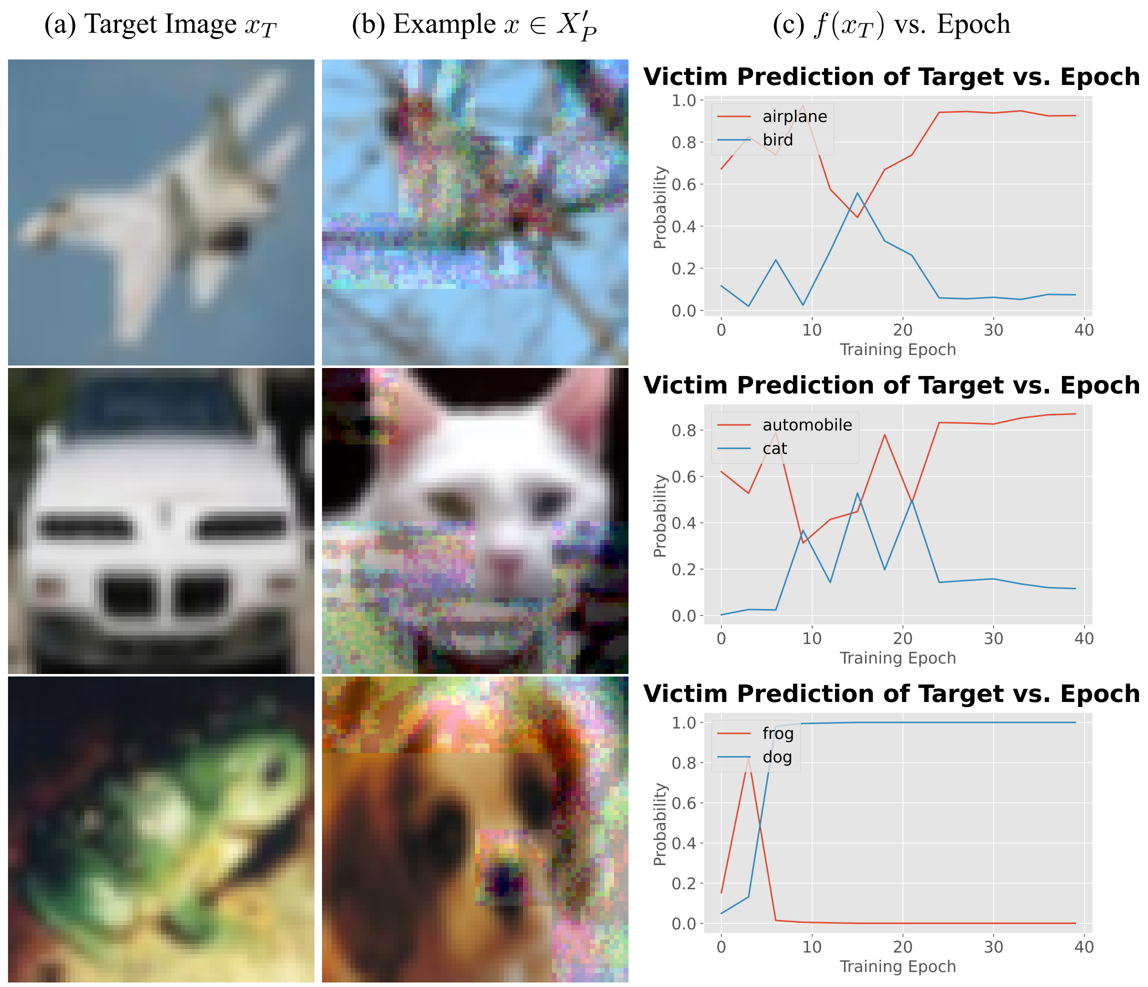


Figure 13: Sample target images, a sample of corresponding poisoned training images, and an example of probability distribution shift predicting the target image x_T class as the victim model f is trained on poisoned data crafted with the AR-PS+R-IS heuristic combination. Constraints used in brewing were ϵ=32, k=8, and a 1% poison image budget. The poisoning attacks' effectiveness was evaluated over 3 validation training runs of f's. For the single validation runs represented above by column (c), only the attack with the frog x_T achieved labeling success. However, each of these attacker objectives achieved at least one labeling success run recorded in Table 3.

els f in this case with all the rest of the constraints held constant to $k = 12$ and a 1% poison budget. The dependence on choice of pair was even more exaggerated in this case, see Table 3 for all details. With $k = 12$ patches, labeling success over 3 trials varies from none of the poisoned models meeting the attacker objective to all of them meeting it. In cases like this, it is valuable to assess probability shift success.

