Are Well-Trained Surrogates Optimal? Rethinking the Surrogate Role with Instability for Unlearnable Examples

Binze Wang
Macau University of Science and Technology
Macau, China
3240007529@student.must.edu.mo

Jinyu Tian *
Macau University of Science and Technology
Macau, China
jytian@must.edu.mo

Xingrun Wang
FoShan University
FoShan, China
wangxingrun@fosu.edu.cn

Jianqing Li
Macau University of Science and Technology
Macau, China
jqli@must.edu.mo

Abstract

Unlearnable Examples (UEs) aim to protect data from unauthorized model training by injecting imperceptible perturbations that degrade model generalization. Existing methods typically rely on well-trained surrogate models to generate such perturbations, implicitly assuming that stronger surrogates yield better degradation. In this work, we challenge this widely adopted assumption and show that it is fundamentally suboptimal. We provide theoretical insights by establishing an upper bound on the target model's generalization error, showing that perturbation effectiveness is closely related to the surrogate's sensitivity to input perturbations. Specifically, a less robust surrogate model leads to stronger poisoning performance. Motivated by this analysis, we propose a simple yet effective stability metric to select optimal surrogate models for single-level methods. We further extend this principle to bi-level optimization frameworks. Beyond revealing the bi-level method's implicit reliance on non-robust surrogates, we substantially boost its poisoning performance by the stability-based surrogate selection as well. Experiments on five datasets and four representative UE methods show that our stability-based surrogate model selection strategy reduces the target model's test accuracy by an average of 18% compared with well-trained surrogates. Notably, this is the first work to reduce CIFAR-10 test accuracy to 10% under a perturbation budget of 1/255, whereas prior UE methods typically rely on the budget of 8/255.

1 Introduction

The vast data resources on the Internet have become the core fuel driving breakthroughs in artificial intelligence. For instance, training models on vast, web-crawled image datasets has propelled the advancement of AI applications. However, behind these technological leaps, ethical crises stemming from unregulated data collection and usage are spreading globally. To address privacy and intellectual property concerns, Unlearnable Examples (UEs) [19] represent a novel image protection technique. By adding imperceptible perturbations to images, UEs tend to induce models to learn spurious "shortcuts" between perturbations and labels, thereby hindering the extraction of true semantic features and degrading generalization on clean datasets [45].

*corresponding author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

The generation of UEs typically relies on surrogate models, where perturbations are created by either maximizing or minimizing the training loss [10, 37, 46]. However, these studies often overlook how the choice of surrogate model affects the poisoning performance of perturbations. It is common practice to generate perturbations based on a well-trained model, yet this approach has not been thoroughly examined. Intuitively, such perturbations should perform best in attacking the well-trained model itself. Nevertheless, after the target model is trained on UEs, there exists a significant difference between the poisoned model and the well-trained surrogate. Therefore, the poisoning performance of the perturbations may not remain consistent. To demonstrate that a well-trained model is not the optimal surrogate for UEs, we produce perturbations on CIFAR-10 using surrogate models at different training stages and compare their poisoning performance. We evaluate three UEs methods: EMax [10], HYPO [37], and TAP [10]. As shown in Figure 1, the poisoning performance of the perturbations does not correlate linearly with the training accuracy of the surrogate model. The X-axis represents the surrogate model's training epochs, and the Y-axis represents the test accuracy of the target model. They are all ResNet18 architectures. The lower test accuracy corresponds to stronger poisoning performance. The results reveal that as training progresses, the surrogate model's accuracy (dotted line) steadily improves. In contrast, the poisoning performance of all three perturbations follows a three-phase trend: an initial increase, a mid-stage plateau, and a final decline. This pattern challenges the conventional reliance on well-trained models as surrogates for UEs tasks [10, 37, 46].

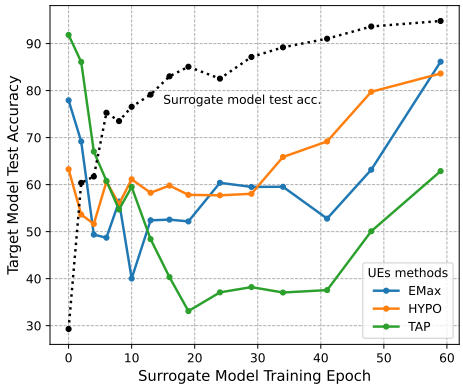


Figure 1: Surrogate training epoch's impact on poisoning effectiveness of perturbation.

We further observe that, at the middle training stages, all three perturbation generation methods achieve their respective optimal poisoning performance. Despite their differing objectives, these methods are fundamentally adversarial, and the resulting perturbations consistently exploit *non-robust features*, imperceptible yet predictive signals learned during standard training [20]. In adversarial attack settings, the effectiveness of perturbations reflects the model's reliance on such *non-robust features*: the stronger the reliance, the greater the performance degradation caused by adversarial examples. In UEs settings, **the variation in poisoning effectiveness across surrogate models at different training stages is likely related to differences in surrogate sensitivity to *non-robust features***. Moreover, this reliance evolves in a non-monotonic manner during training. Such non-monotonic dynamics in model training are consistent with existing theories. The information bottleneck theory [38] characterizes DNN training as consisting of two phases: feature fitting and information compression, reflecting a transition from memorization to abstraction. Additionally, deep neural networks exhibit a *critical learning period*, analogous to that in biological neuroscience [7], where certain capabilities (e.g., vision or language) are more readily acquired; disruptions during this period can have lasting effects that are difficult to recover from later. Similarly, in DNNs, blur-induced distortion in representation learning during early stages can have a disproportionately large impact on final model performance [1]. Building upon this insight, we ask: if the poisoning effectiveness of perturbations is indeed linked to the surrogate model's reliance on *non-robust features*, can we select a non-robust surrogate to generate stronger poisoning perturbations?

In our subsequent theoretical analysis, we show that the poisoning effectiveness of perturbations is closely related to the stability of the surrogate model: the less stable the surrogate, the stronger the poisoning performance. Based on this theory, we propose a stability-based surrogate model selection strategy to enhance the poisoning performance of existing UEs methods. Current surrogate-based UEs methods fall into two categories: single-level optimization and bi-level optimization. In single-level optimization, the surrogate model parameters are fixed, and the model is trained on clean data. We reveal that during standard model training, model stability follows a pattern of stable $\rightarrow$ unstable $\rightarrow$ stable. Our stability metric selects the most unstable model as the surrogate, improving the poisoning effectiveness of the perturbations. In bi-level optimization, both the surrogate model parameters and the perturbations are optimization variables, and they are updated iteratively. Our experiments reveal that bi-level optimization methods implicitly exploit such unstable surrogates, as they tend to drive

the model toward increased instability. However, due to factors such as random initialization and sampling noise during optimization, the instability of the surrogate is not deterministic. Therefore, we propose another surrogate model selection criterion for bi-level methods.

Our contributions can be summarized below:

- We identify and demonstrate that the common practice of using a well-trained model as the surrogate for perturbation generation is suboptimal. We show that the poisoning effectiveness of perturbations varies significantly across different training stages of the surrogate model, and consistently peaks during an intermediate, unstable phase.
- We propose two simple yet effective stability metrics for surrogate model selection. The first is designed for single-level methods, enabling the identification of more effective surrogate models during standard training. The second targets bi-level methods, facilitating the optimal perturbation selection based on surrogate model stability.
- We conduct extensive experiments on four existing poisoning methods across five datasets. For single-level methods, our surrogate selection strategy provides a maximum improvement of 67% in poisoning effectiveness compared to the conventional use of well-trained surrogates. For bi-level methods, we reduce CIFAR-10 test accuracy to 10% under a perturbation budget of 1/255, whereas prior UE methods typically rely on the budget of 8/255.

2 Related Works

2.1 Unlearnable Examples

Unlearnable Examples (UEs) protect data by adding imperceptible perturbations, causing models trained on them to perform poorly on clean test datasets. Several existing works employ a well-trained model as the surrogate to generate UEs, keeping the surrogate model's parameters fixed. They have designed different optimization objectives. For classification tasks, the TAP method [10] minimizes the loss between the model's prediction and an incorrect class label. Its optimization objective is

$$\boldsymbol{\eta} = \arg \min_{\|\boldsymbol{\eta}\|_p \leq \epsilon} \ell(h(\boldsymbol{x} + \boldsymbol{\eta}), y'), \tag{1}$$

where $\boldsymbol{\eta}$ is the perturbation, $\boldsymbol{x}$ is the input, y' is an incorrect label ($y' \neq y$), h is the well-trained surrogate model, ℓ is the training loss, and $\|\boldsymbol{\eta}\|_p \leq \epsilon$ is the p-norm constraint on the perturbation. Similarly, EMax [10] maximizes the training loss with respect to the correct label, while HYPO [37] minimizes the loss to the correct label. Similar methods also include UC [46], EntF [41], OfA [4] and Armor [14]. These methods all default to using a well-trained model as the surrogate, without exploring the surrogate's impact on the poisoning performance.

In all of the aforementioned methods, the perturbation is the sole optimization variable; therefore, they are categorized as *single-level optimization methods*. In contrast, another line of approaches, referred to as *bi-level optimization methods*, treats both the perturbation and the surrogate model as optimization variables.

$$h, \{\boldsymbol{\eta}_i\} = \arg \min_{h \in \mathcal{H}} \mathbb{E}\Big[\min_{\|\boldsymbol{\eta}\|_p \leq \epsilon} \ell(h(\boldsymbol{x}_i + \boldsymbol{\eta}_i), y_i) \Big]. \tag{2}$$

The perturbations $\boldsymbol{\eta}$ and the parameters of the model h are updated iteratively until the training accuracy is more than a threshold. Representative methods include EM [19], REM [12], SEM [28], GUE [27], TUE [33], PUE [39], MIUE [49], and PIL [26]. The surrogate changes dynamically in bi-level optimization, making it more complex. Nevertheless, our core conclusion still holds for it.

2.2 Training Dynamics of DNNs

Training dynamics describe the evolving behavior and internal processes of neural networks during optimization [11, 21, 47]. These include the information-bottleneck principle [34], critical learning periods [1], loss of plasticity [29], grokking [32], double descent [30], and edge of stability [6]. While these works largely track performance metrics like loss and accuracy, our study reveals a distinct, non-monotonic variation in model stability, quantified by the sensitivity to input noise. We observe that during standard training, the model's accuracy improves steadily, but its sensitivity first increases,

peaks during an intermediate phase, and finally recovers in the late convergence phase. According to our survey, this three-phase stability pattern: stable $\rightarrow$ unstable $\rightarrow$ stable has not been explicitly highlighted in prior literature.

3 Problem Statement

Our observation indicates that the poisoning performance of the perturbations depends on the training stages of the surrogate model, with poor poisoning effects in the early and late stages and the best effect in the mid-stage (see Figure 1). This raises a question: How does the surrogate model h at different stages affect the poisoning performance of the perturbation $\boldsymbol{\eta}$ in Equation (1)? **Formally, at a certain moment during model training, this model serves as a surrogate to optimize perturbations and generate poisoned data, and the generalization error of the target model trained on these poisoned data should be maximized. How to determine this moment?** Our theory reveals that the generalization error of the target model can be decomposed into three terms: (i) the **perturbation-induced poisoning gap**, measuring the target model's performance discrepancy between clean and poisoned training data; (ii) the **inherent generalization gap**, measuring the target model's performance discrepancy between test data and clean training data; and (iii) the **optimization error**, characterized by the target model's training loss. Notably, the perturbation-induced poisoning gap is the dominant term in this decomposition. Our analysis further shows that this term is controlled by the stability of the surrogate model: a less stable surrogate yields a larger upper bound on the target model's generalization error. Motivated by this insight, we propose to select the most unstable surrogate model for perturbation generation. Section 3.1 presents the theoretical foundation, while Sections 3.2 and 3.3 introduce our single-level and bi-level surrogate selection strategies, respectively.

