Not All Samples Are Equal: Quantifying Instance-level Difficulty in Targeted Data Poisoning

William Xu[*]
University of Waterloo
w262xu@uwaterloo.ca

Yiwei Lu[*]
University of Ottawa
yiwei.lu@uottawa.ca

Yihan Wang[†]
University of Waterloo
yihan.wang@uwaterloo.ca

Matthew Y.R. Yang[†]
Carnegie Mellon University
myang4@cs.cmu.edu

Zuoqiu Liu[†]
Google
robertliu@google.com

Gautam Kamath[‡]
University of Waterloo
Vector Institute
g@csail.mit.edu

Yaoliang Yu[‡]
University of Waterloo
Vector Institute
yaoliang.yu@uwaterloo.ca

Abstract

Targeted data poisoning attacks pose an increasingly serious threat due to their ease of deployment and high success rates. These attacks aim to manipulate the prediction for a single test sample in classification models. Unlike indiscriminate attacks that aim to decrease overall test performance, targeted attacks present a unique threat to individual test instances. This threat model raises a fundamental question: *what factors make certain test samples more susceptible to successful poisoning than others?* We investigate how attack difficulty varies across different test instances and identify key characteristics that influence vulnerability. This paper introduces three predictive criteria for targeted data poisoning difficulty: ergodic prediction accuracy (analyzed through clean training dynamics), poison distance, and poison budget. Our experimental results demonstrate that these metrics effectively predict the varying difficulty of real-world targeted poisoning attacks across diverse scenarios, offering practitioners valuable insights for vulnerability assessment and understanding data poisoning attacks.

1 Introduction

In the past decade, machine learning (ML) models have achieved great success in various domains, largely due to the vast amount of training data available on the internet. However, this reliance on massive training datasets not only increases computational costs but also introduces significant security vulnerabilities during the data collection process [24, 44]. Adversaries can exploit these vulnerabilities through data poisoning attacks which deliberately inject malicious samples into training data either actively or passively [4, 11, 30, 40, 46]. These attacks are particularly concerning because they can compromise model integrity at its foundation, affecting all downstream applications and users of the poisoned model [14].

Targeted data poisoning attacks represent a specialized form of this threat, where attackers aim to manipulate model behavior for specific test instances while maintaining normal performance on

[*]Equal contribution. [†] Listed in Contribution order. [‡]Listed in alphabetical order.

all other inputs [e.g., 2, 13, 16, 38, 49]. We primarily focus on classification models (and briefly discuss generative models in Appendix E), where the objective is to misclassify a particular sample to a predetermined class while maintaining correct predictions for all other inputs. Such attacks are difficult to detect as they leave little evidence in overall model performance metrics.

Current evaluations of targeted attack threats typically rely on randomly selected test samples to report overall attack success rates [e.g., 2, 13] as an average assessment. However, our observations reveal substantial variation in attack effectiveness across different test instances, with no clear understanding of what characteristics drive these disparities. This paper addresses this critical knowledge gap by investigating two key research questions: *(1) what factors determine why a certain test sample is more vulnerable to targeted poisoning attacks than others? and (2) can we develop reliable metrics to predict the difficulty of poisoning a specific instance?*

We address the first question by identifying three critical factors that influence poisoning difficulty: the inherent classification difficulty during clean training, the distance in model parameter space required to achieve poisoning, and the attacker's resource constraints measured by poison budget. To quantify these factors, we introduce three corresponding metrics that naturally predict poisoning difficulty: (1) ergodic prediction accuracy (`EPA`) derived from clean training dynamics, (2) poisoning distance δ, and (3) poisoning budget lower bound τ. Importantly, all our proposed metrics can be calculated *using only clean training data and processes*, without requiring the simulation of actual poisoning attacks—making them practical and accessible tools for defenders to assess vulnerability and prioritize protection efforts.

Our experimental results confirm that all three proposed factors strongly correlate with real-world attack performance. The metrics we developed effectively predict poisoning difficulty across various test instances, capturing different dimensions and levels of vulnerability. Specifically, we find that ergodic prediction accuracy (`EPA`) serves as a powerful indicator for distinguishing between easy-to-poison and hard-to-poison test samples. Meanwhile, poisoning distance δ and budget lower bound τ provide more fine-grained predictions for specific poison classes from complementary perspectives. Together, these three metrics form a comprehensive framework that enables defenders to assess poisoning vulnerability for any given test sample.

In summary, our work makes three distinct contributions: (1) We identify classification difficulty during clean training, parameter-space poisoning distance, and poison budget as the key factors determining targeted poisoning vulnerability; (2) We introduce three corresponding metrics: ergodic prediction accuracy (`EPA`), poisoning distance, and budget lower bound, all calculable using only clean training processes without any expensive attack and retraining; (3) Our experiments demonstrate the effectiveness of these metrics and introduce a framework that enables defenders to clearly identify and understand instance-level difficulty in targeted data poisoning.

2 Background

Threat model and notations: We first specify our threat model and list our notations below.

- **Objective**: We consider an adversary who tries to *alter the prediction* of a specific test sample $\mathbf{x}_t$[0] from the correct class y_t (or target class) to a specific poison class y_p.
- **Attack deployment**: The attacker reaches the objective by injecting a poisoned set $\mathcal{D}_p$ into the clean training set $\mathcal{D}_c$. We assume the defender is training a machine learning model on the merged dataset, i.e., $\mathcal{D}_{tr} = \mathcal{D}_c \cup \mathcal{D}_p$ and deploy the model on a test set $\mathcal{D}_{test}$ that contains $\mathbf{x}_t$.
- **Attacker's knowledge**: We consider a *white-box attack*,[1] where the attacker is aware of the clean training set $\mathcal{D}_c$, the machine learning model architecture and the training scheme utilized by the defender, and the inclusion of the test sample $\mathbf{x}_t$ in the test set.

[0]We note that the attacker cannot change $\mathbf{x}_t$, but indirectly change the model's behavior by deploying $\mathcal{D}_p$ through retraining. We highlight this is a key difference from adversarial examples, which directly modifies x_t without any poisoned set or retraining.

[1]We note that the attacks could possibly be performed in a partially *black-box* fashion, where the attackers need to apply surrogate models and training procedure. However, such attacks suffer a severe performance drop [37]. To ensure the strongest threat is measured from the defender's perspective to ensure maximum security, we consider the *white-box* setting instead.

- **Attacker's budget & Constraint**: We define the attacker's budget as $\varepsilon = \frac{|\mathcal{D}_p|}{|\mathcal{D}_{tr}|}$, i.e., the (relative) percentage of poisoning data. The budget for targeted attacks is usually low, e.g., $\varepsilon = 1\%$. We also set constraints on $\mathcal{D}_p$, where it only contains "clean-labeled" poison data, namely that elements in $\mathcal{D}_p$ are generated by adding human imperceptible noises (e.g., with ℓ_∞ constraints) to clean training images without changing their original labels.
- **Attack evaluation**: We define an attack to be successful when the prediction of the target sample using the poisoned model $f(\mathbf{x}_t; \mathbf{w}_p)$ (see notation below) is equal to the poisoned label y_p. Note that attack success is strictly stronger than misclassification.

Other notation: We denote the clean model parameters (a model f trained only on $\mathcal{D}_c$) to be $\mathbf{w}_c$ and poisoned model parameters to be $\mathbf{w}_p$. Let $\ell(\mathbf{w}, \mathbf{z})$ be our loss that measures the fitness of our model $\mathbf{w}$ on data $\mathbf{z} \in \mathbb{Z}$, e.g., $\mathbf{z} = (\mathbf{x}, y)$. Let $\mathbf{g}(\mathbf{z}) = \mathbf{g}(\mathbf{z}; \mathbf{w}) = \nabla_{\mathbf{w}} \ell(\mathbf{z}; \mathbf{w})$ be the gradient vector with respect to a fixed model $\mathbf{w}$ evaluated at the data $\mathbf{z}$.

Targeted data poisoning: In this paper, we focus on targeted data poisoning attacks [e.g., 2, 16, 38, 49] that affect only specific test samples and discuss other types of poisoning attacks in Appendix B.1. Given a test sample $(\mathbf{x}_t, y_t)$, the problem can be formulated into the bi-level optimization problem below:

$$\min_{\mathcal{D}_p} \ \ell((\mathbf{x}_t, y_p), \mathbf{w}_*), \ \text{s.t.}\ \mathbf{w}_* \in \operatorname*{argmin}_{\mathbf{w}} \ \ell(\mathcal{D}_c \cup \mathcal{D}_p, \mathbf{w}),$$

where the attacker aims to enforce the prediction of x_t to be y_p through crafting and injecting $\mathcal{D}_p$ into the training set $\mathcal{D}_{tr}$. This problem is hard to solve directly as the outer maximization problem depends on $\mathcal{D}_p$ only implicitly through the solution $\mathbf{w}_*$ of the inner problem. Existing attacks consider relaxations of this primal problem, for example, a fixed feature extractor [2, 49] or approximating the gradient of target parameters [38]. As our poisoning difficulty metrics do not depend on the specific design of attack algorithms, we omit the attack details and refer readers to the above references. We note that data poisoning attacks can be further classified according to their attack scheme, specifically, the construction of $\mathcal{D}_{tr}$, and we extend more discussion in Appendix B.2.