In Figure 14, I show the probability $f_{y_T}(x_T)$ during various epochs during three poisoned training runs. The attacker A wants f to classify x_T, really an airplane (red), as a bird (blue) with maximum probability. Across all poisoned training runs, there was spike in f's assigning more probability to the image being a bird near epoch 15, but in one of the runs the true label prediction quickly recovers. Probability shift success was still achieved in that case; $f_{y_T}(x_T)$ is still a sizeable 20%.

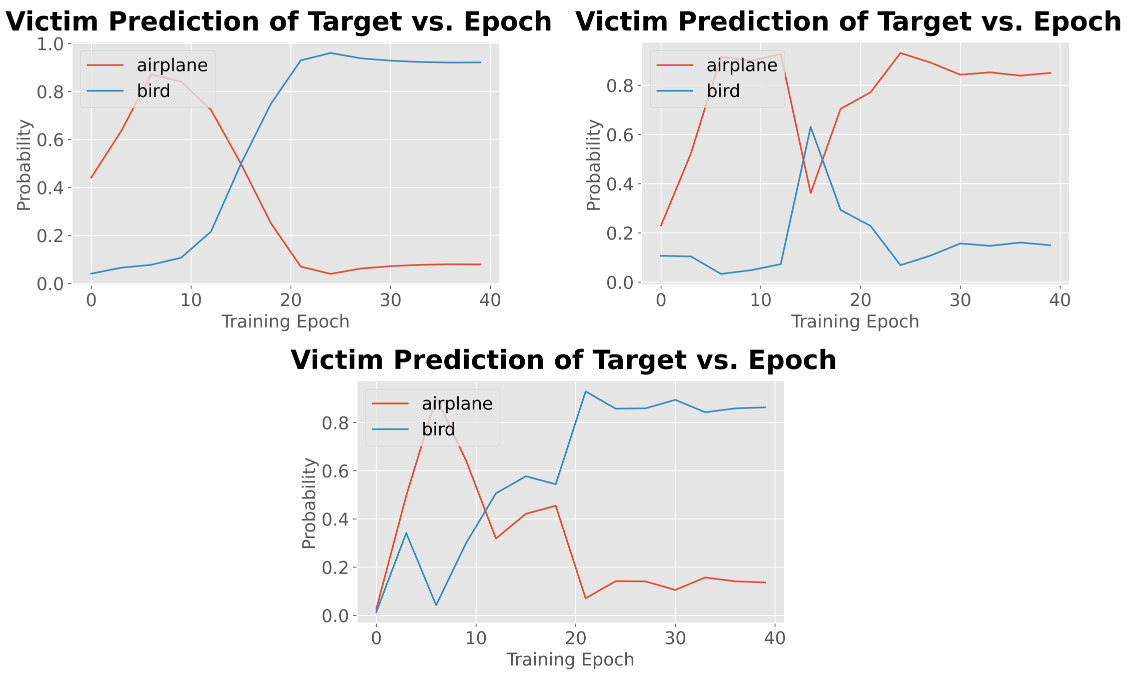


Figure 14: With $k/P = 12/16$ and $\epsilon = 16$, I show three training runs poisoned with a 1% budget with adversarial noise crafted by the ACT method with AR-PS+R-IS. The attack aimed to induce an airplane's picture be classified as a bird, see Figure 13 for the precise airplane. These figures show the probability assigned to the `BIRD` and `AIRPLANE` class by $f(x_T)$ during various epochs of f's fine-tuning. The top-right plot shows a failed poisoning attempt that still shifts the probability of y_T above 0.

In summary, success of this patch selection heuristic with different values of k can vary significantly with choice of target image and poison class.