3.1 Upper Bound of Generalization Error in Single-level UEs

Before formally presenting the generalization error analysis mentioned above, we first provide a necessary description and definitions. In the unlearnable examples setting, the dataset is divided into a clean training set $\mathcal{D}_C = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, a poisoned training set $\mathcal{D}_P = \{(\hat{\boldsymbol{x}}_i, y_i)\}_{i=1}^N$, and a clean test set $\mathcal{D}_T = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^M$. Each poisoned training sample is obtained by adding a perturbation to the corresponding clean training sample: $\hat{\boldsymbol{x}}_i = \boldsymbol{x}_i + \boldsymbol{\eta}_i$, where $\|\boldsymbol{\eta}_i\|_p \leq \epsilon$. The surrogate model h_S is trained on the clean training set $\mathcal{D}_C$. Let h_S^t denote the surrogate model after t parameter update iterations. After convergence following T updates, we obtain the training trajectory of the surrogate model $\{h_S^t\}_{t=1}^T$. The target model h_T is trained to converge on the poisoned training set $\mathcal{D}_P$. While perturbations $\boldsymbol{\eta}_i$ can be generated via various methods, now we focus on the single-level optimization methods [4, 10, 14, 37, 41, 46], as formalized in Equation (1).

Recalling our observations, the poisoning effectiveness of perturbations varies with the training iteration of the surrogate model. This observation is theoretically justified by the following theorem, which states that the stability of the surrogate model governs the upper bound of the target model's generalization error after poisoning. As the stability of the surrogate model changes across training iterations, the poisoning performance of the generated perturbations varies accordingly.

Theorem 3.1. *Suppose the surrogate model h_S is trained on the clean training set $\mathcal{D}_C$ for T iterations, yielding the trajectory $\{h_S^t\}_{t=1}^T$. At iteration t, the poisoning algorithm generates perturbations $\boldsymbol{\eta}^t = \mathcal{A}(h_S^t, \mathcal{D}_C)$, which induces the poisoned training set $\mathcal{D}_P^t$. Let h_T^t denote the target model trained on $\mathcal{D}_P^t$ until convergence. Then the generalization loss on the test data of h_T^t satisfies*

$$\begin{aligned}\mathcal{L}(h_T^t, \mathcal{D}_T) \leq \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_C}\left[\|\boldsymbol{\eta}^t\|_2 \cdot \|\nabla_{\boldsymbol{x}}\ell(h_S^t, \boldsymbol{x})\|_2 + \frac{1}{2}\|\boldsymbol{\eta}^t\|_2^2 \cdot \rho\big(H_{\boldsymbol{x}}\ell(h_S^t, \boldsymbol{x})\big) + \left|o(\|\boldsymbol{\eta}^t\|_2^2)\right|\right] \\ + \Omega + \mathcal{R}(h_T^t) + \mathcal{L}(h_T^t, \mathcal{D}_P^t).\end{aligned} \tag{3}$$

Here, $\nabla_{\boldsymbol{x}}\ell(h_S^t, \boldsymbol{x})$ is the gradient of the surrogate model loss with respect to the input $\boldsymbol{x}$, $H_{\boldsymbol{x}}\ell(h_S^t, \boldsymbol{x})$ is the Hessian matrix of the surrogate model loss with respect to the input $\boldsymbol{x}$, $\|\cdot\|_2$ denotes the Euclidean norm, $\rho(\cdot)$ is the spectral radius of a matrix, e.g., the maximum of the absolute values of eigenvalue of matrix, and $o(\|\boldsymbol{\eta}\|_2^2)$ is a higher-order infinitesimal term with respect to $\|\boldsymbol{\eta}\|_2^2$. Moreover, $\mathcal{R}(h_T^t)$ is the inherent generalization gap, $\mathcal{L}(h_T^t, \mathcal{D}_P^t)$ is the optimization error of the target model, and Ω is the maximum deviation gap.

The proof is provided in Appendix A. We now briefly analyze Theorem 3.1. The upper bound on the generalization error of the target model consists of three parts. **Part 1** consists of an expectation

term and an Ω term. The expectation term involves the gradient norm of the surrogate model's loss with respect to the input and the spectral radius of its Hessian matrix; larger values indicate greater sensitivity of the model to input perturbations. This expectation term reflects the stability of the surrogate model at training time t: the more unstable the surrogate model, the larger the upper bound on the generalization error of the target model. The Ω term represents, over all perturbations generated by the surrogate model across all times $[1, \ldots, T]$, the maximum deviation in perturbation sensitivity between the surrogate model and the target model. Hence, Ω is a constant that does not depend on any specific time t; instead, it captures the largest possible deviation during the entire training process. **Part 2** is the $\mathcal{R}(h_T^t)$ term, which reflects the target model's performance discrepancy between test data and clean training data. This term can be viewed as a small quantity and independent of the training stage t. More precisely, in UEs tasks, there exist two different distributions of data: poisoned data and clean data. The poisoned target model learns spurious features, thereby reducing its generalization ability on clean data. Both the test data and the clean training data are perturbation-free and belong to the same distribution. Although there is some distributional difference between them, this difference is nearly unmeasurable for a target model trained on poisoned data. Stronger poisoning perturbations lead to a more distinct distributional discrepancy between poisoned and clean data. The target model becomes less affected by internal distributional variations within the clean dataset. In the appendix B.1, we replace the loss with accuracy to compute $\mathcal{R}(h_T^t)$ for different t, all computed values are less than 1%. **Part 3** is the optimization error of the target model. Modern optimization algorithms typically drive the model training loss to a small value. Consequently, the substantial variation in the generalization error across different iterations t should be attributed primarily to the expectation term, which reflects the stability of the surrogate model. In particular, a less stable surrogate model tends to induce a larger generalization error for the target model. A natural question, then, is whether this bound provides quantitative guidance for the surrogate selection. We now discuss how to use the model stability to identify a surrogate model.

3.2 Stability Metric for Surrogate Model Selection

According to Theorem 3.1, the upper bound on the generalization error of the target model is related to the surrogate model stability. The weaker the surrogate model stability, the stronger the poisoning performance of the perturbation. In this subsection, we reveal that the stability of the surrogate model during training undergoes a transition from stable to unstable and then back to stable, leading to a corresponding poisoning effect that first increases and then decreases. To enhance the poisoning performance, we need to identify the least stable model during the training process to serve as the surrogate instead of the well-trained one. Below, we present the criterion for selecting the optimal surrogate model $h_S^{t^*}$.

$$t^* = \arg\max_{t \in [1,T]} \mathcal{I}(h_S^t), \quad \text{where } \mathcal{I}(h) = \mathbb{E}_{x \sim \mathcal{D}_C} \left[\|\boldsymbol{\eta}\|_2 \cdot \|\nabla_{\boldsymbol{x}} \ell(h, \boldsymbol{x})\|_2 + \frac{1}{2} \|\boldsymbol{\eta}\|_2^2 \cdot \rho(H_{\boldsymbol{x}} \ell(h, \boldsymbol{x})) \right]. \tag{4}$$

When the stability metric $\mathcal{I}(h_S^t)$ is maximized, the most effective moment for perturbation optimization is found. To evaluate the evolution of model stability during training, we train several standard classification models (including the ResNet 18/50/101/152, MobileNetV2, GoogleNet, and DenseNet121) on benchmark datasets such as CIFAR-10/100, Pets, Cars, and ImageNet. The stability metric $\mathcal{I}(h_S^t)$ in Equation (4) includes the gradient ℓ_2-norm $\|\nabla_{\boldsymbol{x}} \ell(h_S^t, \boldsymbol{x})\|_2$ and the spectral radius of Hessian matrix (sharpness) $\rho(H_{\boldsymbol{x}} \ell(h_S^t, \boldsymbol{x}))$. Specifically, for each epoch t, we freeze the network parameters and compute these metrics with respect to the input variable $\boldsymbol{x}$. We approximate the sharpness using the PyHessian library [44] with Lanczos iterations $k = 20$, and we use 200 randomly selected training samples to estimate

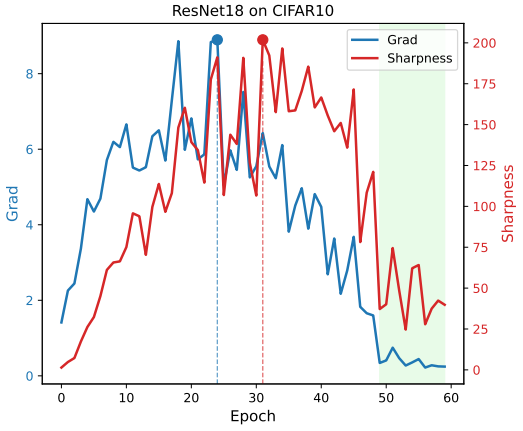


Figure 2: Evolution of gradient norm and sharpness during training.

the stability metric. The runtime for ResNet18 on CIFAR-10 is only 30 seconds per checkpoint. Detailed computational cost and sensitivity analysis are given in the Section 4.6. As shown in Figure 2 and the appendix Figures 4 to 6, the blue curve (left axis) represents the input gradient ℓ_2-norm, while the red curve (right axis) represents sharpness. Both are plotted against the training epochs. The two curves roughly coincide, and both display three distinct phases: initially low and rising (stable), reaching a peak in the middle (unstable), and finally decreasing again to low values (stable). This pattern holds across 5 datasets and 4 architectures (even ResNet152 on ImageNet), confirming that a natural vulnerability period emerges during model training. Our stability metric identifies this period as the optimal timing to optimize perturbation.

Consider integrating multiple models from training as the surrogate, rather than a single model. We present the criterion to select surrogate $\{h_S^{t^*+i}\}_{i=0}^{n-1}$ for ensemble-based poisoning.

$$t^* = \arg \max_{t \in [1, T-n+1]} \left(\frac{1}{n} \sum_{i=0}^{n-1} (\mathcal{I}\left(h_S^{t+i}\right) \right), \tag{5}$$

where n is the number of ensemble models.

3.3 Surrogate Stability Analysis in Bi-level UEs

Although Theorem 3.1 is analyzed in the single-level setting, the conclusion that weaker surrogate robustness leads to stronger poisoning effects also holds for bi-level optimization. Accordingly, we propose a perturbation selection strategy for bi-level methods based on surrogate model stability.