Targeted attacks are insidious as they do not cause significant performance degradation (hence harder to detect) while still capable of causing system failure on targeted test instances. Previous works [13, 38, 49] have demonstrated the efficacy of such attacks against deep neural networks, reporting high attack success rates. However, these reported success rates are typically calculated by averaging over randomly selected test samples [13, 37]. This aggregated metric fails to capture the instance-level difficulty of targeted data poisoning attacks, a critical gap we aim to address in this work.

3 Difficulty of Targeted Data Poisoning

To quantify the variance of attack performance on different test instances, we perform an experiment by applying a state-of-the-art attack called gradient matching [13]. We choose the first 100 test samples (based on target id) in the class "plane" of the CIFAR-10 dataset [23] and perform gradient matching to classify them into the nine other (poison) classes (900 attacks in total). For each attack, we perform 8 independent trials with different model initialization and report the attack success rate, calculated over the 8 independent runs, as a histogram for all 900 attacks in Figure 1. Our results show a high variance in the distribution of attack success rate, which highlights the necessity of understanding instance-level poisoning difficulty. We will show in Section 4 that the choice of the target class also introduces high variance.

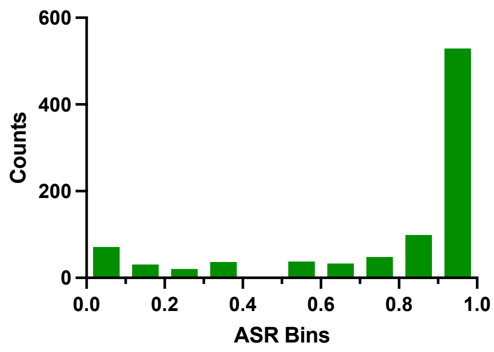


Figure 1: Histogram of the attack success rate of gradient matching over 8 runs on attacking 100 test samples in the class "plane" in CIFAR-10.

In this section, we introduce novel tools and metrics for understanding and quantifying the difficulty of targeted data poisoning. A key strength of our approach is its generality: our tools operate *independently* of specific data poisoning attack implementations and rely solely on *clean-training* dynamics. This attack-agnostic framework enables defenders to assess vulnerability without requiring knowledge of or access to particular attack methodologies. Recall that we denote a clean model as $\mathbf{w}_c$, which we will frequently use in this section.

3.1 Poisoning difficulty prediction with clean training dynamics

Naively, for a linear model such as a support vector machine (SVM), the prediction of a test sample positioned near the decision boundary is likely to be altered by small changes to the model. One might therefore consider using the distance between a test sample and the decision boundary to determine its robustness, or equivalently, its poisoning difficulty. However, this approach falls short for neural networks and fails to account for the complex training dynamics of non-linear models.

To study the poisoning difficulty of neural networks, we propose an intuitive hypothesis, motivated by the linear example above:

Hypothesis A. *The classification difficulty of a test sample* $\mathbf{x}_t$ *is negatively correlated with its poisoning difficulty, i.e., a sample* $\mathbf{x}_t$ *that is easy to classify is correspondingly difficult to poison.*

To verify the above hypothesis, it is necessary to establish a robust measure of the difficulty of classification, which we approach by examining the prediction correctness throughout training:

Definition 1 (Ergodic Prediction Accuracy, `EPA`)**.** *We say the classification difficulty for a target sample* $\mathbf{x}_t$ *can be measured by the ergodic average correctness (denoted by the indicator function) for* N *training epochs with* M *different initializations:*

$$\mathtt{EPA} = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \mathbb{1}\{f_{m,n}(\mathbf{x}_t) = y_t\},$$

i.e., a model with higher `EPA` *is easier to classify, thus harder to attack, and vice versa.*

When the model update is ergodic [34] and with large M and N, `EPA` converges to $\Pr[f(\mathbf{x}_t; \mathbf{w}^*) = y_t]$ where $\mathbf{w}^*$ follows the invariant distribution of the update process.

Although we use the prediction of the model $f_{m,n}(\mathbf{x}_t)$ above, `EPA` can also be calculated using the model's confidence (i.e., logits). We demonstrate the performance of both approaches in Section 4. We acknowledge that `EPA` represents just one method for measuring classification difficulty, and we discuss alternative approaches in Appendix B.3.

While `EPA` provides a convenient measure, it does not fully capture a test sample's robustness against attacks targeting a specific poisoned label y_p. We observe significant variations across different choices of y_p as demonstrated in Figure 2. Furthermore, `EPA` is calculated solely based on clean training and does not account for realistic attack scenarios.

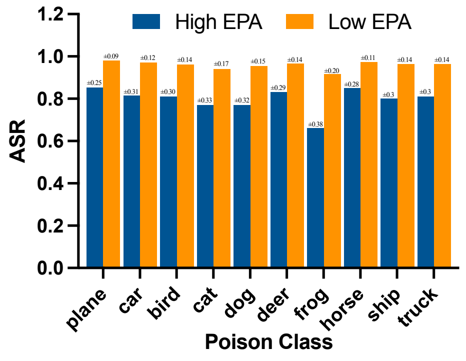


Figure 2: The attack success rates of gradient matching on CIFAR-10 on different poison classes y_p for test samples $\mathbf{x}_t$ with the 50 highest/lowest `EPA`.

3.2 Poisoning difficulty prediction with poisoning distance

To address the above limitations of `EPA`, we provide an alternative tool called *poisoning distance* to measure poisoning difficulty, which takes the choice of y_p into account. Specifically, we consider the ultimate goal of targeted poisoning:

Goal 1*: An adversary aims at modifying model parameters from clean parameters* $\mathbf{w}_c$ *to poisoned ones* $\mathbf{w}_p$ *such that* $f(\mathbf{x}_t; \mathbf{w}_p) = y_p$.

Data poisoning implements an indirect way to achieve this goal through crafting $\mathcal{D}_p$ and training on $\mathcal{D}_c \cup \mathcal{D}_p$. Goal 1 enables us to measure poisoning difficulty by comparing $\mathbf{w}_p$ and $\mathbf{w}_c$ directly. Specifically, we propose a hypothesis on poisoning distance:

Hypothesis B. *The distance $\delta = d(\mathbf{w}_c, \mathbf{w}_p)$ between a clean model $\mathbf{w}_c$ and a (targeted) poisoned model $\mathbf{w}_p$ is positively correlated with poisoning difficulty, i.e., a test sample $\mathbf{x}_t$ with higher δ is correspondingly more difficult to poison.*

Note that $d(\cdot)$ is a distance function that we will specify soon. Here δ is a sample-wise metric as its calculation depends on the sample-specific poisoned parameter $\mathbf{w}_p$. Moreover, Hypothesis B naturally considers the choice of y_p, which is embedded in the definition of $\mathbf{w}_p$.

However, from a defender's perspective, validating Hypothesis B directly is non-trivial as the calculation of δ depends on the poisoned parameters $\mathbf{w}_p$, which are unknown without performing an actual attack. Luckily, data poisoning is not the only viable way to achieve Goal 1, and we propose a proxy to generate $\mathbf{w}_p$ and measure δ without performing any data poisoning attacks:

Definition 2 (Poisoning Distance δ)**.** *Starting from a clean model $\mathbf{w}_c$, we say the poisoning distance is the smallest step size required to modify $\mathbf{w}_c$ in one step such that the model classifies $\mathbf{x}_t$ as y_p:*

$$\delta = \min\{\eta > 0 : \tilde{f}(\mathbf{x}_t; \mathbf{w}_c - \eta \cdot \mathbf{g}) = y_p\},$$

where $\mathbf{g} = \nabla_{\mathbf{w}} \ell(\tilde{f}(\mathbf{x}_t; \mathbf{w}_c), y_p)$. Naturally, we also obtain our proxy of $\mathbf{w}_p = \mathbf{w}_c - \eta \cdot \mathbf{g}$.

While such a proxy may be different from a real data poisoning attack, δ intuitively measures the efforts needed to achieve the attack goal from a gradient perspective[2].