Finally, I returned to the old `FROG-DOG` objective combination with $\epsilon = 32$. I relaxed the constraint on poison budget to be 2%. For $k = 8$, labeling success was achieved in 2 out of 3 trials, whereas for $k = 4$ no labeling success was observed over 3 trials. These results are not shown in the table, but see Figure 15 for a probability distribution $f(x_T)$ versus training epoch observed with $k = 4$. This representative sample shows that not even probability shift success was achieved in any meaningful sense in this 25% image coverage case. However, I observed labeling success for the first time on the `FROG-DOG` task with 50% coverage. This suggests that increasing the number of images in the poison set can increase effectiveness, but not as much as increasing the number of poisoned patches per image. That is, constraining the number of patches per image is a more difficult challenge for an attack to overcome than a constrained poison budget. With this observation, I shifted focus to experiments to evaluate the effectiveness of image selection heuristics.

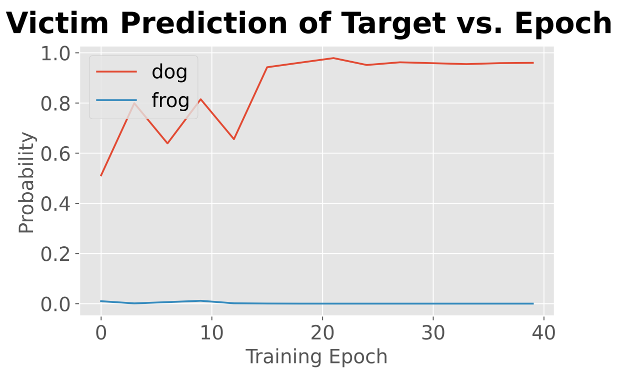


Figure 15: Using AR-PS with $k = 4$ and $\epsilon = 32$, attacking with the `FROG-DOG` objective, and randomly choosing a 2% poison budget, here is a representative sample of the probability distribution shift of $f(x_T)$ across poisoned training epochs. The attacker's goal is for x_T to be classified as a frog, but this was not achieved in any meaningful sense.

8.4.4 Gradient Image Selection

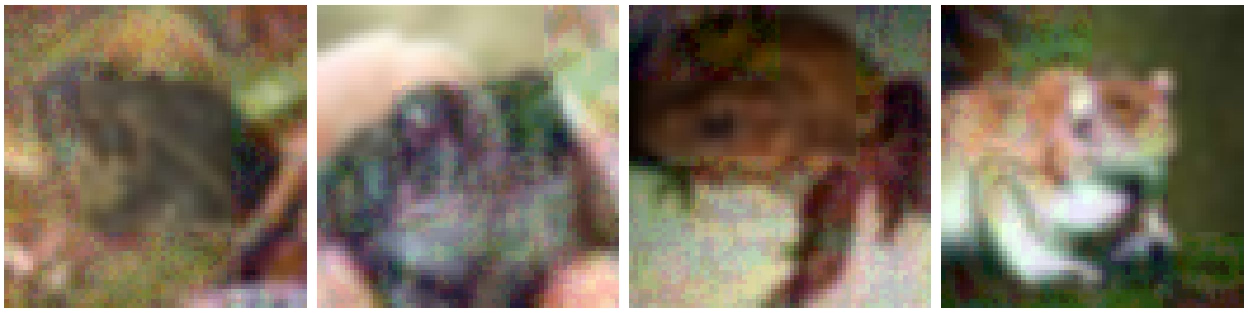


Figure 16: A sample of images selected by the gradient heuristic, $k = 12$ patches within them selected by attention rollout, and then those patches poisoned within $\epsilon = 16$ magnitude.