A key challenge in the bi-level setting is that the surrogate model evolves dynamically throughout the optimization process. Eventually, the surrogate model stops updating and the perturbation becomes fixed. Consequently, there is no definitive surrogate model with which to measure its stability. Taking EM [19] as an example, the method alternates between updating the surrogate model and optimizing the perturbations until a target accuracy threshold is reached. Using the EM method to generate perturbations with budget $\epsilon = 1, 2, 4, 8$ on CIFAR-10, we observe a substantial variation in poisoning performance under the same hyperparameter settings: test accuracy can range from 10% to 90%. This is attributed to stochastic factors such as the initialization of surrogate model parameters and sampling noise in the optimization process. To analyze what happened in the bi-level, we first track the surrogate model stability during this iterative process, using the metric $\mathcal{I}(h_S^t)$ for $t = 1, \ldots, m$. The results are shown in Figure 3 and appendix Figure 8. Starting from different random initializations, all surrogate models exhibit a sharp increase in instability after a certain number of iterations (around 20). For instance, the peak value of the blue curve exceeds 80,000. In contrast, during standard model training, the stability metric typically remains below 150. This indicates that **bi-level optimization also implicitly relies on highly unstable surrogate models for generating effective perturbations.**

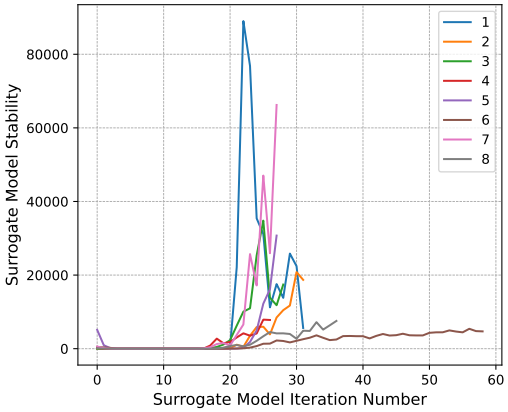


Figure 3: Variation of the surrogate model's stability during perturbation generation in the bi-level optimization method.

Although bi-level optimization tends to drive the surrogate toward higher instability, this instability is not deterministic; it varies across runs due to factors such as random initialization and sampling noise during optimization. Consequently, different runs of the same procedure yield surrogate models with varying instability levels, which may lead to different poisoning performances. Therefore, explicitly selecting a surrogate is still necessary. We hypothesize that if the surrogate model exhibits consistently higher instability throughout optimization, the resulting perturbations are more likely to produce stronger poisoning effects. Motivated by this intuition, we compute the average stability

metric over all intermediate surrogate models, denoted by

$$\bar{\mathcal{I}}(h) = \frac{1}{m}\sum_{t=1}^{m}\mathcal{I}(h_S^t). \tag{6}$$

We argue that a larger $\bar{\mathcal{I}}(h)$ indicates greater poisoning effectiveness of the perturbations.

4 Experiments

As established in Section 3, our theoretical analysis demonstrates that the upper bound of the target model's generalization error is fundamentally related to the robustness of the surrogate model. Based on this, we propose a stability-based surrogate selection criterion for single-level methods. We further extend this stability-based framework to analyze the dynamic evolution in bi-level methods. In this chapter, we validate these insights through comprehensive experiments. First, we employ the most unstable model as the surrogate to enhance the poisoning performance of single-level methods. Next, we significantly enhanced the bi-level methods through explicit stability-based selection. Finally, we show the computational complexity and sensitivity analysis of stability metrics.

4.1 Experimental Setup

Dataset. We conducted experiments on five widely-used datasets: CIFAR-10/100 [24], Pets [31], Car [23], and ImageNet [8].

Surrogate & target models. We use ResNet18 [15] as the surrogate model, while the target models with different architectures include MobileNetV2 [17], GoogleNet [36], DenseNet121 [18] and ResNet18. The standard training strategy for all models employs the SGD optimizer with a momentum of 0.9, an initial learning rate of 0.1 decayed via CosineAnnealingLR. The training epochs are set to 60, except for 100 epochs on ImageNet-10 and 300 epochs on ImageNet.

Metric. We train target models on the poisoned datasets and evaluate their accuracy on test datasets. The lower the test accuracy, the stronger the poisoning performance of perturbations. The perturbation constraint is the ℓ_∞-norm. Theorem 3.1 applies through norm equivalence shown in Appendix A.9.

4.2 Enhancing Poisoning for Single-level Methods

This subsection experimentally verifies that the stability-based surrogate model selection strategy in Section 3.2 can enhance the single-level methods. In the single-level setting, the surrogate model is trained on the clean training set. We identify three key points of comparison: i) **Well-trained surrogate**: the final epoch model as surrogate. ii) **Stability-guided surrogate**: the surrogate that maximizes the stability metric. iii) **Optimal surrogate**: the surrogate that achieves the lowest test accuracy in practice. We evaluate with two single-model methods: TAP [10] (class-wise perturbation), and HYPO-p2 [37] (sample-wise perturbation), and an ensemble-based method: SEP [5].

Experiments are conducted on five datasets with different perturbation budgets $\|\boldsymbol{\eta}\|_\infty \leq \epsilon$. For each poisoning method, the ResNet18 is used as the surrogate network architecture. We train surrogate models on the clean train dataset and evaluate their stability metric $\mathcal{I}(h_S^t)$ at each training epoch. For each epoch t, we generate the corresponding perturbation $\boldsymbol{\eta}_t$ following the poisoning method's original optimization procedure, train the target model on the poisoned dataset, and record the test accuracy. This yields a comprehensive trajectory of stability metric $\mathcal{I}(h_S^t)$ and their corresponding poisoning effects across the entire training process. As shown in Table 1, the well-trained model is not an optimal surrogate to optimize perturbations. Across all experiments, the stability-guided surrogate consistently yields stronger poisoning than the well-trained surrogate. Moreover, in the majority of cases, the optimal surrogate also exhibits a higher stability metric than the well-trained surrogate. A representative case is TAP on CIFAR-10 with the $\epsilon = 1$. The well-trained surrogate of epoch 60 with $\mathcal{I}$=20.15 (the 5-th row of columns 4-5) yields an average accuracy of 46.02% (the 5-th row of last column), while the stability-guided surrogate of epoch 32 with $\mathcal{I}$=107.35 (the 6-th row of columns 4-5) achieves a much stronger poisoning effect, lowering the average accuracy to 22.98% (the 6-th row of last column). Importantly, the optimal surrogate is the stability-guided model in this case, demonstrating that our stability metric $\mathcal{I}(h_S^t)$ accurately identifies the optimal surrogate.

Table 1: Performance improvement on single surrogate model poisoning.

Method	Dataset	ϵ	Surrogate (Epoch)	$\mathcal{I}(h)$	ResNet18	MobileNet	GoogleNet	DenseNet	Avg
TAP [10]	CIFAR-10	2	Well-trained (60)	20.15	13.23	12.85	12.56	14.26	13.23
			Stability-guided (32)	**107.35**	10.63	10.34	11.70	11.11	10.95
			Optimal (49)	62.15	**10.06**	**10.10**	**10.17**	**10.16**	**10.12**
		1	Well-trained (60)	20.15	47.77	40.83	48.84	46.62	46.02
			Stability-guided (32)	**107.35**	**24.70**	**19.17**	**24.26**	**23.80**	**22.98**
			Optimal (32)	**107.35**	**24.70**	**19.17**	**24.26**	**23.80**	**22.98**
	Pets	8	Well-trained (60)	0.64	14.20	14.13	13.18	17.78	14.82
			Stability-guided (25)	**3.53**	11.16	10.68	**8.72**	13.46	11.01
			Optimal (7)	1.00	**6.63**	**7.03**	9.20	**9.06**	**7.98**
		4	Well-trained (60)	0.64	35.23	51.45	41.92	45.50	43.53
			Stability-guided (25)	**3.53**	27.86	35.56	30.09	37.86	32.84
			Optimal (7)	1.00	**19.88**	**19.34**	**25.49**	**24.21**	**22.23**
	Car	8	Well-trained (60)	0.50	5.72	12.16	6.98	7.67	8.13
			Stability-guided (25)	**3.48**	3.54	5.58	3.67	4.39	4.29
			Optimal (7)	0.46	**2.98**	**2.41**	**2.14**	**2.85**	**2.60**
		4	Well-trained (60)	0.50	38.33	55.78	50.95	47.58	48.16
			Stability-guided (25)	**3.48**	12.16	32.37	29.91	18.58	23.26
			Optimal (5)	0.25	**7.72**	**7.79**	**6.82**	**8.39**	**7.68**
	ImageNet-10	8	Well-trained (100)	0.16	33.00	29.40	31.60	35.20	32.30
			Stability-guided (35)	**2.52**	27.60	27.40	**26.40**	**30.60**	28.00
			Optimal (20)	1.79	**24.60**	**25.20**	26.80	31.40	**27.00**
		4	Well-trained (100)	0.16	39.00	37.60	38.00	44.20	39.70
			Stability-guided (35)	**2.52**	35.60	**35.40**	35.40	41.40	36.95
			Optimal (14)	1.23	**33.40**	36.80	**33.80**	**40.40**	**36.10**
HYPO-p2 [37]	CIFAR-10	2	Well-trained (60)	20.15	89.76	86.02	89.60	88.94	88.58
			Stability-guided (32)	**107.35**	78.04	73.84	79.21	76.82	76.97
			Optimal (4)	12.04	**64.72**	**62.77**	**64.26**	**63.32**	**63.77**
		1	Well-trained (60)	20.15	91.66	88.61	91.78	91.55	90.90
			Stability-guided (32)	**107.35**	82.19	79.82	83.17	82.60	81.95
			Optimal (4)	12.04	**77.74**	**73.30**	**78.75**	**77.13**	**76.73**
	Pets	8	Well-trained (60)	0.64	58.15	54.77	66.40	56.52	58.96
			Stability-guided (25)	**3.53**	38.88	38.00	48.61	35.84	40.33
			Optimal (11)	1.68	**19.47**	**19.61**	**22.92**	**18.26**	**20.07**
		4	Well-trained (60)	0.64	61.73	71.13	60.18	59.43	63.12
			Stability-guided (25)	**3.53**	51.52	46.11	58.01	49.97	51.40
			Optimal (11)	1.68	**33.60**	**33.87**	**42.26**	**32.32**	**35.51**
	Car	8	Well-trained (60)	0.50	32.56	39.98	42.03	37.91	38.12
			Stability-guided (25)	**3.48**	15.02	18.24	21.18	16.89	17.83
			Optimal (9)	0.63	**5.96**	**4.43**	**5.29**	**5.35**	**5.26**
		4	Well-trained (60)	0.50	41.26	43.09	50.27	48.75	45.84
			Stability-guided (25)	**3.48**	23.70	25.62	30.68	28.43	27.11
			Optimal (5)	0.25	**10.40**	**8.54**	**9.27**	**9.96**	**9.54**
	ImageNet-10	8	Well-trained (100)	0.16	57.80	56.80	61.80	62.00	59.60
			Stability-guided (35)	**2.52**	54.00	52.60	57.20	59.80	55.90
			Optimal (5)	0.62	**49.80**	**51.80**	**50.60**	**50.20**	**50.60**
		4	Well-trained (100)	0.16	65.20	60.00	64.80	67.00	64.25
			Stability-guided (35)	**2.52**	59.60	57.00	62.00	65.60	61.05
			Optimal (17)	1.88	**54.80**	**53.80**	**53.00**	**54.60**	**54.05**

To further validate the relationship between surrogate instability and poisoning effectiveness, we evaluate all 60 checkpoints of a ResNet18 surrogate trained on CIFAR-10 under an ℓ_∞ perturbation budget of 1/255. We compute the Pearson correlation between surrogate instability and target model test accuracy across four representative UE methods. The correlation coefficients range from -0.50 to -0.61 (all $p < 0.001$), indicating a consistent negative correlation across the entire training trajectory. Detailed scatter plots and regression results are provided in Appendix B.3.

In addition to the single surrogate model, SEP [5] integrates multiple training checkpoints to generate perturbations. Specifically, SEP uniformly samples n checkpoints from the training process and uses their ensemble to optimize perturbations. Based on theoretical analysis, we proposed the surrogate selection strategy in Equation (5). We compare two surrogate selection strategies on CIFAR-10 and CIFAR-100: the uniform sampling and the stability-guided selection. For $n = 5$, the most unstable windows are (32,33,34,35,36) on CIFAR-10 and (31,32,33,34,35) on CIFAR-100. We generate perturbations with budgets $\epsilon = 2$ and 4 and evaluate poisoning performance across four target models. Table 2 summarizes the results. Perturbations generated using the most unstable window consistently yield stronger poisoning than uniformly sampled models. For example, on CIFAR-100 with the $\epsilon = 4$, the original SEP achieves an average accuracy of 40.33% (the 6-th row of last column), while

Table 2: Performance improvement on ensemble-based surrogate poisoning.