Of course, there exist various algorithms for finding δ. In this paper, we provide a simple δ estimator in Algorithm 1 in Appendix C using binary search. Here we highlight that one advantage of the estimation of δ over `EPA` is that it does not depend on the training process or the clean data $\mathcal{D}_c$, which is extremely handy for users that has access only to the model weights (which is very common for foundation models) to quantify the poisoning difficulty of their own personal data.

3.3 Poisoning difficulty prediction with poison budget

Aside from classification difficulty and poisoning distance, an alternative way to measure poisoning difficulty is through the poison budget ε. It is clear that an attack is easier if less poisoned data is required, i.e., a lower ε suggests an easier attack. Here we aim to answer an intriguing question:

Is it possible to measure the lowest ε needed to poison a model such that a given test sample $\mathbf{x}_t$ is misclassified as y_p, without performing any attacks?

Conveniently, Lu et al. [27] provides valuable theoretical tools for measuring the (relative) number of poisoned samples $|\mathcal{D}_p|$ needed to reach some target parameters $\mathbf{w}_p$ (e.g., the proxy we generated in Definition 2), i.e., the role of poison budget ε. Specifically, [27] provides a lower bound (or necessary condition) with respect to ε on poisoning reachability. Without diving into all technical details and derivations, we directly present a simplified version of their results:

Theorem 1 (Poisoning reachability, Theorem 2 of Lu et al. [27])**.** *Given a classification task with c classes and a set of target parameters $\mathbf{w}_p$, $\mathbf{w}_p$ is poisoning reachable (defined by vanishing gradient over training on $\mathcal{D}_c \cup \mathcal{D}_p$) only if the condition below holds (necessary condition):*[3]

$$\varepsilon \geq \tau := \max\left\{ \frac{\langle \mathbf{w}_p, \mathbf{g}(\mathcal{D}_c) \rangle}{\mathscr{W}(c - 1/e)}, 0 \right\},$$

where $\mathscr{W}(\cdot)$ is Lambert's W function, $\mathbf{g}(\mathcal{D}_c) = \mathbf{g}(\mathcal{D}_c; \mathbf{w}_p) = \frac{1}{|\mathcal{D}_c|} \sum_{\mathbf{z} \in \mathcal{D}_c} \nabla_{\mathbf{w}_p} \ell(\mathbf{z}; \mathbf{w}_p)$.

Theorem 1 enables us to calculate τ, the lower bound of poisoning budget ε for a given target test sample $\mathbf{x}_t$, the corresponding poison class y_p, the target parameter $\mathbf{w}_p$, and the clean training set $\mathcal{D}_c$, In Section 4, we will show that τ is a direct indicator of poisoning difficulty.

[2]We note that our core idea of poisoning distance is closely related to model-targeted indiscriminate attacks and we extend our discussion in Appendix B.4.

[3]Note that Theorem 2 in Lu et al. [27] presents poisoning reachability for binary linear models, and we consider the general form in Equation (10) on multiclass neural networks.

4 Experiments

In this section, we (1) introduce our experimental settings; (2) present our results on poisoning difficulty prediction using `EPA`; (3) demonstrate the effectiveness of the poisoning distance δ and poisoning reachability τ on measuring poisoning difficulty; (4) show our ablation studies on datasets, model architectures, and poisoning budget.

4.1 Experimental settings

We mainly examine the targeted attacks in the unified benchmark provided in [37][4]. Note that we do not consider any backdoor attacks as they require trigger injection during inference and the attack difficulty highly depends on the trigger, and thus are beyond the scope of this paper.

Datasets & models: We consider classification tasks on CIFAR-10 [23] with 10 classes, 50000 clean training samples and 10000 test samples in our main experiments, and TinyImageNet [25] with 200 classes, 100000 clean training samples, 10000 validation samples, and 10000 test samples in our ablation study. We apply ResNet-18 [18] for CIFAR-10 and VGG-16 [41] for TinyImageNet.

Training schemes: We consider two training schemes: (1) Training from scratch, where we initialize the model with random weights. For clean training, we use the clean training set $\mathcal{D}_c$, for data poisoning we use $\mathcal{D}_c \cup \mathcal{D}_p$[5]; (2) Transfer learning for CIFAR-10, where we utilize a frozen model pretrained on CIFAR-100, and fine-tune an additional linear head on a subset of CIFAR-10 that contains the first 250 images per class. For both scenarios, we train the model for 40 epochs.

Targeted attacks: We examine three attack methods listed in the unified benchmark: (1) Gradient matching (GM) [13][6] for training from scratch; (2) Feature collision (FC) [38] for transfer learning; and (3) Bullseye polytope (BP) [2] for both.[7] For training from scratch, we perform 8 random model initializations and calculate the attack success rate (ASR) by dividing the number of successful attacks by 8. For transfer learning, as the model initialization is mostly fixed, we only consider one attack trial each. For all attacks, unless specified otherwise, we use a poisoning budget $\varepsilon = 1\%$.

Measuring poisoning difficulty: (1) To calculate `EPA` for each test sample, we train the model with $M = 100$ (for CIFAR-10), and $M = 8$ (for TinyImageNet) random initializations for $N = 40$ epochs. We consider the model prediction for training from scratch and model confidence for transfer learning; (2) To obtain δ, for each choice of $(\mathbf{x}_t, y_p)$, we consider 8 model initializations to generate 8 $\mathbf{w}_c$, apply Algorithm 1 on each $\mathbf{w}_c$ and obtain the average; (3) For the calculation of τ, we only apply one set of $\mathbf{w}_c$ and consider the number of classes $c = 10$ for CIFAR-10 and $c = 200$ for TinyImageNet and apply Theorem 1.

We further report our resource and computational time in Appendix D.1.

4.2 Poisoning difficulty prediction with `EPA`

Recall in Section 3.1 Hypothesis A we hypothesize a negative correlation between classification difficulty and poisoning difficulty. In this section, we aim to show that (1) `EPA`, defined in Definition 1, is a good indicator of poisoning difficulty; (2) `EPA` is not capable of differentiating between different poison classes y_p for a test sample x_t, and is ineffective on further ranking the poisoning difficulty within groups of targets with similar `EPA`.

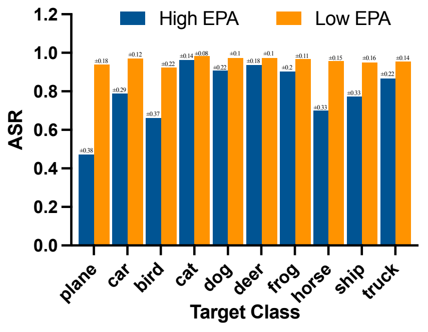
Training from scratch: For the from-scratch setting on ResNet-18/CIFAR-10, we perform clean training on $\mathcal{D}_c$ with the prespecified M and N to identify the 50 target samples with the highest and lowest `EPA` in each target class y_t. For each target sample, we perform the GM attack on all possible (9) poison classes y_p for 8 randomly initialized model weights. We thus run $(50 + 50) \times 10 \times 9 \times 8 = 72000$ attack instances in total. We report the computation time for each attack instance in Appendix D.1. We present our main results in Figure 3 and observe that `EPA` is a reliable indicator

[4]`https://github.com/aks2203/poisoning-benchmark`

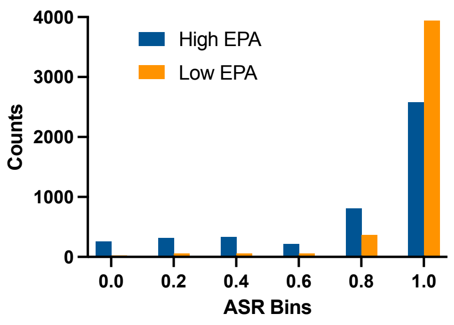
[5]Note that for replacing attacks, $\mathcal{D}_c$ can be changed after poisoning, see Appendix B.2 for discussion.

[6]`https://github.com/JonasGeiping/poisoning-gradient-matching`

[7]We neglect Convex Polytope (CP) [49] as it is extremely expensive. Specifically, it takes 100 seconds and 40 seconds to run one attack instance for BP and FC, respectively, while it takes more than 1 hour to run CP on a NVIDIA 4090 GPU. As our experiments require thousands of attack instances, it is infeasible to run CP.



(a) ASR for different y_t



(b) ASR distribution

Figure 3: Measuring the poisoning difficulty of GM on CIFAR-10 (training from scratch) using EPA. We plot the ASR for low/high EPA test samples in each target class y_t in (a), and the overall ASR distribution as a histogram in (b).