k	ϵ	POISON-TARGET	AR-PS+R-IS LSR	AR-PS+R-IS $\overline{\mathcal{L}_{att}}$	AR-PS+G-IS LSR	AR-PS+G-IS $\overline{\mathcal{L}_{att}}$	G-PS+G-IS LSR	G-PS+G-IS $\overline{\mathcal{L}_{att}}$	AR-PS+GAS-IS LSR	AR-PS+GAS-IS $\overline{\mathcal{L}_{att}}$
12	32	FROG-DOG	4/6	2.117	3/3	0.001	3/3	0.004	3/3	**0.000**
		CAT-AUTOMOBILE	3/3	0.002	3/3	**0.000**	3/3	**0.000**	3/3	**0.000**
		BIRD-AIRPLANE	2/3	0.748	3/3	0.004	3/3	**0.000**	3/3	**0.000**
		DOG-FROG	3/3	**0.000**	3/3	**0.000**	3/3	**0.000**	3/3	**0.000**
12	16	FROG-DOG	0/3	6.727	3/3	0.154	1/3	2.307	3/3	**0.011**
		CAT-AUTOMOBILE	1/3	1.436	2/3	0.244	3/3	0.038	3/3	**0.004**
		BIRD-AIRPLANE	2/3	0.709	3/3	**0.000**	3/3	0.062	3/3	**0.000**
		DOG-FROG	3/3	**0.000**	3/3	**0.000**	3/3	**0.000**	3/3	**0.000**
8	32	FROG-DOG	0/3	6.201	3/3	**0.015**	3/3	0.243	2/3	0.594
		CAT-AUTOMOBILE	1/3	1.388	3/3	**0.000**	3/3	0.056	3/3	0.088
		BIRD-AIRPLANE	1/3	1.786	2/3	0.284	3/3	0.011	3/3	**0.001**
		DOG-FROG	3/3	**0.000**	3/3	**0.000**	3/3	**0.000**	3/3	**0.000**

Table 3: Quantitative comparisons of poisoning attack strengths with ACT methods using various heuristic combinations and attacking under a variety of constraints. All experiments used a 1% poison budget and $64{\times}64$ CIFAR-10 images which have $P{=}16$ patches. The best average attacker loss achieved at the end of training is highlighted in bold.

For my next experiments, I used G-IS to choose images in X_P. I also used one of AR-PS or G-PS for patch selection. I had two goals for these experiments: determine whether deliberate image selection increases attack strength and determine which patch selection heuristic works best with G-IS. To meet the first goal, I used the same patch constraints k and poison magnitude constraints ϵ that I used in the previous experiments to evaluate AR-PS+R-IS.

For all experiments, I upscaled CIFAR images to 64×64 and set a 1% poison budget. I brewed poison under several combinations of $k \in \{8, 12\}$ and $\epsilon \in \{16, 32\}$, tightening constraints to evaluate how strong attacks with this image selection and the two patch selection heuristics are. For each, I evaluated effectiveness over 3 poisoned models. I used the familiar `FROG-DOG` poison-target attacker objective combination as well as `BIRD-AIRPLANE`, `CAT-AUTOMOBILE`, and `DOG-FROG`.

In summary, labeling success was achieved in nearly all trials, see Table 3 for details. While attacks with AR-PS alone struggled to achieve labeling success, even failing to achieve probability shift success in some trials of $k = 4$ and $\epsilon = 32$, incorporating image selection leads to very consistent labeling success. Among 8 experiments in which using AR-PS versus G-PS had different $\overline{\mathcal{L}_{att}}$ at the end of training that I can compare, in 5 experiments AR-PS had more probability shift success by that measure. This suggests that AR-PS has a slight probability shift effectiveness advantage over G-PS when deliberate image selection is being used, contrary to the pattern observed with R-IS. In any case, this strong success suggests that image choice is more important than patch selection.

8.4.5 GAS Image Selection

At this point, the next image selection heuristic to evaluate was GAS-IS. Using the familiar setup of 1% poison budget, upscaled CIFAR-10 images to 64×64, and three validation runs per poisoned image set X'_P, I collected the results in Table 3.

For the selection of attacker objectives and constraints I evaluated, ties were somewhat frequent. Comparing to AR-PS+G-IS, of the 7 experiments in which one attack had a lower $\overline{\mathcal{L}_{att}}$ than the other, this image selection with AR-PS was stronger in 5. Comparing to G-PS+G-IS, of the 6 experiments with a disparity in average attacker losses, AR-PS+GAS-IS was stronger in 4 experiments. These results suggest that AR-PS+GAS-IS has a slight attack strength advantage over AR-PS+G-IS, which makes it at least competitive for the best heuristic combination in ACT methods.

Unlike earlier experiments, but like the others represented in Table 3, a tighter restriction on k is easier to overcome than a tighter restriction on ϵ; compare the $k = 12$ and $\epsilon = 32$ losses with the other constraint combinations. More experiments are needed to draw a strong conclusion about relative difficulty. For now, the evidence is inconclusive.