Method	Dataset	ϵ	Surrogate (Epochs)	ResNet18	MobileNet	GoogleNet	DenseNet	Avg
SEP [5]	CIFAR-10	4	Uniform sampling (24,48,72,96,120)	17.03	18.86	20.56	19.47	18.98
			Stability-guided (32,33,34,35,36)	**6.63**	**9.03**	**8.69**	**8.08**	**8.11**
		2	Uniform sampling (24,48,72,96,120)	53.56	89.40	56.41	51.73	62.78
			Stability-guided (32,33,34,35,36)	**34.85**	**29.06**	**42.49**	**31.64**	**34.51**
	CIFAR-100	4	Uniform sampling (24,48,72,96,120)	42.67	30.28	46.25	42.13	40.33
			Stability-guided (31,32,33,34,35)	**1.10**	**1.00**	**1.00**	**1.00**	**1.03**
		2	Uniform sampling (24,48,72,96,120)	75.16	68.29	66.37	64.02	68.46
			Stability-guided (31,32,33,34,35)	**1.14**	**1.00**	**1.00**	**1.00**	**1.04**

our selection reduces it to 1.03% (the 7-th row of last column), significantly improving it. Similar gains are observed across both datasets and budgets.

4.3 Analysis and Enhancement of Bi-level Methods

This subsection experimentally verifies that the perturbation selection strategy for bi-level methods based on surrogate model stability in Section 3.3 can enhance the bi-level methods. We compute the average stability metric $\bar{\mathcal{I}}(h)$ according to Equation (6). We adopt the EM method to generate perturbations on CIFAR-10 with budgets $\epsilon = 1, 2, 4, 8$, using ResNet18 as the surrogate model. For each setting, we repeat the procedure 8 times and evaluate the poisoning performance on four target models. As shown in the Table 3 and appendix Table 7, the run with the largest $\bar{\mathcal{I}}(h)$ consistently achieves the strongest poisoning effect, confirming the effectiveness of our selection in the bi-level setting. It is worth noting that after applying our proposed selection criterion to EM-generated perturbations, the poisoning accuracy under a perturbation budget of 1 is reduced from 94.20% to 10.67%. This dramatic improvement transforms EM from an almost ineffective poisoning method to one that achieves near-optimal poisoning performance. Importantly, this improvement is obtained without any modification to the EM algorithm itself; by merely selecting the most unstable surrogate models, we are able to identify the most effective poisoning perturbations. To the best of our knowledge, this is the first work in the UEs setting to achieve approximately 10% poisoning accuracy on CIFAR-10 with a perturbation budget of 1. Moreover, this result surpasses the state-of-the-art methods, PIL [26] and Armor [14], under a substantially larger perturbation budget of 8.

Table 3: Performance improvement on bi-level methods.

Method	ϵ	$\bar{\mathcal{I}}(h)$	RN	MB	GL	DN	Avg
EM [19]	2	377.00	28.97	39.70	22.92	45.56	34.29
		496.05	29.16	19.81	41.56	25.06	28.90
		520.34	34.47	89.35	**10.01**	10.04	34.47
		638.56	38.20	18.91	16.87	26.54	25.13
		763.90	11.42	10.26	14.41	15.17	12.82
		1270.27	10.64	10.79	15.66	11.97	12.27
		1977.59	**10.11**	11.17	10.61	11.33	10.80
		5082.22	**10.11**	**10.29**	10.10	**10.34**	**10.21**
	1	1398.99	94.15	92.90	94.61	95.14	94.20
		1756.68	94.24	92.86	94.46	94.89	94.11
		2005.78	94.79	92.97	94.50	94.91	94.29
		2583.57	94.41	92.40	94.53	95.25	94.15
		2848.87	40.18	29.85	33.66	95.23	49.73
		4020.47	12.95	15.69	10.00	10.00	12.16
		7047.32	15.57	12.39	11.99	10.11	12.49
		10989.26	**10.36**	**12.37**	**9.99**	**9.96**	**10.67**
PIL [26]	8	– –	12.27	13.97	18.57	17.70	15.62
Armor [14]	8	– –	12.85	– –	– –	11.09	11.97

4.4 Evaluation on Modern Pretrained Architectures

To further investigate whether the proposed surrogate stability criterion remains effective for modern pretrained models, we conduct additional experiments using four representative pretrained target architectures: ResNet-18, ConvNeXt-Tiny, ViT-B/16, and Swin-Tiny. Unlike the previous experiments that train target models from scratch, these experiments evaluate the poisoning effectiveness when pretrained models are adapted to poisoned CIFAR-10. We consider two commonly used adaptation strategies: linear probing and full fine-tuning. All target models are trained for 20 epochs using AdamW, with learning rates of 1×10^{-5} for the pretrained backbone and 1×10^{-3} for the classification head. Lower test accuracy indicates stronger poisoning effectiveness. Detailed experimental results are provided in Appendix B.5.

We first evaluate the sample-wise TAP, where a separate perturbation is optimized for each training sample ($\epsilon = 4$). We compare the well-trained surrogate checkpoint with the stability-guided check-

point selected by our metric, as well as the empirically optimal checkpoint. The results show that our stability metric successfully identifies the empirically optimal checkpoint (epoch 25), which exhibits substantially higher instability than the well-trained checkpoint (epoch 60). Under full fine-tuning, this selection reduces the average target accuracy from 89.93% to 76.03%.

We further investigate whether the proposed criterion generalizes to bi-level optimization. We evaluating eight independently generated EM perturbation sets ($\epsilon = 4$) with different surrogate stability scores. The results show that the perturbation set associated with the highest stability score (425.19) achieves the strongest overall poisoning performance under both adaptation strategies. Specifically, it reduces the average target accuracy from 89.97% to 77.41% under linear probing, and from 97.03% to 15.47% under full fine-tuning.

4.5 Generalization to Additional Surrogate Architectures.

We further evaluate stability-guided selection using different surrogate architectures. We consider MobileNetV2, DenseNet121, and GoogleNet as surrogate models, and employ TAP to generate class-wise perturbations on CIFAR-10 and ImageNet-10. The resulting poisoned datasets are evaluated on four target architectures: ResNet18, MobileNetV2, GoogleNet, and DenseNet121. For both datasets, we compare three checkpoint selection strategies: the well-trained checkpoint, the stability-guided checkpoint, and the empirically optimal checkpoint. Checkpoint selection is performed independently within each surrogate's training trajectory. More Detailed results are provided in Appendix B.6.

On CIFAR-10, we use a perturbation budget of $\epsilon = 2$ and select checkpoints from a 60-epoch training trajectory. The results show that stability-guided selection improves poisoning effectiveness across all three surrogate architectures. Specifically, it reduces the average target test accuracy from 63.98% to 41.39% for MobileNetV2, from 39.39% to 31.74% for DenseNet121, and from 82.09% to 69.45% for GoogleNet. Averaged across the three surrogate architectures, the target accuracy decreases from 61.82% to 47.52%, corresponding to an improvement of 14.30 percentage points over the well-trained checkpoints. We further extend the evaluation to ImageNet-10, using a perturbation budget of $\epsilon = 4$ and a 100-epoch training trajectory. Stability-guided selection reduces the average target accuracy from 68.05% to 56.80% for DenseNet121, from 68.60% to 67.20% for GoogleNet, and from 69.30% to 68.00% for MobileNetV2. Overall, the average target accuracy decreases from 68.65% to 64.00%, corresponding to an improvement of 4.65 percentage points over the well-trained checkpoints.

4.6 Computational Cost & Sensitivity Analysis

The computational complexity of the stability metric is relatively low, and the selection of the surrogate model is not sensitive to variations in parameters. We do not explicitly compute the full Hessian matrix. Instead, we compute Hessian-vector products and use PyHessian with k Lanczos iterations to approximate the dominant eigenvalue. The sensitivity analysis and runtime on ResNet18/CIFAR-10 are shown in the Table 4. Specifically, we test with 200/2000 samples and k=20/200, and report the most unstable epoch together with the corresponding stability metric. The results show that different parameter settings have only a limited effect on selecting the optimal surrogate model. Although the absolute value of changes to some extent, the selected epoch remains highly stable. Importantly, for the setting k=20, samples=200, the runtime is only 30 seconds per checkpoint, which is practically affordable.

Table 4: Computational cost & sensitivity analysis.

Step (k)	Samples	Epoch	$\mathcal{I}(h)$	Time/checkpoint
20	200	32	107.35	30s
20	2000	33	123.87	5 min
200	200	32	135.93	10 min

5 Conclusion

We challenge the common practice of using a well-trained model as the surrogate for generating unlearnable examples. We theoretically show that the target model's test accuracy is closely related to the robustness of the surrogate, and accordingly propose a stability-based surrogate selection strategy. We further find that bi-level methods also implicitly exploit unstable surrogates for perturbation generation. Extensive experiments show that our strategy consistently improves the effectiveness of various existing methods, underscoring the importance of surrogate stability in unlearnable examples.

Acknowledgments and Disclosure of Funding

This research was partially supported by the Macau Science and Technology Development Fund Grant Nos. 0022/2026/ITP1, the Basic and Applied Basic Research Foundation of Guangdong Province (Grant No. 2024A1515011755), the Department of Education of Guangdong Province (Grant No. 2026KQNCX173), and the Foshan Social Science Fund of Guangdong Province (Grant No. 2026-GJ009).

References

[1] Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical learning periods in deep networks. In *International Conference on Learning Representations*, pages 7191–7200, 2019.

[2] Maksym Andriushchenko and Nicolas Flammarion. Towards understanding sharpness-aware minimization. In *International conference on machine learning*, pages 639–668. PMLR, 2022.

[3] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. In *International Conference on Learning Representations*, pages 1470–1483, 2017.

[4] Chaochao Chen, Jiaming Zhang, Yuyuan Li, and Zhongxuan Han. One for all: A universal generator for concept unlearnability via multi-modal alignment. In *Proceedings of International Conference on Machine Learning*, pages 7700–7711, 2024.

[5] Sizhe Chen, Geng Yuan, Xinwen Cheng, Yifan Gong, Minghai Qin, Yanzhi Wang, and Xiaolin Huang. Self-ensemble protection: Training checkpoints are good data protectors. In *International Conference on Learning Representations*, pages 33231–33240, 2023.

[6] Jeremy Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations*, pages 10449–10461, 2021.

[7] Nigel W. Daw. Critical periods and amblyopia. *Archives of Ophthalmology*, 116(4):502–505, 04 1998.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[9] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, pages 2315–2324, 2021.

[10] Liam Fowl, Micah Goldblum, Ping-yeh Chiang, Jonas Geiping, Wojciech Czaja, and Tom Goldstein. Adversarial examples make strong poisons. In *Advances in Neural Information Processing Systems*, pages 30339–30351, 2021.

[11] Jonathan Frankle, David J. Schwab, and Ari S. Morcos. The early phase of neural network training. In *International Conference on Learning Representations*, pages 4946–4944, 2020.

[12] Shaopeng Fu, Fengxiang He, Yang Liu, Li Shen, and Dacheng Tao. Robust unlearnable examples: Protecting data against adversarial learning. In *International Conference on Learning Representations*, pages 5826–5839, 2022.