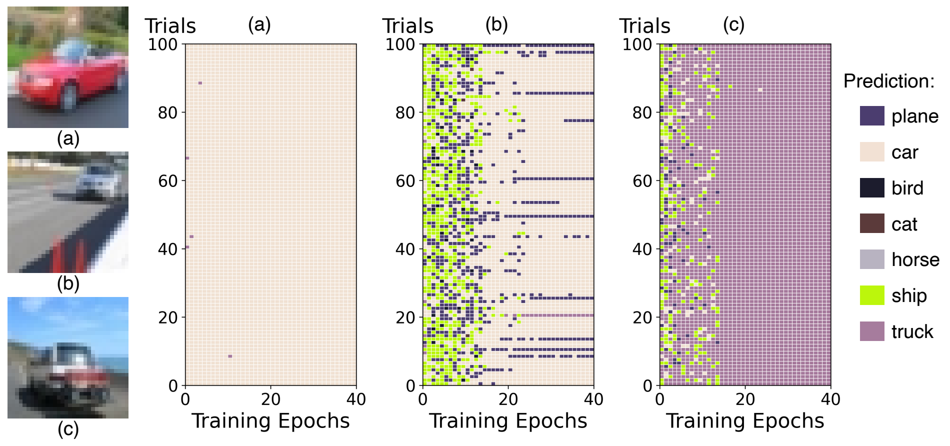


Figure 4: EPA for three test instances in the class “car”. Image (a): high EPA: 0.9988; ASR: 22.22%. Image (b): medium EPA: 0.6775; ASR: 90.28%. Image (c): low EPA: 0.0275; ASR: 98.61%.

of poisoning difficulty, where a higher EPA suggests much lower ASR on average. We note that Figure 3(a) demonstrates a discrepancy between different target classes, specifically, for classes cat and dog, the ASR differences are relatively small between high EPA test samples and low EPA ones. We argue that these classes are generally easier to poison as they are more difficult to classify, with 82.66% (cat) and 85.73% (dog) clean test accuracy, while that of all other classes is greater than 90%. We present a visualization of EPA for instances with high, medium, and low EPA values in Figure 4, demonstrating its effectiveness as an indicator of instance-level poisoning difficulty[8].

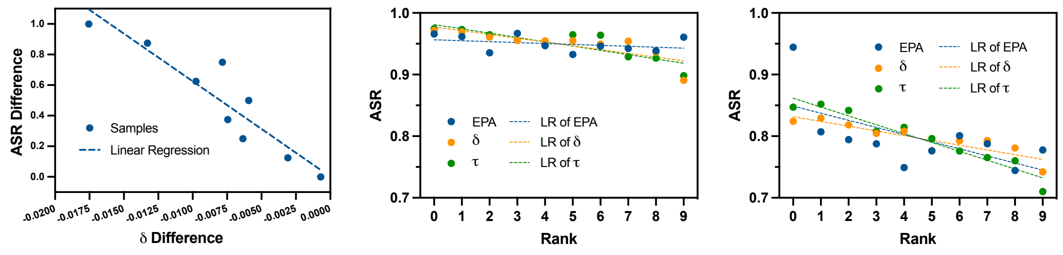
Transfer learning: For the transfer learning setting on ResNet-18 and CIFAR-10, we again identify the 50 target samples with the highest and lowest EPA and apply an additional restriction that all identified target images are classified correctly at the final epoch in all M clean training runs. Table 1 shows our main result on CIFAR-10 for two attacks FC and BP. We observe that the average ASR for test samples with high EPA is much lower than the ones with low EPA for both attacks. Additionally, as we start from a pre-trained model with reasonable performance, it is interesting to visualize the model change after an attack. Thus, we report the average change of confidence for y_p and y_t for each test sample and confirm that EPA is a reliable metric to measure poisoning difficulty. Moreover, to check whether EPA is a reliable tool for predicting attack success, we report the average EPA of test targets that are successfully and unsuccessfully poisoned in Table 2. We observe that EPA is capable of clearly differentiating between successful attacks and failed attacks in most cases, while the prediction region may occasionally overlap.

Table 1: The ASR, change of confidence for y_p and y_t before/after attack for high/low EPA test samples with FC and BP attack on CIFAR-10 with transfer learning.

$\mathbf{x}_t$	ASR		change of confidence (y_p)		change of confidence (y_t)	
	FC	BP	FC	BP	FC	BP
high EPA	0.012	0.498	0.040 ± 0.031	0.479 ± 0.133	-0.048 ± 0.036	-0.503 ± 0.137
low EPA	0.284	0.947	0.156 ± 0.041	0.661 ± 0.063	-0.198 ± 0.053	-0.750 ± 0.058

Table 2: The average EPA and confidence of y_p after clean training for successful/failed attacks using the FC and BP attack on CIFAR-10 with transfer learning.

Attack Success	Average EPA		Confidence of y_p	
	FC	BP	FC	BP
✓	0.411 ± 0.116	0.589 ± 0.255	0.142 ± 0.092	0.043 ± 0.074
✗	0.725 ± 0.264	0.912 ± 0.148	0.012 ± 0.033	0.001 ± 0.002



(a) (b) Low EPA samples. (c) High EPA samples.

Figure 5: (a) Correlation between pairwise δ difference and ASR difference; (b) and (c) Comparison between all of our metrics for low/high EPA samples.

4.3 Poisoning distance and poisoning budget

In previous sections, we showed that EPA is generally a reliable indicator of poisoning difficulty. However, it has 2 major shortcomings: (1) it does not take into account y_p in determining poisoning difficulty, and (2) it becomes ineffective within groups of target samples that have similar EPA. Motivated by the inadequacy of EPA, we apply the poisoning distance and the poison budget measure τ, where our experience suggests that a larger δ or a larger τ indicates a more difficult attack (lower ASR). Specifically, given a target sample $\mathbf{x}_t$, we would like to confirm whether δ and τ are capable of predicting its vulnerability to poisoning towards a poison class y_p.

Specifically, we examine our prior results in the from-scratch setting. First, we look at the choice of y_p with respect to each target class. For each target class, we examine the same 100 test samples (50 highest/lowest EPA targets) and calculate the average ASR for each y_p. We report the two y_p classes with the lowest/highest average ASR and compare the average δ and τ values in Table 3. Note that we focus on *hard-to-classify* samples by performing a pre-screening process to rule out $(\mathbf{x}_t, y_p)$ pairs (1135 out of 9000) that are already classified as y_p at the final epoch of clean training in any of the M trials. While we observe δ and τ are generally capable of identifying easy/hard poison classes, there are some anomalies: for example, the target classes dog and cat have a smaller difference in ASR between the highest/lowest ASR y_p, making them more difficult to differentiate.

To provide further understanding on the role of the poison class y_p, for every individual instance $\mathbf{x}_t$, we enumerate the choice of y_p and create pairs $((\mathbf{x}_t, y_p^1), (\mathbf{x}_t, y_p^2))$ (there are 9 choose 2, which is 36 pairs in total). For each pair, we calculate its corresponding ASR difference and δ difference. After obtaining all 1000×36 pairwise ASR and δ differences, we plot the correlation of the average δ difference with respect to the 9 possible ASR differences[9] in Figure 5(a) and observe that δ is generally reliable even for differentiating pair-wise differences.

[8]We will release an open website enabling verification of EPA for any target sample included in our paper.

[9]Note that for each attack instance, as we perform 8 trials, ASR can take 9 values in $[0, 1]$ with an interval of $1/8$. The ASR difference can only take the same 9 values as we restrict the difference to be positive.

Table 3: Measuring the poisoning difficulty of GM on CIFAR-10 using δ and τ for the poison classes with the highest and lowest ASR over all target classes y_t. Anomaly cases where the prediction does not conform with ASR are marked with underline.

y_t	**Lowest ASR** y_p			**Highest ASR** y_p		
	distance δ	budget τ	ASR	distance δ	budget τ	ASR
plane	0.119 ± 0.042	0.00237 ± 0.00336	0.57 ± 0.42	0.105 ± 0.030	0.00190 ± 0.00321	0.74 ± 0.35
car	0.124 ± 0.040	0.00171 ± 0.00617	0.83 ± 0.28	0.112 ± 0.030	0.00097 ± 0.00187	0.90 ± 0.23
bird	0.114 ± 0.035	0.00237 ± 0.00425	0.60 ± 0.42	0.097 ± 0.033	0.00140 ± 0.00229	0.84 ± 0.28
cat	$\underline{0.093 \pm 0.031}$	0.00049 ± 0.00093	0.92 ± 0.21	0.095 ± 0.026	0.00043 ± 0.00121	0.99 ± 0.05
deer	0.123 ± 0.042	0.00221 ± 0.00494	0.86 ± 0.26	0.092 ± 0.034	0.00117 ± 0.00180	0.99 ± 0.07
dog	0.115 ± 0.038	$\underline{0.00095 \pm 0.00128}$	0.91 ± 0.19	0.102 ± 0.033	0.00246 ± 0.00526	0.98 ± 0.07
frog	0.137 ± 0.042	0.00197 ± 0.00219	0.83 ± 0.26	0.112 ± 0.035	0.00158 ± 0.00240	0.98 ± 0.07
horse	$\underline{0.106 \pm 0.041}$	$\underline{0.00081 \pm 0.00227}$	0.69 ± 0.36	0.137 ± 0.048	0.00123 ± 0.00214	0.87 ± 0.24
ship	0.119 ± 0.037	0.00336 ± 0.00497	0.67 ± 0.38	0.085 ± 0.031	0.00232 ± 0.00564	0.93 ± 0.19
truck	0.125 ± 0.033	0.00130 ± 0.00138	0.80 ± 0.28	0.106 ± 0.031	0.00058 ± 0.00097	0.95 ± 0.11

Moreover, we previously mentioned that EPA is not very reliable in further identifying hard/easy to poison samples within the groups of samples with similar EPA. In Figure 5(b)(c), we rank samples in the high EPA and low EPA region according to their EPA/δ/τ into 10 tiers and plot the average ASR for each tier. We observe that δ and τ cover a much wider range of ASR and are able to further predict the difficulty of poisoning in a more fine-grained way.