8.4.6 Mixup and Cutmix Defense

In my final four experiments, I isolated the effects of Mixup and Cutmix augmentations in the case of CIFAR-10 classification. While my preliminary study found that Witches' Brew failed to successfully poison DeiT-Ti/16 under recommended training hyperparameters as it learned to classify CIFAR-100, here I will explain how I isolated the effect of Mixup, Cutmix, and label smoothing. To be commensurate in various ways with both my preliminary experiments as well as those I performed for the broader class of ACT methods, I used a 0.1% poison budget, an $\epsilon = 32$ magnitude constraint, and used the attacker objective FROG-DOG for all experiments. For a fair comparison with Witches' Brew (WB) once more, I did not use patch constraints, opting instead to only use image selection. As usual, I performed three validation runs per poisoned image set as brewed with a specific image selection heuristic and using the augmentations or not.

For each image selection heuristic R-IS, G-IS, and GAS-IS I brewed a poisoned set X'_P with models using the mix augmentations or not. For those not using the mix augmentations, they used the same light augmentations described in section 8.2 above. In those experiments using Mixup and Cutmix, 100% of training images were batch-wise applied either Mixup or Cutmix, each having equal probability. Mixup was limited to $\alpha \leq 0.8$ while Cutmix was allowed unlimited cutting and pasting with appropriate label interpolations. Labels were also smoothed so that each training label would have at least 0.1 in each class.

Even with my strong ACT method, labeling success with such few images was lacking. So, I focused on probability shift success as captured by the quantitative average attacker loss $\overline{\mathcal{L}_{att}}$ measure. Note that the attack using GAS-IS against a victim using the mix augmentations was only validated twice due to the job it was part of running out of time on the Talapas cluster. I found that GAS-IS provides the greatest reduction in loss compared to Witches' Brew for both augmentation cases, but it is still not strong enough to create a successful attack in this setting. With only a 0.1% poison budget, only the attack using GAS-IS against a victim not using mix augmentations achieved labeling success in 3 of 3 trials. None of the rest of this batch achieved

$\overline{\mathcal{L}_{att}}$	R-IS (WB)	G-IS	GAS-IS
No Mix Augmentations	8.652	3.047	**0.197**
Mix Augmentations	4.442	3.789	**3.130**

Table 4: Average attacker loss at the end of training for a variety of image selection heuristics against a victim using Mixup/Cutmix/label smoothing or not. The best loss for each augmentation configuration is bolded. Here, Witches' Brew (WB) is equivalent to using R-IS as in the first column.

labeling success even once. See Table 4 for details.

There are two important takeaways from these results. First, in light of the slight advantage that GAS-IS already had in previous experiments, its performance here solidifies it as the best image selection heuristic I evaluated. Second, the Mixup, Cutmix, and label smoothing augmentations in combination provide an effective defense against ACT methods. Using strong heuristics, as opposed to random image selection as in Witches' Brew, improves probability shift success, but in this environment is not enough to achieve labeling success.

9 Discussion and Conclusion

While introducing this work, I presented three questions to investigate. Through extensive experiments, I have answered all three. Additionally, I discovered that the previous SOTA data poisoning attack method, Witches' Brew, is weakened when poisoning Vision Transformers compared with other computer vision architectures. I developed a class of poisoning attack methods, ACT methods, to target Vision Transformers and exploit their image patch tokenization in a novel way with an attention rollout heuristic. I evaluated a variety of heuristic combinations, progressively tightening constraints to measure attack strength. The strongest ACT method I evaluated used GAS image selection in combination with attention rollout patch selection. Future directions of study include evaluating the effectiveness of this attack on larger PFMs, different machine learning tasks, and further studying the comparitive effects of constraints on k, ϵ, and n. Through a careful literature review, my preliminary study of unmodified Witches' Brew, and experiments to isolate the augmentation combination Mixup+Cutmix+label smoothing, I found for the first time (to my knowledge) that such training augmentations promote robustness against train-time attacks.

Mixup and Cutmix are already known to be effective at improving robustness against test-time adversarial examples (Yun et al., 2019; Zhang et al., 2018). In my evaluation of training with heavy Transformer-style augmentations including both, I observed consistent attack failures until such heavy augmentation was removed. Given the results here, I recommend heavy augmentation such as Mixup and Cutmix for defending against poisoning attacks. To directly answer my question of how Vision Transformers may be defended against train-time attacks: use heavy augmentations like Mixup and Cutmix as are generally recommended for Transformer training.