[13] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*, pages 2232–2241. PMLR, 2019.

[14] Xueluan Gong, Yuji Wang, Yanjiao Chen, Haocheng Dong, Yiming Li, Mengyuan Sun, Shuaike Li, and Qian Wang. Armor: Shielding unlearnable examples against data augmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(4):3988–4004, 2026.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[16] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural computation*, 9(1):1–42, 1997.

[17] Andrew G Howard. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, pages 1–9, 2017.

[18] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017.

[19] Hanxun Huang, Xingjun Ma, Sarah Monazam Erfani, James Bailey, and Yisen Wang. Unlearnable examples: Making personal data unexploitable. In *International Conference on Learning Representations*, pages 2769–2781, 2021.

[20] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, pages 125–136, 2019.

[21] Dayal Singh Kalra and Maissam Barkeshli. Phase diagram of early training dynamics in deep neural networks: effect of the learning rate, depth, and width. *Advances in Neural Information Processing Systems*, 36:51621–51662, 2023.

[22] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, pages 2874–2883, 2017.

[23] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *2013 IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013.

[24] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[25] Qianxiao Li, Cheng Tai, and Weinan E. Stochastic modified equations and dynamics of stochastic gradient algorithms i: Mathematical foundations. *Journal of Machine Learning Research*, 20(40):1–47, 2019.

[26] Jinlin Liu, Wei Chen, and Xiaojin Zhang. Perturbation-induced linearization: Constructing unlearnable data with solely linear classifiers. In *International Conference on Learning Representations*, pages 98080–98108, 2026.

[27] Shuang Liu, Yihan Wang, and Xiao-Shan Gao. Game-theoretic unlearnable example generator. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 21349–21358, 2024.

[28] Yixin Liu, Kaidi Xu, Xun Chen, and Lichao Sun. Stable unlearnable example: Enhancing the robustness of unlearnable examples via stable error-minimizing noise. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3783–3791, 2024.

[29] Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. In *International Conference on Machine Learning*, pages 23190–23211. PMLR, 2023.

[30] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021.

[31] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3498–3505. IEEE, 2012.

[32] Alethea Power, Yuri Burda, Harrison Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *ArXiv*, abs/2201.02177, 2022.

[33] Jie Ren, Han Xu, Yuxuan Wan, Xingjun Ma, Lichao Sun, and Jiliang Tang. Transferable unlearnable examples. In *International Conference on Learning Representations*, pages 33069–33078, 2023.

[34] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, pages 1–17, 2017.

[35] Samuel L. Smith and Quoc V. Le. A bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations*, pages 5660–5668, 2018.

[36] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[37] Lue Tao, Lei Feng, Jinfeng Yi, Sheng-Jun Huang, and Songcan Chen. Better safe than sorry: Preventing delusive adversaries with adversarial training. In *Advances in Neural Information Processing Systems*, volume 34, pages 16209–16225, 2021.

[38] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pages 1–5. IEEE, 2015.

[39] Derui Wang, Minhui Xue, Bo Li, Seyit Camtepe, and Liming Zhu. Provably unlearnable data examples. In *The Network and Distributed System Security (NDSS) Symposium*, pages 1–13, 2025.

[40] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of International Conference on Machine Learning*, pages 681–688, 2011.

[41] Rui Wen, Zhengyu Zhao, Zhuoran Liu, Michael Backes, Tianhao Wang, and Yang Zhang. Is adversarial training really a silver bullet for mitigating data poisoning? In *International Conference on Learning Representations*, pages 19185–19194, 2023.

[42] Lei Wu, Chao Ma, et al. How sgd selects the global minima in over-parameterized learning: A dynamical stability perspective. *Advances in Neural Information Processing Systems*, 31:8289–8298, 2018.

[43] Zeke Xie, Issei Sato, and Masashi Sugiyama. A diffusion theory for deep learning dynamics: Stochastic gradient descent exponentially favors flat minima. In *International Conference on Learning Representations*, pages 6034–6043, 2021.

[44] Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W Mahoney. Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE international conference on big data (Big data)*, pages 581–590. IEEE, 2020.

[45] Da Yu, Huishuai Zhang, Wei Chen, Jian Yin, and Tie-Yan Liu. Availability attacks create shortcuts. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2367–2376, 2022.

[46] Jiaming Zhang, Xingjun Ma, Qi Yi, Jitao Sang, Yu-Gang Jiang, Yaowei Wang, and Changsheng Xu. Unlearnable clusters: Towards label-agnostic unlearnable examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3984–3993, 2023.

[47] Junpeng Zhang, Qing Li, Liang Lin, and Quanshi Zhang. Two-phase dynamics of interactions explains the starting point of a dnn learning over-fitted features. *arXiv preprint arXiv:2405.10262*, 2024.

[48] Zhanpeng Zhou, Mingze Wang, Yuchen Mao, Bingrui Li, and Junchi Yan. Sharpness-aware minimization efficiently selects flatter minima late in training. In *The Thirteenth International Conference on Learning Representations*, pages 18336–18346, 2025.

[49] Yifan Zhu, Yibo Miao, Yinpeng Dong, and Xiao-Shan Gao. Why do unlearnable examples work: A novel perspective of mutual information. In *International Conference on Learning Representations*, pages 107849–107880, 2026.

[50] Zhanxing Zhu, Jingfeng Wu, Bing Yu, Lei Wu, and Jinwen Ma. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of International Conference on Machine Learning*, volume 97, pages 7654–7663, 2019.

Due to space constraints in the main text, we present theoretical proofs Section A and additional experiments Section B in the appendix.

A Preliminaries and Proofs

In the unlearnable examples setting, the dataset is divided into a clean training set $\mathcal{D}_C = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, a poisoned training set $\mathcal{D}_P = \{(\hat{\boldsymbol{x}}_i, y_i)\}_{i=1}^N$, and a clean test set $\mathcal{D}_T = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^M$. Each poisoned training sample is obtained by adding a perturbation to the corresponding clean training sample: $\hat{\boldsymbol{x}}_i = \boldsymbol{x}_i + \boldsymbol{\eta}_i, \quad \text{where } \|\boldsymbol{\eta}_i\|_p \leq \epsilon$. The surrogate model h_S is trained on the clean training set $\mathcal{D}_C$. Let h_S^t denote the surrogate model after t parameter update iterations. After convergence following T updates, we obtain the training trajectory of the surrogate model $\{h_S^t\}_{t=1}^T$. The target model h_T is trained to converge on the poisoned training set $\mathcal{D}_P$.

We first introduce the standard notion of loss, which measures how wrong the model is on a given training example and across the dataset.

Definition A.1 (Loss Function)**.** For a model h, the sample-wise loss on a given example $(\boldsymbol{x}, y)$ is denoted by

$$\ell(h(\boldsymbol{x}), y), \tag{7}$$

abbreviated as $\ell(h, \boldsymbol{x})$. The expected loss of h on a dataset $\mathcal{D}$ is defined as

$$\mathcal{L}(h, \mathcal{D}) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\big[\ell(h(\boldsymbol{x}), y)\big]. \tag{8}$$

Depending on which dataset is considered, $\mathcal{L}(h, \mathcal{D})$ corresponds to either training loss or generalization error. In the unlearnable examples setting, the surrogate model h_S is trained on the clean training set $\mathcal{D}_C$, while the target model h_T is trained on the poisoned training set $\mathcal{D}_P$. Their corresponding training losses are

$$\mathcal{L}(h_S, \mathcal{D}_C) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_C}\big[\ell(h_S(\boldsymbol{x}), y)\big], \tag{9}$$

$$\mathcal{L}(h_T, \mathcal{D}_P) = \mathbb{E}_{(\hat{\boldsymbol{x}},y)\sim\mathcal{D}_P}\big[\ell(h_T(\hat{\boldsymbol{x}}), y)\big]. \tag{10}$$

The generalization error of a model h is evaluated on the clean test set $\mathcal{D}_T$:

$$\mathcal{L}(h, \mathcal{D}_T) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_T}\big[\ell(h(\boldsymbol{x}), y)\big]. \tag{11}$$

Even without poisoning, a model may exhibit different losses on the clean training set and the clean test set due to distribution mismatch and finite-sample effects. We use the following quantity to capture this inherent discrepancy.

Definition A.2 (Intrinsic Generalization Gap)**.** For a model h, the intrinsic generalization gap is defined as

$$\mathcal{R}(h) = \mathcal{L}(h, \mathcal{D}_T) - \mathcal{L}(h, \mathcal{D}_C). \tag{12}$$

We next quantify the performance shift caused specifically by the poisoning perturbation.

Definition A.3 (Perturbation-Induced Poisoning Gap)**.** For a model h, the perturbation-induced poisoning gap is defined as

$$\Delta(h, \boldsymbol{\eta}) = \mathcal{L}(h, \mathcal{D}_C) - \mathcal{L}(h, \mathcal{D}_P). \tag{13}$$

Based on the notions above, we can decompose the generalization error of the target model into three interpretable components.

Lemma A.4. *For the target model h_T trained on the poisoned training set $\mathcal{D}_P$, its generalization error on the clean test set $\mathcal{D}_T$ can be written as*

$$\mathcal{L}(h_T, \mathcal{D}_T) = \mathcal{L}(h_T, \mathcal{D}_P) + \Delta(h_T, \boldsymbol{\eta}) + \mathcal{R}(h_T). \tag{14}$$

Proof.

$$\begin{aligned}
&\mathcal{L}(h_T, \mathcal{D}_T) && (15)\\
=&\mathcal{L}(h_T, \mathcal{D}_P) + \mathcal{L}(h_T, \mathcal{D}_C) - \mathcal{L}(h_T, \mathcal{D}_P) + \mathcal{L}(h_T, \mathcal{D}_T) - \mathcal{L}(h_T, \mathcal{D}_C) && (16)\\
=&\mathcal{L}(h_T, \mathcal{D}_P) + \Delta(h_T, \boldsymbol{\eta}) + \mathcal{R}(h_T). && \textit{Equations } (11) \textit{to } (13)\\
& && (17)
\end{aligned}$$

□

According to Lemma A.4, the generalization error of the target model can therefore be decomposed into the training loss $\mathcal{L}(h_T, \mathcal{D}_P)$, the perturbation-induced poisoning gap $\Delta(h_T, \boldsymbol{\eta})$, and the intrinsic generalization gap $\mathcal{R}(h_T)$. When $\boldsymbol{\eta} = \mathbf{0}$, we have $\mathcal{D}_P = \mathcal{D}_C$ and $\Delta(h_T, \boldsymbol{\eta}) = 0$. In this case, the generalization error consists only of the training loss $\mathcal{L}(h_T, \mathcal{D}_P)$ and the intrinsic generalization gap $\mathcal{R}(h_T)$. Since modern optimization algorithms typically drive the training loss to a small value, the generalization error mainly arises from distribution mismatch between datasets, finite sample effects, and excessive model complexity. In the unlearnable examples setting, the dataset, model, and optimization algorithm are all fixed; therefore, we collectively refer to these factors as the *intrinsic generalization gap*. When the poisoning algorithm uses the surrogate model to generate perturbations, $\boldsymbol{\eta} = \mathcal{A}(h_S, \mathcal{D}_C)$, the generalization error of the target model $\mathcal{L}(h_T, \mathcal{D}_T)$ can become significantly larger. So we reasonably assume that the perturbation-induced poisoning gap dominates the intrinsic generalization gap, namely, $\Delta(h_T, \boldsymbol{\eta}) \gg \mathcal{R}(h_T)$. In the following, we focus on the perturbation-induced poisoning gap $\Delta(h_T, \boldsymbol{\eta})$.