4.4 Ablation studies

Necessity of training dynamics: A simple baseline method to predict poisoning difficulty is the confidence of the target label y_t. We run GM on CIFAR-10 and set $y_t = \text{plane}, y_p = \text{bird}$ and report the ASR, EPA, and confidence of y_t for high EPA test samples and lower EPA samples[10] in Table 4. We observe that the confidence of y_t is unable to differentiate samples that are difficult/easy to poison and it is necessary to consider the training dynamics with EPA.

Table 4: Ablation study on predicting poisoning difficulty with the confidence of y_t.

$\mathbf{x}_t$	confidence of y_t	average EPA	average ASR
high EPA	0.9985 ± 0.00309	0.9955 ± 0.00165	0.468 ± 0.3718
lower EPA	0.9999 ± 0.00003	0.9249 ± 0.03210	0.905 ± 0.1899

Poison budget: In our CIFAR-10 experiments using GM, the class "dog" is generally easier to attack, indicating the attack budget may be too high. While in such cases our EPA does not provide clear guidance, we conduct additional experiments by lowering the attack budget ε in Table 5. We observe EPA is more reliable when the attacker's budget is limited, which is usually true in practice.

Table 5: Ablation study on predicting poisoning difficulty with EPA for various attack budget ε.

$\mathbf{x}_t$	$\varepsilon = 1\%$	$\varepsilon = 0.75\%$	$\varepsilon = 0.5\%$	$\varepsilon = 0.25\%$	$\varepsilon = 0.1\%$
high EPA	0.963 ± 0.084	0.963 ± 0.119	0.738 ± 0.375	0.588 ± 0.382	0.263 ± 0.216
low EPA	0.988 ± 0.040	1.000 ± 0.000	0.963 ± 0.060	0.950 ± 0.121	0.662 ± 0.391

TinyImageNet: Finally, we present our results on using EPA to predict poisoning difficulty on TinyImageNet in Table 6. We choose 4 highest/lowest samples from $y_t = 0$ and use GM and BP to poison towards $y_p = 1, 2$, with an attack budget $\varepsilon = 0.05\%$. Our results again confirm that EPA can predict poisoning difficulty.

5 Conclusion

In this paper, we investigated the varying vulnerability of test samples to targeted data poisoning attacks. Our work establishes that poisoning difficulty is not uniform across samples but rather

[10] We restrict the confidence of $y_t > 0.98$ for samples with lower EPA.

Table 6: Poisoning difficulty prediction using EPA on TinyImageNet.

x_t	$y_p = 1$		$y_p = 2$	
	GM	BP	GM	BP
high EPA	0.188 ± 0.375	0.031 ± 0.062	0.531 ± 0.373	0.344 ± 0.472
low EPA	0.812 ± 0.298	0.781 ± 0.438	1.000 ± 0.000	0.781 ± 0.438

depends on specific, measurable characteristics. We identified three key factors, classification difficulty during clean training, parameter space poisoning distance, and poison budget, that significantly influence vulnerability. Based on these factors, we developed three complementary metrics: ergodic prediction accuracy (EPA), poisoning distance and budget lower bound. These metrics provide a comprehensive framework for predicting poisoning difficulty without requiring the execution of actual attacks. Our experimental results validate that EPA effectively separates easy-to-poison from hard-to-poison samples, while δ and τ offer fine-grained predictions for specific poison classes. We extend more discussions on limitations and future works in Appendix A.

References

[1] C. Agarwal, D. D'souza, and S. Hooker. "Estimating example difficulty using variance of gradients". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10368–10378.

[2] H. Aghakhani, D. Meng, Y.-X. Wang, C. Kruegel, and G. Vigna. "Bullseye polytope: A scalable clean-label poisoning attack with improved transferability". In: *IEEE European Symposium on Security and Privacy (EuroS&P)*. 2021, pp. 159–178.

[3] B. Biggio, B. Nelson, and P. Laskov. "Poisoning attacks against support vector machines". In: *Proceedings of the 29th International Conference on Machine Learning (ICML)*. 2012, pp. 1467–1474.

[4] N. Carlini, M. Jagielski, C. A. Choquette-Choo, D. Paleka, W. Pearce, H. Anderson, A. Terzis, K. Thomas, and F. Tramèr. "Poisoning web-scale training datasets is practical". In: *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2024, pp. 407–425.

[5] X. Chen, C. Liu, B. Li, K. Lu, and D. Song. "Targeted backdoor attacks on deep learning systems using data poisoning". arXiv:1712.05526. 2017.

[6] R. A. Davis, K.-S. Lii, and D. N. Politis. "Remarks on some nonparametric estimates of a density function". In: *Selected Works of Murray Rosenblatt*. Springer, 2011, pp. 95–100.

[7] L. Fowl, P.-y. Chiang, M. Goldblum, J. Geiping, A. Bansal, W. Czaja, and T. Goldstein. "Preventing unauthorized use of proprietary data: Poisoning for secure dataset release". arXiv preprint arXiv:2103.02683. 2021.

[8] L. Fowl, M. Goldblum, P.-y. Chiang, J. Geiping, W. Czaja, and T. Goldstein. "Adversarial Examples Make Strong Poisons". In: *Advances in Neural Information Processing Systems*. 2021, pp. 30339–30351.

[9] S. Fu, F. He, Y. Liu, L. Shen, and D. Tao. "Robust unlearnable examples: Protecting data privacy against adversarial learning". In: *International Conference on Learning Representations*. 2021.

[10] R. Gal, Y. Alaluf, Y. Atzmon, O. Patashnik, A. H. Bermano, G. Chechik, and D. Cohen-Or. "An image is worth one word: Personalizing text-to-image generation using textual inversion". In: *The Eleventh International Conference on Learning Representations*. 2023.

[11] L. Gao et al. "The Pile: An 800GB Dataset of Diverse Text for Language Modeling". arXiv preprint arXiv:2101.00027. 2020.

[12] L. A. Gatys, A. S. Ecker, and M. Bethge. "A neural algorithm of artistic style". arXiv preprint arXiv:1508.06576. 2015.

[13] J. Geiping, L. H. Fowl, W. R. Huang, W. Czaja, G. Taylor, M. Moeller, and T. Goldstein. "Witches' Brew: ial Scale Data Poisoning via Gradient Matching". In: *International Conference on Learning Representations*. 2021.

[14] M. Goldblum, D. Tsipras, C. Xie, X. Chen, A. Schwarzschild, D. Song, A. Madry, B. Li, and T. Goldstein. "Dataset Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 2 (2023), pp. 1563–1580.

[15] T. Gu, B. Dolan-Gavitt, and S. Garg. "Badnets: Identifying vulnerabilities in the machine learning model supply chain". arXiv:1708.06733. 2017.

[16] J. Guo and C. Liu. "Practical Poisoning Attacks on Neural Networks". In: *European Conference on Computer Vision*. 2020, pp. 142–158.

[17] D. M. Hawkins. "The detection of errors in multivariate data using principal components". *Journal of the American Statistical Association*, vol. 69, no. 346 (1974), pp. 340–344.

[18] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[19] H. Huang, X. Ma, S. M. Erfani, J. Bailey, and Y. Wang. "Unlearnable Examples: Making Personal Data Unexploitable". In: *International Conference on Learning Representations*. 2021.

[20] X. Huang and S. Belongie. “Arbitrary style transfer in real-time with adaptive instance normalization”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 1501–1510.