My first question was "Are Vision Transformer classifiers vulnerable to fast train-time attacks optimized by exploiting tokenization of images?" I found that DeiT-Ti/16 is vulnerable, at least when lighter augmentations are used. Reliable labeling success is possible with constraints on patch coverage up to 50% as long as patch selection is accompanied by deliberate image selection. The coverage of images determines attack success rate stronger than the number of patches k poisoned alone. This success allows faster attack computations in theory by constraining patches poisoned, but my personal implementation of the ACT method did not leverage this speedup. I answered the next question of which heuristics can lead to more effective train-time attacks while I was answering this one through extensive experiments.

I studied three patch selection heuristics and found the most consistently high LSR with individual gradients, if deliberate image selection is not used. I studied AR-PS most in-depth and found that the success of using it across a variety of problem setups is highly dependent on choice of target image and poison class. When image selection heuristics were introduced, attacks choosing patches by AR-PS achieved slightly more probability shift success. The Vision Transformer DeiT-Ti/16 proved to have a natural robustness against the previous SOTA for single-target attacks Witches' Brew, but the strength of my ACT method attacks using heuristics is enough to observe high attack success rate despite this robustness.

I observed the highest LSR when image selection heuristics were used in combination with patch selection, specifically GAS-IS and AR-PS. This strength allowed attacks with tight constraints on ϵ and k to be not only be successful, but minimize the attacker loss $\mathcal{L}_{att}$ the most by the end of training. In future work, adaptations of the same image and patch selection heuristic combination I used may yield similarly strong train-time attacks in different settings such as multi-target attacks. For now, I contribute a new class of data poisoning attacks, ACT methods, which are to my knowledge the new SOTA. Given that such successful attacks are possible, I recommend use of heavy augmentations like Mixup and Cutmix when training Vision Transformers.

10 Appendix

10.1 96 x 96 CIFAR-10 Experiments and Results

In addition to poisoning 32×32 and 64×64 images, I performed limited experiments with 96×96 images. Using $k = 27$ for 75% coverage, $\epsilon = 32$, and a 1% poison budget, I only observed a 1/3 success rate. See Figure 17 for an example of a poisoned image under this budget. The clean model f_c trained for poison brewing achieved 86% validation accuracy, almost 10 percentage points higher than the clean model for 32×32 images. This was not degraded more than 5 percentage points by poisoning; stealth success was always achieved. This suggests a future direction of study into the relationship between model validation accuracy and robustness against poisoning attacks. Intuitively, higher validation accuracy suggests that more test images will be classified correctly, including target images x_T. Geiping et al. (2021) found an inverse relationship between poison success (increased by larger ϵ allowance) and validation accuracy. My result seems to extend the inverse relationship pattern to differences in models' ability to achieve high validation accuracy.

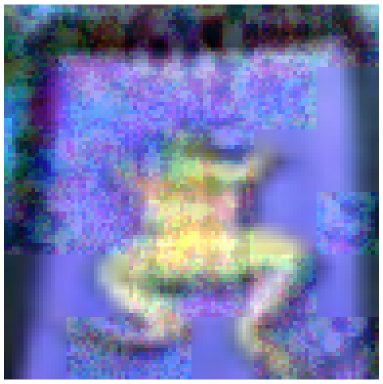
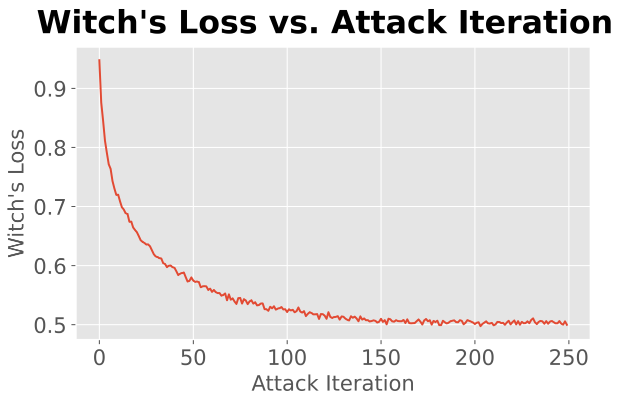