In practice, the perturbation is generated based on the surrogate model rather than the target model. As a result, under the same perturbation, the surrogate model and the target model may exhibit different perturbation-induced poisoning gaps. We use the following quantity to characterize the worst-case discrepancy between them along the surrogate training trajectory.

We denote the perturbation generated by the surrogate model h_S^t at the t-th iteration as $\boldsymbol{\eta}^t = \mathcal{A}(h_S^t, \mathcal{D}_C)$, and the corresponding poisoned training set as $\mathcal{D}_P^t$. Let h_T^t denote the target model trained on $\mathcal{D}_P^t$ until convergence.

Definition A.5 (Maximum Deviation Gap)**.** The maximum deviation gap is defined as

$$\Omega = \sup_t \left| \Delta(h_T^t, \boldsymbol{\eta}^t) - \Delta(h_S^t, \boldsymbol{\eta}^t) \right|. \tag{18}$$

Based on this quantity, we can relate the perturbation-induced poisoning gap of the target model to that of the surrogate model.

Lemma A.6. *For the target model h_T^t, the perturbation-induced poisoning gap caused by $\boldsymbol{\eta}^t$ has the following upper bound:*

$$\Delta(h_T^t, \boldsymbol{\eta}^t) \leq \left| \Delta(h_S^t, \boldsymbol{\eta}^t) \right| + \Omega. \tag{19}$$

Proof.

$$\begin{aligned}
\Delta(h_T^t, \boldsymbol{\eta}^t) &= \Delta(h_S^t, \boldsymbol{\eta}^t) + \Delta(h_T^t, \boldsymbol{\eta}^t) - \Delta(h_S^t, \boldsymbol{\eta}^t) && (20)\\
&\leq \left| \Delta(h_S^t, \boldsymbol{\eta}^t) \right| + \left| \Delta(h_T^t, \boldsymbol{\eta}^t) - \Delta(h_S^t, \boldsymbol{\eta}^t) \right| && (21)\\
&\leq \left| \Delta(h_S^t, \boldsymbol{\eta}^t) \right| + \Omega \qquad \textit{Equation (18)} && (22)
\end{aligned}$$

□

To further quantify the poisoning gap, we next derive an upper bound based on the local sensitivity of the loss to the input. Specifically, the bound depends on the input gradient and Hessian of the loss function.

Lemma A.7. *For a model h, the absolute value of the perturbation-induced poisoning gap $|\Delta(h, \boldsymbol{\eta})|$ admits the following upper bound:*

$$|\Delta(h, \boldsymbol{\eta})| \leq \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_C} \left[\|\boldsymbol{\eta}\|_2 \cdot \|\nabla_{\boldsymbol{x}} \ell(h, \boldsymbol{x})\|_2 + \frac{1}{2} \|\boldsymbol{\eta}\|_2^2 \cdot \rho(H_{\boldsymbol{x}} \ell(h, \boldsymbol{x})) + \left| o(\|\boldsymbol{\eta}\|_2^2) \right| \right]. \tag{23}$$

Here, $\nabla_{\boldsymbol{x}} \ell(h, \boldsymbol{x})$ is the gradient of the loss function with respect to the input $\boldsymbol{x}$, $H_{\boldsymbol{x}} \ell(h, \boldsymbol{x})$ is the Hessian matrix of the loss function with respect to the input $\boldsymbol{x}$, $\|\cdot\|_2$ denotes the Euclidean norm, $\rho(A)$ is the spectral radius of a matrix $A \in \mathbb{R}^{n \times n}$, $\rho(A) = \max_{1 \leq i \leq n} |\lambda_i|$, λ_i is the eigenvalue of A and $o(\|\boldsymbol{\eta}\|_2^2)$ denotes a higher-order infinitesimal term with respect to $\|\boldsymbol{\eta}\|_2^2$.

Proof.

$$|\Delta(h,\boldsymbol{\eta})| = |\mathcal{L}(h,\mathcal{D}_C) - \mathcal{L}(h,\mathcal{D}_P)| \tag{24}$$

$$= \left|\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_C}[\ell(h(\boldsymbol{x}),y)] - \mathbb{E}_{(\hat{\boldsymbol{x}},y)\sim\mathcal{D}_P}[\ell(h(\hat{\boldsymbol{x}}),y)]\right| \tag{25}$$

$$= \left|\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_C}[\ell(h(\boldsymbol{x}),y) - \ell(h(\boldsymbol{x}+\boldsymbol{\eta}),y)]\right| \tag{26}$$

$$= \left|\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_C}[\ell(h(\boldsymbol{x}+\boldsymbol{\eta}),y) - \ell(h(\boldsymbol{x}),y)]\right| \tag{27}$$

$$= \left|\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_C}\left[\boldsymbol{\eta}^T\nabla_{\boldsymbol{x}}\ell(h(\boldsymbol{x}),y) + \frac{1}{2}\boldsymbol{\eta}^T H_{\boldsymbol{x}}\ell(h(\boldsymbol{x}),y)\boldsymbol{\eta} + o(\|\boldsymbol{\eta}\|_2^2)\right]\right| \tag{28}$$

$$\leq \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_C}\left[\left|\boldsymbol{\eta}^T\nabla_{\boldsymbol{x}}\ell(h(\boldsymbol{x}),y)\right| + \frac{1}{2}\left|\boldsymbol{\eta}^T H_{\boldsymbol{x}}\ell(h(\boldsymbol{x}),y)\boldsymbol{\eta}\right| + \left|o(\|\boldsymbol{\eta}\|_2^2)\right|\right] \tag{29}$$

$$\leq \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_C}\left[\|\boldsymbol{\eta}\|_2\cdot\|\nabla_{\boldsymbol{x}}\ell(h(\boldsymbol{x}),y)\|_2 + \frac{1}{2}\|\boldsymbol{\eta}\|_2^2\cdot\rho(H_{\boldsymbol{x}}\ell(h(\boldsymbol{x}),y)) + \left|o(\|\boldsymbol{\eta}\|_2^2)\right|\right]. \tag{30}$$

□

We are now ready to combine the previous results to derive an upper bound on the generalization error of the target model. The key idea is to decompose the generalization error and control the perturbation-induced poisoning gap via the surrogate model and the maximum deviation gap. This leads to the following result.

Theorem A.8. *Suppose the surrogate model h_S is trained on the clean training set $\mathcal{D}_C$ for T iterations, yielding the trajectory $\{h_S^t\}_{t=1}^T$. At iteration t, the poisoning algorithm generates perturbations*

$$\boldsymbol{\eta}^t = \mathcal{A}(h_S^t, \mathcal{D}_C),$$

which induces the poisoned training set $\mathcal{D}_P^t$. Let h_T^t denote the target model trained on $\mathcal{D}_P^t$ until convergence. Then the clean generalization error of h_T^t satisfies

$$\begin{aligned}\mathcal{L}(h_T^t,\mathcal{D}_T) \leq \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_C}\big[\|\boldsymbol{\eta}^t\|_2\cdot\|\nabla_{\boldsymbol{x}}\ell(h_S^t,\boldsymbol{x})\|_2 + \frac{1}{2}\|\boldsymbol{\eta}^t\|_2^2\cdot\rho\big(H_{\boldsymbol{x}}\ell(h_S^t,\boldsymbol{x})\big) + \big|o(\|\boldsymbol{\eta}^t\|_2^2)\big|\,\big] \\ + \Omega + \mathcal{L}(h_T^t,\mathcal{D}_P^t) + \mathcal{R}(h_T^t).\end{aligned} \tag{31}$$

Here, $\nabla_{\boldsymbol{x}}\ell(h_S^t,\boldsymbol{x})$ is the gradient of the surrogate model loss with respect to the input $\boldsymbol{x}$, $H_{\boldsymbol{x}}\ell(h_S^t,\boldsymbol{x})$ is the Hessian matrix of the surrogate model loss with respect to the input $\boldsymbol{x}$, $\|\cdot\|_2$ denotes the Euclidean norm, $\rho(A)$ is the spectral radius of a matrix $A\in\mathbb{R}^{n\times n}$, $\rho(A)=\max_{1\leq i\leq n}|\lambda_i|$, λ_i is the eigenvalue of A and $o(\|\boldsymbol{\eta}\|_2^2)$ is a higher-order infinitesimal term with respect to $\|\boldsymbol{\eta}\|_2^2$. Moreover, $\mathcal{R}(h_T^t)$ is the intrinsic generalization gap of the target model, $\mathcal{L}(h_T^t,\mathcal{D}_P^t)$ is the training loss of the target model, and Ω is the maximum deviation gap.

Proof.

$$\mathcal{L}(h_T^t,\mathcal{D}_T) = \mathcal{L}(h_T^t,\mathcal{D}_P^t) + \Delta(h_T^t,\boldsymbol{\eta}^t) + \mathcal{R}(h_T^t) \qquad \textit{Equation (14)} \tag{32}$$

$$\leq \left|\Delta(h_S^t,\boldsymbol{\eta}^t)\right| + \Omega + \mathcal{L}(h_T^t,\mathcal{D}_P^t) + \mathcal{R}(h_T^t) \qquad \textit{Equation (19)} \tag{33}$$

$$\leq \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_C}\big[\|\boldsymbol{\eta}^t\|_2\cdot\|\nabla_{\boldsymbol{x}}\ell(h_S^t,\boldsymbol{x})\|_2 + \frac{1}{2}\|\boldsymbol{\eta}^t\|_2^2\cdot\rho\big(H_{\boldsymbol{x}}\ell(h_S^t,\boldsymbol{x})\big) \tag{34}$$

$$+ \big|o(\|\boldsymbol{\eta}^t\|_2^2)\big|\,\big] + \Omega + \mathcal{L}(h_T^t,\mathcal{D}_P^t) + \mathcal{R}(h_T^t). \qquad \textit{Equation (23)} \tag{35}$$

□

The previous discussion was about adding perturbations to all training data. Now we consider a case of partial poisoning, namely Partial poisoning dataset $\mathcal{D}_\beta = \{\mathcal{D}_C, \mathcal{D}_P\}$ contains N samples, including $\mathcal{D}_C$ contains βN samples from a training dataset, $\mathcal{D}_P$ contains $(1-\beta)N$ samples from the poisoning dataset, $\beta\in[0,100\%]$. The model trained to converge on some poisoning datasets is called h_β, and its error on the training dataset is

$$\mathcal{L}(h_T,\mathcal{D}_T) = \Delta(h_T,\boldsymbol{\eta}(\beta)) + \mathcal{L}(h_T,\mathcal{D}_\beta) + \mathcal{R}(h_T). \tag{36}$$

Proof.

$$\mathcal{L}(h_T, \mathcal{D}_T) = \Delta(h_T, \boldsymbol{\eta}(\beta)) + \mathcal{L}(h_T, \mathcal{D}_\beta) + \mathcal{R}(h_T) \qquad \textit{Equation (14)} \tag{37}$$

□

Corollary A.9 (Generalization Bound under ℓ_∞-constrained Perturbations)**.** *Under the assumptions of Theorem A.8, suppose that the perturbation satisfies $\|\eta^t\|_\infty \leq \epsilon$. Then, for a fixed input dimension d, the clean generalization error of the target model h_t^T admits the following upper bound:*

$$\begin{aligned} L(h_t^T, \mathcal{D}_T) \leq & \mathbb{E}_{x \sim \mathcal{D}_C} \left[\sqrt{d}\epsilon \|\nabla_x \ell(h_t^S, x)\|_2 + \frac{d\epsilon^2}{2} \rho\big(H_x \ell(h_t^S, x)\big) + o(d\epsilon^2) \right] \\ & + \Omega + L(h_t^T, \mathcal{D}_P^t) + R(h_t^T). \end{aligned} \tag{38}$$

Proof. Since $\|\eta^t\|_\infty \leq \epsilon$, the norm equivalence gives

$$\|\eta^t\|_2 \leq \sqrt{d}\|\eta^t\|_\infty \leq \sqrt{d}\epsilon. \tag{39}$$

Substituting this inequality into the perturbation-dependent terms in Theorem A.8 directly yields the stated bound. □

B Supplementary Experiments

The supplementary experiments cover: 1) verification of theoretical assumptions in Section B.1, 2) three-phase evolution of model robustness during training in Section B.2, 3) correlation analysis between surrogate instability and poisoning effectiveness across training checkpoints in Section B.3, 4) analysis and enhancement of bi-level poisoning methods in Section B.4, 5) additional results on modern pre-trained architectures under different fine-tuning strategies in Section B.5, 6) generalization to additional surrogate architectures across CIFAR-10 and ImageNet-10 in Section B.6, 7) generalization to additional UE generation objectives in Section B.7, 8) online early stopping for bi-level optimization in Section B.8, 9) ineffectiveness of robustness surrogate models in Section B.9, 10) empirical validation of the proposed stability metric in Section B.10, and 11) nonlinear degradation under partial poisoning in Section B.11.