[21] P. W. Koh and P. Liang. “Understanding black-box predictions via influence functions”. In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 2017, pp. 1885–1894.

[22] P. W. Koh, J. Steinhardt, and P. Liang. “Stronger Data Poisoning Attacks Break Data Sanitization Defenses”. *Machine Learning*, vol. 111 (2022), pp. 1–47.

[23] A. Krizhevsky. “Learning multiple layers of features from tiny images”. tech. report. 2009.

[24] R. S. S. Kumar, M. Nyström, J. Lambert, A. Marshall, M. Goertzel, A. Comissoneru, M. Swann, and S. Xia. “Adversarial machine learning-industry perspectives”. In: *IEEE Security and Privacy Workshops (SPW)*. 2020, pp. 69–75.

[25] Y. Le and X. Yang. “Tiny imagenet visual recognition challenge”. *CS 231N*, vol. 7, no. 7 (2015), p. 3.

[26] W. Liu and S. Chawla. “Mining adversarial patterns via regularized loss minimization”. *Machine learning*, vol. 81, no. 1 (2010), pp. 69–83.

[27] Y. Lu, G. Kamath, and Y. Yu. “Exploring the Limits of Model-Targeted Indiscriminate Data Poisoning Attacks”. In: *Proceedings of the 40th International Conference on Machine Learning*. 2023.

[28] Y. Lu, G. Kamath, and Y. Yu. “Indiscriminate Data Poisoning Attacks on Neural Networks”. *Transactions on Machine Learning Research* (2022).

[29] Y. Lu, M. Y. Yang, Z. Liu, G. Kamath, and Y. Yu. “Disguised Copyright Infringement of Latent Diffusion Model”. In: *International Conference on Machine Learning*. PMLR. 2024.

[30] L. Lyu, H. Yu, and Q. Yang. “Threats to federated learning: A survey”. arXiv preprint arXiv:2003.02133. 2020.

[31] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli. “Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization”. In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security (AISec)*. 2017, pp. 27–38.

[32] S. Rabanser, A. Thudi, K. Hamidieh, A. Dziedzic, and N. Papernot. “Selective classification via neural network training dynamics”. *arXiv preprint arXiv:2205.13532* (2022).

[33] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. “High-resolution image synthesis with latent diffusion models”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695.

[34] H. Royden and P. Fitzpatrick. “Real Analysis”. 4th. Pearson, 2010.

[35] A. Saha, A. Subramanya, and H. Pirsiavash. “Hidden trigger backdoor attacks”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2020.

[36] P. Sandoval-Segura, V. Singla, J. Geiping, M. Goldblum, T. Goldstein, and D. W. Jacobs. “Autoregressive Perturbations for Data Poisoning”. In: *Advances in Neural Information Processing Systems*. 2022.

[37] A. Schwarzschild, M. Goldblum, A. Gupta, J. P. Dickerson, and T. Goldstein. “Just How Toxic is Data Poisoning? A Unified Benchmark for Backdoor and Data Poisoning Attacks”. In: *Proceedings of the 38th International Conference on Machine Learning (ICML)*. 2021.

[38] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein. “Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018, pp. 6103–6113.

[39] S. Shan, W. Ding, J. Passananti, S. Wu, H. Zheng, and B. Y. Zhao. “Nightshade: Prompt-specific poisoning attacks on text-to-image generative models”. In: *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2024, pp. 807–825.

[40] V. Shejwalkar, A. Houmansadr, P. Kairouz, and D. Ramage. “Back to the Drawing Board: A Critical Evaluation of Poisoning Attacks on Production Federated Learning”. In: *IEEE Symposium on Security and Privacy (SP)*. 2022, pp. 1354–1371.

[41] K. Simonyan and A. Zisserman. “Very deep convolutional networks for large-scale image recognition”. *arXiv preprint arXiv:1409.1556* (2014).

[42] X. Sun, Z. Zhang, X. Ren, R. Luo, and L. Li. “Exploring the vulnerability of deep neural networks: A study of parameter corruption”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2020.

[43] F. Suya, S. Mahloujifar, A. Suri, D. Evans, and Y. Tian. “Model-targeted poisoning attacks with provable convergence”. In: *Proceedings of the 38th International Conference on Machine Learning (ICML)*. 2021, pp. 10000–10010.

[44] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. “Intriguing properties of neural networks”. In: International Conference on Learning Representation. 2014.

[45] B. Tran, J. Li, and A. Madry. “Spectral Signatures in Backdoor Attacks”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[46] J. Wakefield. “Microsoft chatbot is taught to swear on Twitter”. *BBC News* (2016).

[47] A. Wan, E. Wallace, S. Shen, and D. Klein. “Poisoning language models during instruction tuning”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 35413–35425.

[48] D. Yu, H. Zhang, W. Chen, J. Yin, and T.-Y. Liu. “Availability Attacks Create Shortcuts”. In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2022, pp. 2367–2376.

[49] C. Zhu, W. R. Huang, H. Li, G. Taylor, C. Studer, and T. Goldstein. “Transferable clean-label poisoning attacks on deep neural nets”. In: *International Conference on Machine Learning*. 2019, pp. 7614–7623.

A Limitations and Future works

Practical Utility for Defenders. The framework proposed in this paper has potential for practical application in defending against data poisoning attacks. Defenders could integrate these metrics into a continuous monitoring pipeline to achieve proactive vulnerability assessment. For instance, `EPA` could be periodically computed for a set of critical or high-stakes test samples to identify any that exhibit unstable predictions during routine model updates. For samples flagged with low `EPA`, or for particularly sensitive targets (e.g., a specific face in a facial recognition system), a defender could then compute δ and τ against a set of likely or dangerous poison classes. A high-risk combination could trigger an alert, mandate human-in-the-loop verification for that sample's predictions, or initiate a forensic analysis of the training data pipeline.

Limitations. Our work still has limitations: (1) As shown in our experiments, all of our metrics may occasionally yield inaccurate predictions of poisoning difficulty, indicating room for improvement; (2) Our method still relies on test sample labels to accurately predict poisoning difficulty, which could be infeasible in practice. Thus, a label-agnostic approach is largely desired; (3)Our quantitative metrics are currently limited to classification models, with our diffusion model analysis in Appendix E remaining qualitative.

Future Works. Several potential future directions emerge from our work: (1) Data-centric defenses that optimize test samples to defend against targeted data poisoning attacks. For example, defenders might apply carefully crafted adversarial noise to test data, similar to techniques used in adversarial examples; (2) While we make initial attempts to extend our discussion to diffusion models in Appendix E, future work could explore how these vulnerability insights inform the development of more robust LLMs against targeted data poisoning attacks. Moreover, developing universal quantitative metrics for assessing poisoning difficulty across different model types is a crucial future step.

B Related works

B.1 Data poisoning attacks

Data poisoning, an emerging training-time concern in modern ML pipelines, refers to the threat of (actively or passively) crafting "poisoned" training data $\mathcal{D}_p$ so that systems trained on it (along with possibly clean in-house data $\mathcal{D}_c$) are skewed toward certain behaviors. Significant research has been proposed to study the impact of such attacks on classification models. For example, indiscriminate data poisoning [e.g., 3, 21, 22, 27, 28, 29, 31] is a general-purpose attack that aims to decrease the overall test accuracy. Similar formulations have been proposed for protecting user data [e.g., 7, 8, 9, 19, 26, 36, 48]. While data poisoning attacks can also involve testing-time manipulation—such as backdoor attacks [e.g., 5, 15, 35, 45] that aim to trigger malicious model behavior with particular patterns on test samples, we focus exclusively on training-time attacks in this paper.

B.2 Adding attack vs Replacing attack

Realistically, an attacker would have no control on the clean set $\mathcal{D}_c$, and data poisoning attacks [e.g., 3, 22, 27, 28, 31] usually consider *adding-only* attack where $\mathcal{D}_c$ is intact and the size of $\mathcal{D}_{tr}$ increases. However, targeted attacks [e.g., 2, 16, 38, 49] consider *replacing* attacks where part of the clean set $\mathcal{D}_c$ is substituted[11] with $\mathcal{D}_p$ while the size of $\mathcal{D}_{tr}$ is unchanged. In this paper, we follow previous works and consider *replacing attacks*. While the practicality of such attacks are beyond our scope, we note that the key technical differences comparing with adding-only attacks: replacing attacks are notably easier as it reduces $|\mathcal{D}_c|$ and considers a slightly higher ε as $|\mathcal{D}_{tr}|$ is a constant (see Appendix C.10 in [27] for a detailed discussion).

[11] Such substitution is performed by simply adding noise to the original clean samples. Such a setting could resonate in targeted settings as it would keep the balance between classes.