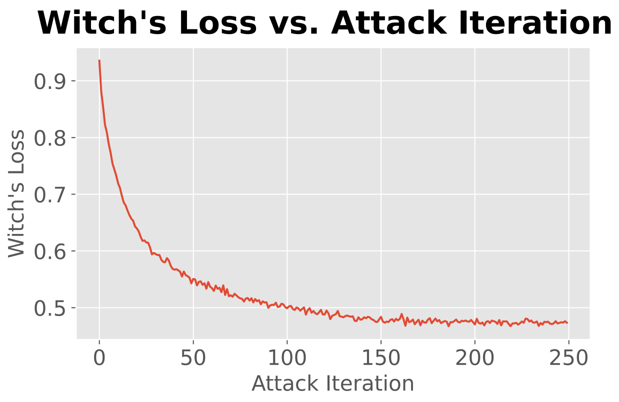
Figure 17: A 96×96 image of a frog, $k = 27$ patches poisoned.

10.2 Poison Brewing Loss Curves

Here, I present a representative sample of attacker surrogate loss $\mathcal{B}$, "witch's loss" to use Witches' Brew terminology, as a function of iteration over the poison generation step.



(a) A brewing run to poison 64×64 images. Using gradient image selection and attention rollout patch selection with constraints $k = 8$, $\epsilon = 32$, and a 1% poison image budget, this brew successfully poisoned 3/3 validation models.



(b) This brewing run to poison 64×64 images had an objective of classifying an airplane as a bird. Using random image selection and attention rollout patch selection with constraints $k = 12$, $\epsilon = 16$, and a 1% poison image budget, this brew successfully poisoned 2/3 validation models.

Figure 18: A representative selection of poison brewing curves: $\mathcal{B}$, the "Witch's Loss", versus attack steps during poison generation. Notice that final loss is not diagnostic of attack success rate; run (b) achieved a smaller final loss than (a) yet had a lower labeling success rate.

11 References

Abnar, S., & Zuidema, W. (2020, May 31). Quantifying attention flow in transformers. Retrieved October 3, 2022, from http://arxiv.org/abs/2005.00928

Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., & Mukhopadhyay, D. (2018). Adversarial attacks and defences: A survey. *arXiv:1810.00069 [cs, stat]*. Retrieved January 4, 2022, from http://arxiv.org/abs/1810.00069

Deng, J., Dong, W., Socher, R., Li, L.-J., Kai Li, & Li Fei-Fei. (2009). ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255. https://doi.org/10.1109/CVPR.2009.5206848

Dong, X., Bao, J., Zhang, T., Chen, D., Zhang, W., Yuan, L., Chen, D., Wen, F., Yu, N., & Guo, B. (2022, December 7). PeCo: Perceptual codebook for BERT pre-training of vision transformers. Retrieved April 16, 2023, from http://arxiv.org/abs/2111.12710

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021, June 3). An image is worth 16x16 words: Transformers for image recognition at scale. Retrieved May 16, 2022, from http://arxiv.org/abs/2010.11929

Geiping, J., Fowl, L., Huang, W. R., Czaja, W., Taylor, G., Moeller, M., & Goldstein, T. (2021). Witches' brew: Industrial scale data poisoning via gradient matching. *arXiv:2009.02276 [cs]*. Retrieved May 3, 2022, from http://arxiv.org/abs/2009.02276

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Gu, J., Tresp, V., & Qin, Y. (2022, July 18). Are vision transformers robust to patch perturbations? Retrieved October 3, 2022, from http://arxiv.org/abs/2111.10659

Hammoudeh, Z., & Lowd, D. (2022). Identifying a training-set attack's target using renormalized influence estimation. *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 1367–1381. https://doi.org/10.1145/3548606.3559335

He, K., Zhang, X., Ren, S., & Sun, J. (2015, December 10). Deep residual learning for image recognition. Retrieved April 16, 2023, from http://arxiv.org/abs/1512.03385

Huang, W. R., Geiping, J., Fowl, L., Taylor, G., & Goldstein, T. (2021). MetaPoison: Practical general-purpose clean-label data poisoning. *NeurIPS*. https://arxiv.org/pdf/2004.00225.pdf

Jagielski, M., Severi, G., Pousette Harger, N., & Oprea, A. (2021). Subpopulation data poisoning attacks. *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 3104–3122. https://doi.org/10.1145/3460120.3485368

Joshi, A., Jagatap, G., & Hegde, C. (2021, October 8). Adversarial token attacks on vision transformers. Retrieved May 16, 2022, from http://arxiv.org/abs/2110.04337