B.1 Verification of Theoretical Hypotheses

According to Lemma A.4, the generalization error of the target model can therefore be decomposed into the training loss $\mathcal{L}(h_T, \mathcal{D}_P)$, the perturbation-induced poisoning gap $\Delta(h_T, \boldsymbol{\eta})$, and the intrinsic generalization gap $\mathcal{R}(h_T)$. In Theorem 3.1, we assume that the perturbation-induced poisoning gap dominates the intrinsic generalization gap, namely, $\Delta(h_T, \boldsymbol{\eta}) \gg \mathcal{R}(h_T)$. On the CIFAR-10 dataset, we evaluate the classification accuracy of the target model under three different poisoning methods on the clean training set $\mathcal{D}_C$, and the clean test set $\mathcal{D}_T$, as shown in Table 5. The performance difference between the clean test set and the clean training set is very small (less than 1%), which we refer to as the intrinsic generalization gap. In contrast, the performance difference between the clean training set and the poisoned training set is large (over 50%), termed the perturbation-induced poisoning gap. This experiment validates $\Delta(h_T, \boldsymbol{\eta}) \gg \mathcal{R}(h_T)$.

B.2 Three-Phase Model Robustness in Training

The three-phase evolution of model stability during standard training—stable → unstable → stable—is a common phenomenon. We evaluate this on the CIFAR-10 dataset using different model architectures, including ResNet18, MobileNetV2, GoogleNet, DenseNet121, and VGG19, as shown in Figure 4. The x-axis represents training epochs, the blue curve denotes the gradient norm (referencing the left y-axis), and the red curve represents sharpness (referencing the right y-axis). Smaller values indicate greater model stability. Across different architectures, model stability consistently exhibits an initial stable phase, followed by an unstable phase, and finally a stable phase again. However, the degree of stability varies across architectures: GoogleNet and MobileNetV2 show the largest fluctuations, while ResNet18 and VGG19 exhibit relatively smaller fluctuations.

A more detailed inspection reveals two notable phenomena: i) Shift between gradient and sharpness. The rise and fall of the sharpness (red curve) consistently lag behind those of the gradient (blue

Table 5: The Intrinsic Generalization Gap $\mathcal{R}(h_T)$ of the Target Model (accuracy %).

Epoch	Attack type		
	Emax	HYPO-p4 [37]	TAP [10]
0	-0.92	-0.90	-0.90
5	-0.90	-0.90	-1.03
10	-0.90	-0.65	-0.90
15	-0.40	-0.13	-0.05
20	-0.76	-0.41	-0.50
25	-0.64	-0.19	-0.03
30	-0.25	-0.49	-0.39
35	-0.47	-0.38	-0.12
40	-0.49	-0.53	-0.53
45	-0.29	-0.66	-0.33
50	-0.24	-0.73	-0.40
55	-0.16	-0.64	-0.61
59	-0.83	-0.58	-0.42

curve). And the sharpness peak typically occurs later than the gradient peak. ii) Behavior in the convergent phase. Although the gradient norm decays to nearly zero as training converges (the blue shaded area), sharpness remains relatively high for an extended period before eventually approaching zero. These observations align with established findings in generalization theory and optimization that well-generalizing solutions often reside in flat minima where the sharpness is suppressed [9, 48, 2, 3, 16, 13, 22]. The stochastic gradient noise inherent in SGD helps the optimizer escape sharp minima and drift toward flatter regions [40, 35, 42, 25, 43, 50], which explains the eventual decay of the sharpness after sufficient training.

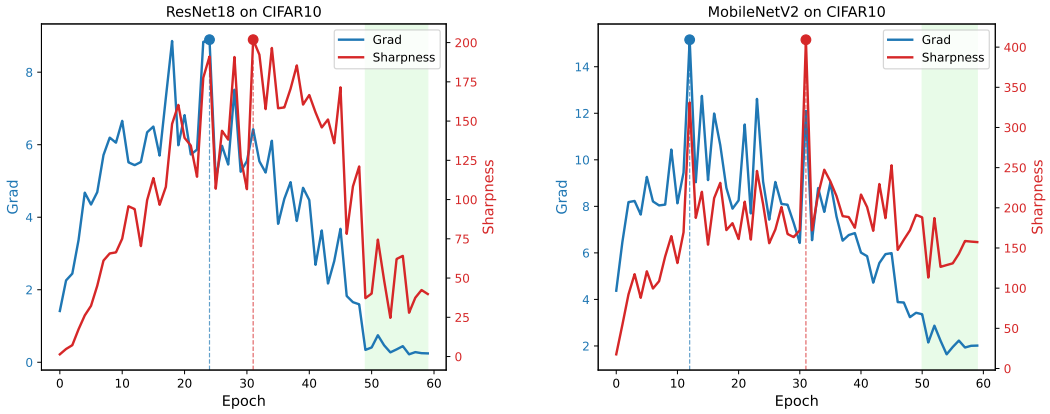
We further evaluate ResNet18 on different datasets, including CIFAR-100, Pets, Car, ImageNet10, and ImageNet, as shown in Figure 5. The stability dynamics vary significantly across datasets. CIFAR-100 exhibits the largest fluctuations, with sharpness reaching nearly 500 and gradient norm exceeding 20. On the other datasets, sharpness remains below 5 and gradient norm below 3. Note that these differences are partly due to variations in dataset size and number of label classes.

In addition, we examine stability dynamics when training large models on large-scale datasets. Specifically, we train ResNet50, ResNet101, and ResNet152 on ImageNet for 300 epochs. As shown in Figure 6, overall, larger model parameters lead to greater stability throughout the training process.

B.3 Correlation Analysis Across Training Checkpoints

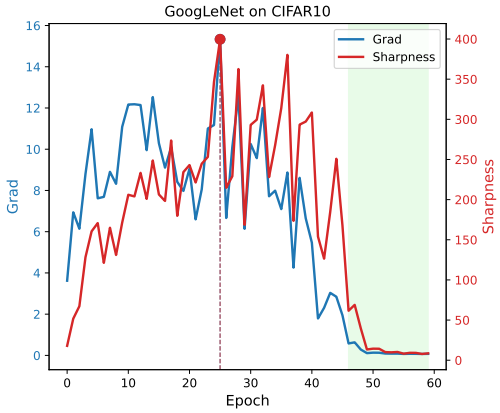
To further investigate the relationship between surrogate instability and poisoning effectiveness, we conduct a correlation analysis across 60 checkpoints of a ResNet-18 surrogate trained on CIFAR-10. Specifically, we generate perturbations using four representative UE methods: sample-wise HYPO-p2 [37], class-wise HYPO-p4 [37], EMax [10], and TAP [10], under an ℓ_∞ perturbation budget of $\epsilon = 1$. For each checkpoint, we evaluate the resulting poisoned dataset by training four target architectures, including ResNet-18, MobileNetV2, GoogleNet, and DenseNet121. We then calculate the average test accuracy across the four target models and examine its relationship with the corresponding surrogate stability metric $\mathcal{I}(h_S^t)$.

The corresponding scatter plots and linear regression fits are presented in Figure 7. HYPO-p2 (blue points) exhibits a relatively compact distribution, while HYPO-p4 (orange points), EMax (green points), and TAP (red points) show larger variations in poisoning performance. This may suggest that class-wise perturbations can more strongly exploit label-aligned shortcut features at highly unstable checkpoints, while their poisoning effectiveness is also more sensitive to checkpoint selection. We perform linear regression and Pearson correlation analysis to quantify this relationship. The statistical results are summarized in Table 6. All four methods exhibit statistically significant negative correlations, with Pearson correlation coefficients ranging from -0.50 to -0.61 (all $p < 0.001$). In particular, HYPO-sample exhibits a relatively small regression slope, whereas HYPO-class, EMax-class, and TAP-class show steeper negative slopes with larger standard errors.