B.3 Measuring classification difficulty

The problem of measuring classification difficulty has been explored in prior literature. For instance, Agarwal et al. [1] proposed variance of gradient (VOG) as a method to rank examples by classification difficulty. VOG could potentially serve as an alternative to EPA for measuring classification difficulty in Hypothesis A and may function as an indicator for predicting poisoning vulnerability—a direction we intend to investigate in future work. Additionally, out-of-distribution (OoD) detection techniques such as PCA [17] and KDE [6] could potentially identify *hard-to-classify* (and possibly *easy-to-poison*) anomalous samples.

Furthermore, our approach relates to selective classification [32], where models reject inputs likely to be misclassified while maintaining high performance on accepted inputs. Specifically, Rabanser et al. [32] leverages prediction agreement between intermediate training stages and the final epoch—a strategy similar to our EPA metric that also analyzes clean training dynamics. However, unlike selective classification, we assign a EPA score to every test instance rather than implementing a rejection mechanism.

B.4 Connection and differences with model-targeted attacks

We note that our core idea of poisoning distance is closely related to model-targeted indiscriminate attacks which we denote as MTA [22, 27, 43], where these attacks consider a set of target parameters $\mathbf{w}_p$ as the target and apply gradient-based poisoning attacks to achieve $\mathbf{w}_p$. While the concept of target parameters is also used in our paper, we emphasize key differences: (1) *Task*: MTA considers $\mathbf{w}_p$ to be a model with low test accuracy, which can be generated with a gradient-based parameter corruption attack [42]. We consider a set of $\mathbf{w}_p$ that only misclassifies one single test sample. (2) *Using* $\mathbf{w}_p$: MTA uses $\mathbf{w}_p$ as the endgoal to generate poisoning attacks, we use $\mathbf{w}_p$ as proxies to quantify poisoning difficulty. (3) *Attack vs Defense*: MTA are designed for more effective attacks, while our algorithm estimate $\mathbf{w}_p$ and δ to help practitioners understand targeted attack difficulties and design better defenses.

Algorithm 1: Poisoning Distance Estimation

```
Input: clean parameters $\mathbf{w}_c$, target $\mathbf{x}_t$, poison label $y_p$, precision parameter $\alpha = 10^{-4}$
1  calculate the gradient $\mathbf{g} = \nabla_{\mathbf{w}_c} \ell(f(\mathbf{x}_t; \mathbf{w}_c), y_p)$
2  instantiate the upper bound $u = \infty$, lower bound $l = 0$, and medium $m = 0.5$ for binary search
3  while $u - l > \alpha$ do
4      if $u = \infty$ then
5          set $m = 2m$
6      else
7          set $m = \frac{u+l}{2}$
8      if $f(\mathbf{x}_t; \mathbf{w} - m \cdot \mathbf{g}) = y_p$ then
9          set $u = m$
10     else
11         set $l = m$
12 return the estimated poisoning distance $\delta = u$
```

C Algorithm on Estimating Poisoning Distance

Recall that in Section 3.2 we propose to use a binary search based algorithm to estimate the poisoning distance δ. We present the algorithm in Algorithm 1.

D Additional Experiments on Classification Models

D.1 Computing resource & time

Targeted attacks: Due to the extensive number of attacks conducted, we distributed our experiments across three distinct clusters equipped with NVIDIA 4090 (cluster 1), A100 (cluster 2), and

RTX6000 GPUs (cluster 3). The computational requirements varied significantly by task: training models from scratch (GM experiments) required up to 1 hour 40 minutes on all clusters for CIFAR-10/ResNet-18 configurations, while TinyImageNet/VGG16 experiments (GM and BP) demanded up to 3 hours 10 minutes on cluster 2. Transfer learning experiments were considerably more efficient, requiring only 66-72 seconds for BP and 60-63 seconds for FC on clusters 2 and 3, respectively.

Measuring poisoning difficulty: We conducted all experiments on the NVIDIA 4090 cluster. For `EPA` calculations, the computational cost scales linearly with the number of trials M multiplied by the clean training time. For individual test samples, computing all nine possible δ values for a single set of $\mathbf{w}_c$ requires just 1.3 seconds, while calculating all nine possible τ values takes approximately 30 seconds on our ResNet-18/CIFAR-10 experimental setup.

D.2 Other baselines for poisoning difficulty prediction on y_p

In Section 4.3, we introduce δ as the indicator for poisoning difficulty for poison classes y_p with the highest and lowest ASR. Here we perform the same task for two baseline methods: (1) the average confidence of a given y_p at the end of clean training; (2) the poison prediction area (`PPA`), where we apply a similar definition with `EPA`, but only considers the prediction of y_p. Our results in Table 7 and Table 8 show that these baseline methods fail to predict poisoning difficulty for most cases.

Table 7: Measuring the poisoning difficulty of GM on CIFAR-10 using the average confidence of y_p at the end of clean training with the highest and lowest ASR over all target classes y_t. Cases where the prediction conforms with ASR are marked in blue, and anomalies are marked in red.

y_t	**Lowest ASR** y_p		**Highest ASR** y_p	
	Avg Confidence of y_p	ASR	Avg Confidence of y_p	ASR
plane	0.002 ± 0.005	0.57 ± 0.42	0.001 ± 0.005	0.74 ± 0.35
car	0.001 ± 0.001	0.83 ± 0.28	0.000 ± 0.001	0.90 ± 0.23
bird	0.001 ± 0.003	0.60 ± 0.42	0.003 ± 0.008	0.84 ± 0.28
cat	0.002 ± 0.006	0.92 ± 0.21	0.001 ± 0.004	0.99 ± 0.05
deer	0.002 ± 0.006	0.86 ± 0.26	0.003 ± 0.006	0.99 ± 0.07
dog	0.002 ± 0.005	0.91 ± 0.19	0.002 ± 0.006	0.98 ± 0.07
frog	0.002 ± 0.005	0.83 ± 0.26	0.001 ± 0.004	0.98 ± 0.07
horse	0.003 ± 0.011	0.69 ± 0.36	0.002 ± 0.005	0.87 ± 0.24
ship	0.001 ± 0.002	0.67 ± 0.38	0.002 ± 0.006	0.93 ± 0.19
truck	0.000 ± 0.002	0.80 ± 0.28	0.001 ± 0.004	0.95 ± 0.11

Table 8: Measuring the poisoning difficulty of GM on CIFAR-10 using poison prediction area (`PPA`) with the highest and lowest ASR over all target classes y_t. Cases where the prediction conforms with ASR are marked with blue, and anomalies are marked with red.

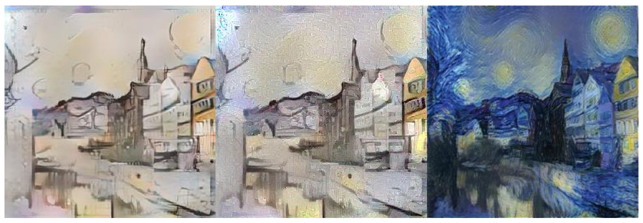
y_t	**Lowest ASR** y_p		**Highest ASR** y_p	
	`PPA`	ASR	`PPA`	ASR
plane	0.004 ± 0.016	0.57 ± 0.42	0.003 ± 0.012	0.74 ± 0.35
car	0.004 ± 0.014	0.83 ± 0.28	0.001 ± 0.005	0.90 ± 0.23
bird	0.005 ± 0.009	0.60 ± 0.42	0.005 ± 0.010	0.84 ± 0.28
cat	0.016 ± 0.026	0.92 ± 0.21	0.006 ± 0.013	0.99 ± 0.05
deer	0.017 ± 0.031	0.86 ± 0.26	0.004 ± 0.007	0.99 ± 0.07
dog	0.008 ± 0.013	0.91 ± 0.19	0.003 ± 0.011	0.98 ± 0.07
frog	0.003 ± 0.012	0.83 ± 0.26	0.006 ± 0.015	0.98 ± 0.07
horse	0.012 ± 0.042	0.69 ± 0.36	0.003 ± 0.013	0.87 ± 0.24
ship	0.004 ± 0.015	0.67 ± 0.38	0.004 ± 0.012	0.93 ± 0.19
truck	0.002 ± 0.008	0.80 ± 0.28	0.002 ± 0.007	0.95 ± 0.11

E Targeted Attacks on Latent Diffusion Models

For generative models, targeted attacks aim to alter the generation of specific concepts or prompts while maintaining expected behavior for other inputs [29, 39, 47]. In this paper, we aim to extend the examination on instance-level difficulty in targeted data poisoning attacks to latent diffusion models [33]. Specifically, we consider the disguised copyright infringement (DCI) attack in [29]. DCI considers a threat where an attacker aims to mimic the style of a copyrighted image x_c without directly training on x_c, but creates a disguise image (or poison sample) x_d that visually assembles another base image x_b while containing the latent information of x_c. Although the task and poisoning mechanism are very different from those of classification models that we consider in the main paper, we find two factors that would affect the poisoning difficulty.