Kingma, D. P., & Ba, J. (2017, January 29). Adam: A method for stochastic optimization. Retrieved April 16, 2023, from http://arxiv.org/abs/1412.6980

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. (2021, August 17). Swin transformer: Hierarchical vision transformer using shifted windows. Retrieved April 16, 2023, from http://arxiv.org/abs/2103.14030

Loshchilov, I., & Hutter, F. (2019, January 4). Decoupled weight decay regularization. Retrieved April 19, 2023, from http://arxiv.org/abs/1711.05101

Lv, P., Ma, H., Zhou, J., Liang, R., Chen, K., Zhang, S., & Yang, Y. (2021, November 22). DBIA: Data-free backdoor injection attack against transformer networks. Retrieved May 16, 2022, from http://arxiv.org/abs/2111.11870

Muñoz-González, L., Pfitzner, B., Russo, M., Carnerero-Cano, J., & Lupu, E. (2019). Poisoning attacks with generative adversarial nets. *arXiv:1906.07773v2 [cs.LG]*. https://arxiv.org/abs/1906.07773

Razmi, F., & Xiong, L. (2021). Classification auto-encoder based detector against diverse data poisoning attacks. *arXiv:2108.04206 [cs]*. Retrieved April 20, 2022, from http://arxiv.org/abs/2108.04206

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015, January 29). ImageNet large scale visual recognition challenge. Retrieved April 16, 2023, from http://arxiv.org/abs/1409.0575

Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., & Goldstein, T. (2018). Poison frogs! targeted clean-label poisoning attacks on neural networks. *arXiv:1804.00792 [cs, stat]*. Retrieved January 19, 2022, from http://arxiv.org/abs/1804.00792

Subramanya, A., Saha, A., Koohpayegani, S. A., Tejankar, A., & Pirsiavash, H. (2022, June 16). Backdoor attacks on vision transformers. Retrieved October 7, 2022, from http://arxiv.org/abs/2206.08477

Sun, C., Shrivastava, A., Singh, S., & Gupta, A. (2017, August 3). Revisiting unreasonable effectiveness of data in deep learning era. Retrieved April 3, 2023, from http://arxiv.org/abs/1707.02968

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., & Jégou, H. (2021, January 15). Training data-efficient image transformers & distillation through attention. Retrieved October 4, 2022, from http://arxiv.org/abs/2012.12877

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. https://doi.org/10.48550/ARXIV.1706.03762

Wang, X., Li, J., Kuang, X., Tan, Y.-a., & Li, J. (2019). The security of machine learning in an adversarial setting: A survey. *Journal of Parallel and Distributed Computing*, *130*, 12–23. https://doi.org/10.1016/j.jpdc.2019.03.003

Xiao, H., Xiao, H., & Eckert, C. (2012). Adversarial label flips attack on support vector machines. *European Conference on Artificial Intelligence*, 6. https://doi.org/https://doi.org/10.3233/978-1-61499-098-7-870

Yu, J., Wang, Z., Vasudevan, V., Yeung, L., Seyedhosseini, M., & Wu, Y. (2022, June 13). CoCa: Contrastive captioners are image-text foundation models. Retrieved April 16, 2023, from http://arxiv.org/abs/2205.01917

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., & Yoo, Y. (2019, August 7). CutMix: Regularization strategy to train strong classifiers with localizable features. Retrieved April 27, 2023, from http://arxiv.org/abs/1905.04899

Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2018, April 27). Mixup: Beyond empirical risk minimization. Retrieved February 11, 2023, from http://arxiv.org/abs/1710.09412

Zhao, M., An, B., Gao, W., & Zhang, T. (2017). Efficient label contamination attacks against black-box learning models. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 3945–3951. https://doi.org/10.24963/ijcai.2017/551

Zhou, C., Li, Q., Li, C., Yu, J., Liu, Y., Wang, G., Zhang, K., Ji, C., Yan, Q., He, L., Peng, H., Li, J., Wu, J., Liu, Z., Xie, P., Xiong, C., Pei, J., Yu, P. S., & Sun, L. (2023, February 18). A comprehensive survey on pretrained foundation models: A history from BERT to ChatGPT. Retrieved March 17, 2023, from http://arxiv.org/abs/2302.09419