Structure of x_b: We follow the implementation of [29][12] and consider the task of disguising style. We pick the drawing: The Neckarfront in Tubingen, Germany (photo by Andreas Praefcke) in the style of *The Starry Night*, generated with Neural Style Transfer [12] as x_c. The base image x_b is x_c with another style (watercolor), generated with AdaIN-based [20] style transfer[13]. The disguise x_d is generated using Algorithm 1 in [29] and we train the disguise x_d using textual inversion [10] for generation. We fix x_t and study the role of the structure of x_b by applying gaussian blur with different kernel size (a larger kernel size results in a more blurry image). We report our results in Figure 6,Figure 7, Figure 8, Figure 9 and Figure 10. We observe that by increasing the kernel size, the cirrus effect of the generated images dramatically decreases. When the kernel size is bigger than 10, the textual inversion model cannot learn any useful information. We conclude that preserving the structure of x_b is essential for a successful data poisoning attack, highlighting the role on the appearance of the poison image in poisoning difficulty.

Structure of x_c: We also observe that the structure of x_c (target or copyright image) also affects the poisoning difficulty. In Figure 11, we choose another x_t[14] with much simpler layout in the same style of *The Starry Night*. We observe that style mimicry is unsuccessful for this poison instance, validating that poisoning success is also correlated with the structure of x_c.


(a) Base x_b (b) Disguise x_d (c) x_c

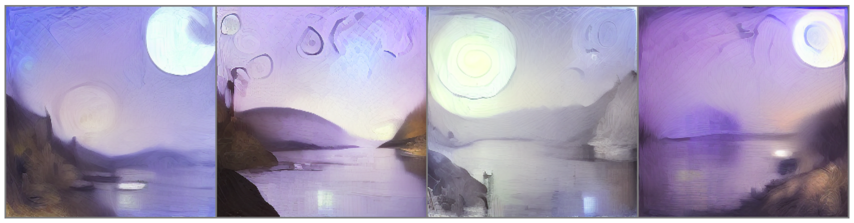
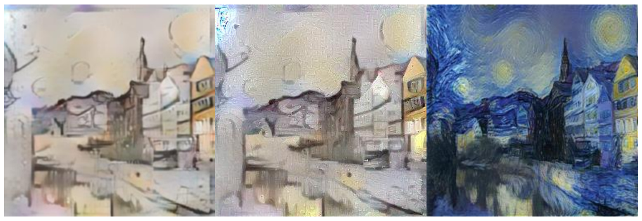

(d) Images generated by textual inversion after training on the x_d

Figure 6: Disguised copyrighted style on textual inversion with the original x_b.

[12]`https://github.com/watml/disguised_copyright_infringement`

[13]`https://github.com/tyui592/AdaIN_Pytorch`

[14]https://stock.adobe.com/images/glowing-moon-on-a-blue-sky-abstract-background-seamless-vector-pattern-in-the-style-of-impressionist-paintings/475101004


(a) Base x_b (b) Disguise x_d (c) x_c

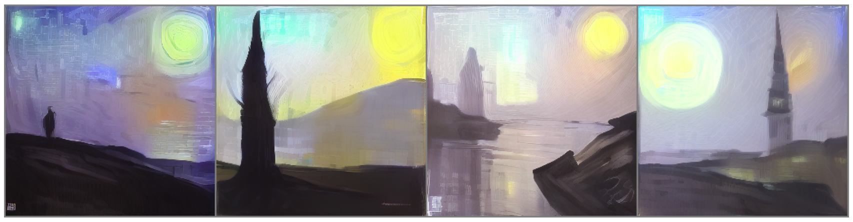
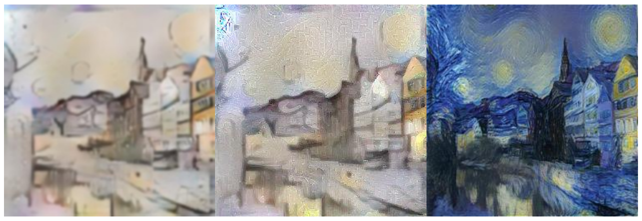

(d) Images generated by textual inversion after training on the x_d

Figure 7: Disguised copyrighted style on textual inversion with the blurry x_b (kernel size = 3).


(a) Base x_b (b) Disguise x_d (c) x_c

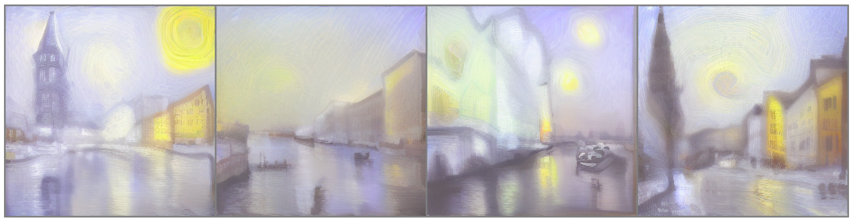
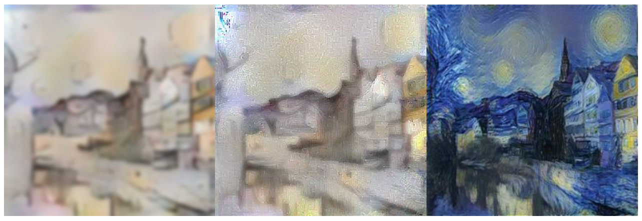

(d) Images generated by textual inversion after training on the x_d

Figure 8: Disguised copyrighted style on textual inversion with the blurry x_b (kernel size = 7).


(a) Base x_b (b) Disguise x_d (c) x_c

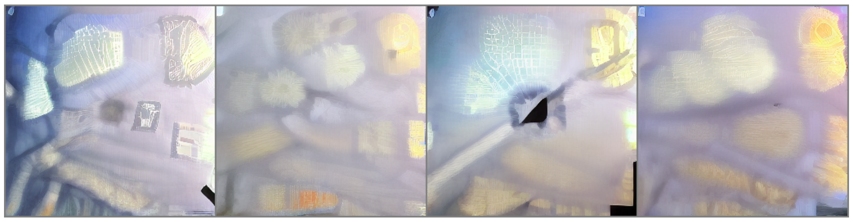
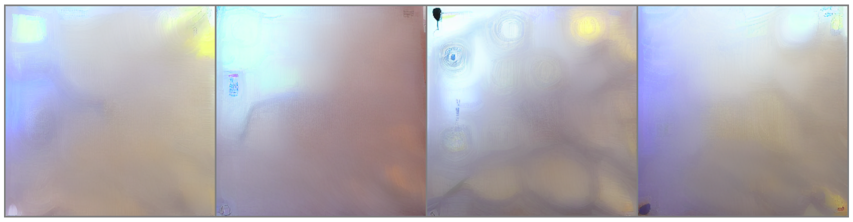

(d) Images generated by textual inversion after training on the x_d

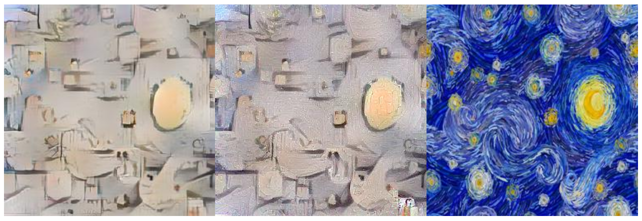
Figure 9: Disguised copyrighted style on textual inversion with the blurry x_b (kernel size = 13).

(a) Base x_b (b) Disguise x_d (c) x_c

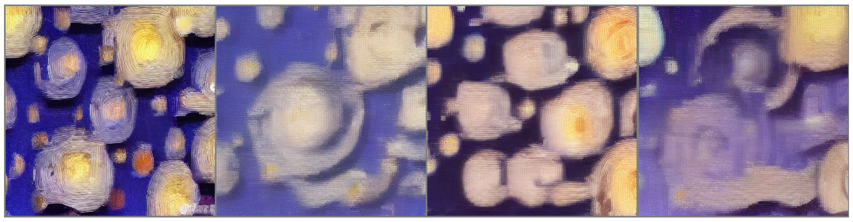


(d) Images generated by textual inversion after training on the x_d

Figure 10: Disguised copyrighted style on textual inversion with the blurry x_b (kernel size = 49).



(a) Base x_b (b) Disguise x_d (c) x_c



(d) Images generated by textual inversion after training on the x_d

Figure 11: Disguised copyrighted style on textual inversion with a different choice of x_t.