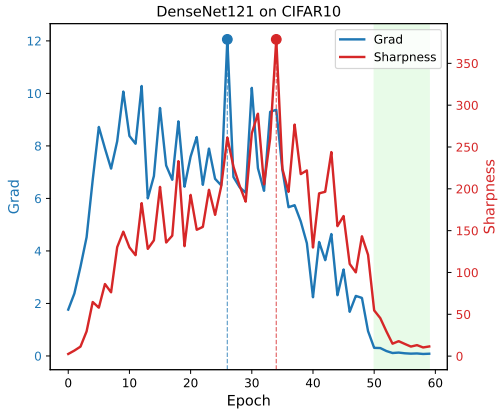


(a) ResNet18

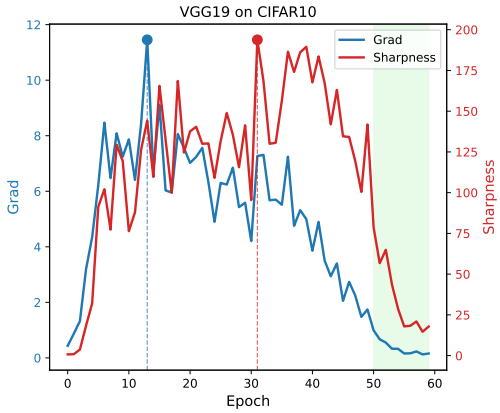
(b) MobileNetV2



(c) GoogleNet

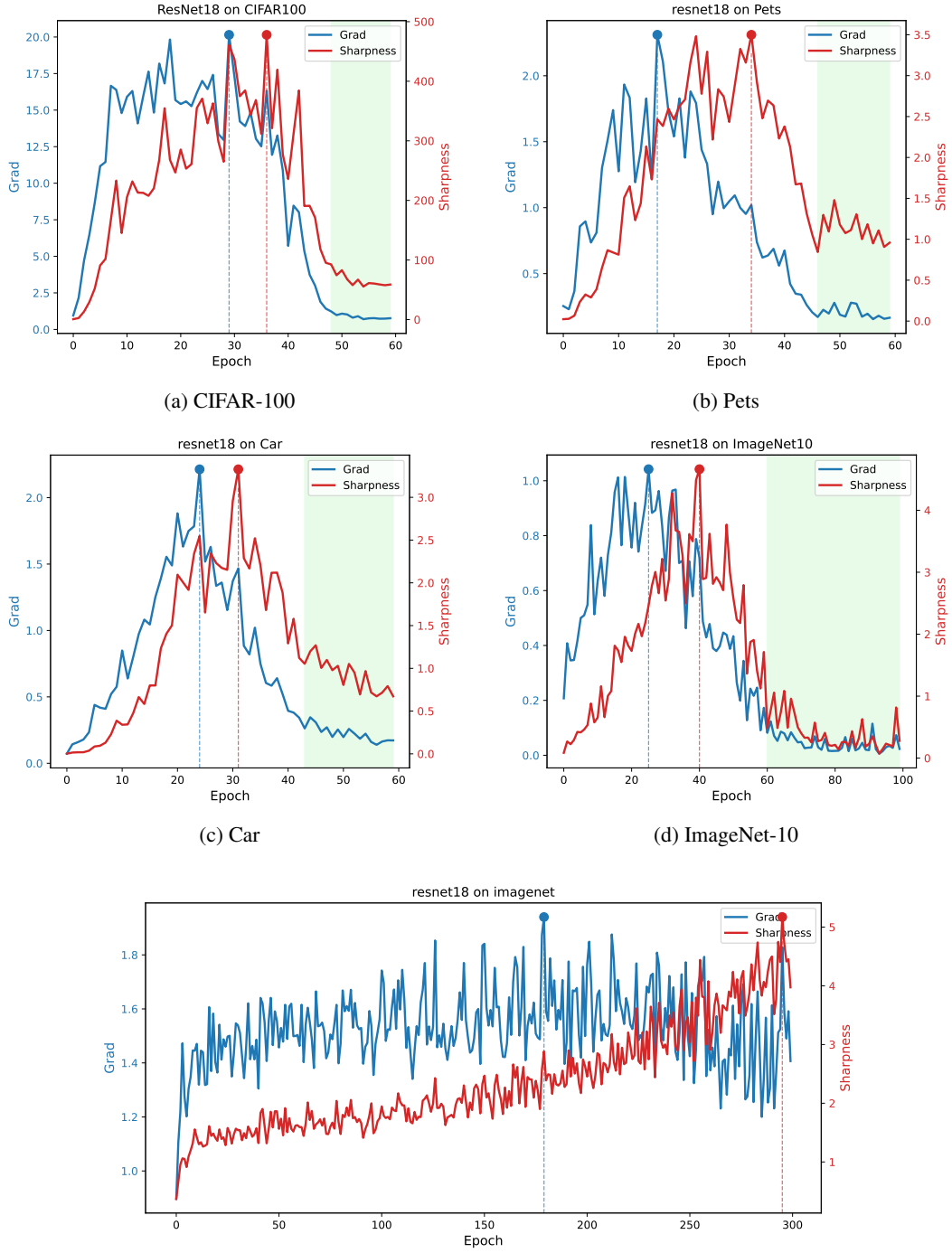


(d) DenseNet121



(e) VGG19

Figure 4: Different models' grad norm and sharpness in training process on CIFAR-10.



(a) CIFAR-100

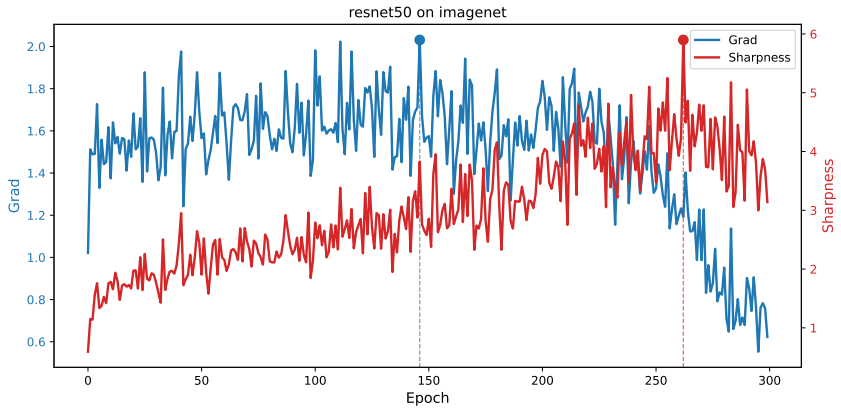
(b) Pets

(c) Car

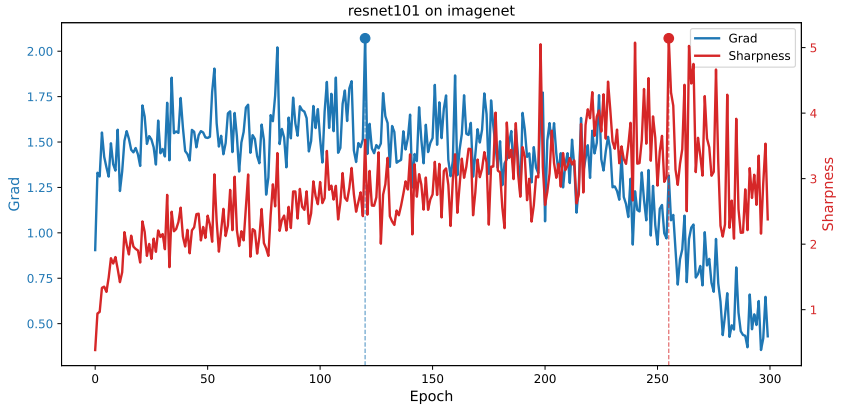
(d) ImageNet-10

(e) ImageNet

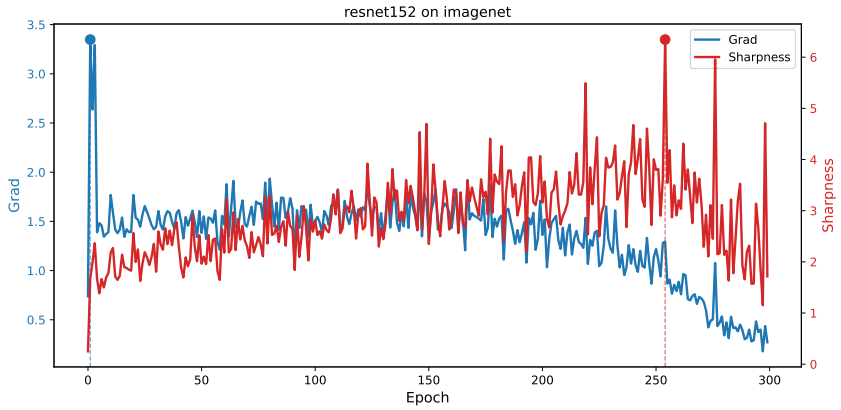
Figure 5: ResNet18 models' grad norm and sharpness in the training process on CIFAR-100, Pets, Car, ImageNet-10, ImageNet



(a) ResNet50 on ImageNet



(b) ResNet101 on ImageNet



(c) ResNet152 on ImageNet

Figure 6: ResNet50, ResNet101, ResNet152's grad norm and sharpness in the training process on ImageNet.

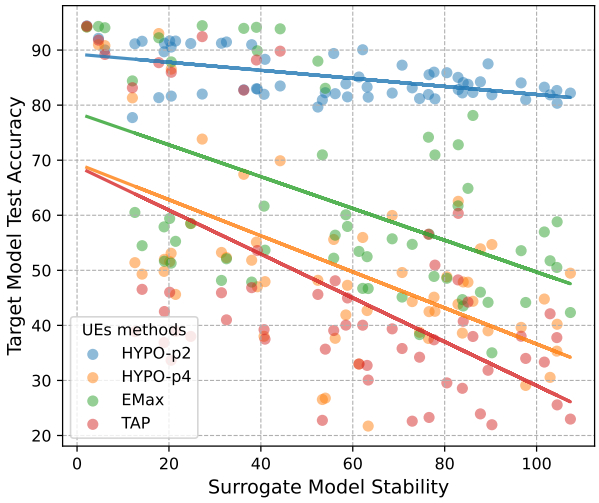


Figure 7: Surrogate model stability and poisoning effectiveness.

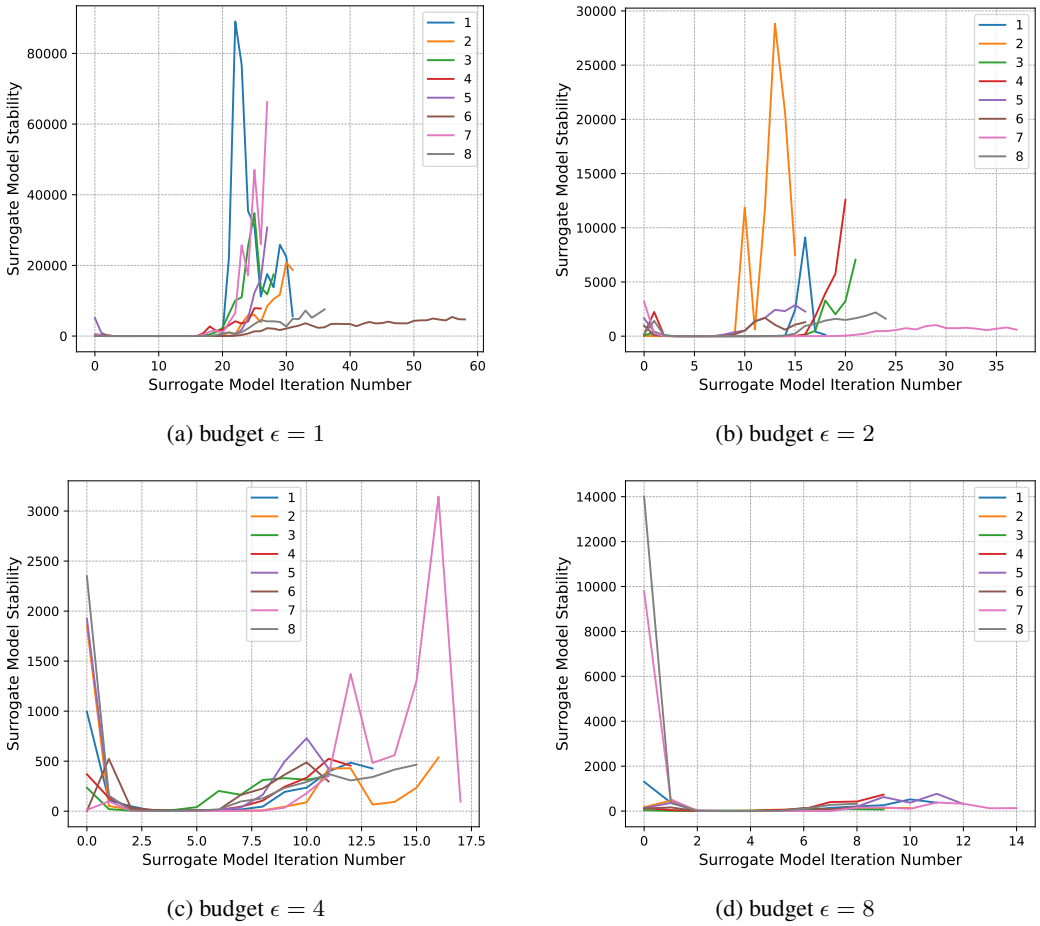
Table 6: Linear regression and correlation analysis between surrogate instability and poisoning effectiveness

Method	Slope	Intercept	Pearson's r	p-value	Std. Error
HYPO-sample	-0.07	89.25	-0.54	< 0.001	0.01
HYPO-class	-0.33	69.38	-0.61	< 0.001	0.06
EMax-class	-0.29	78.56	-0.50	< 0.001	0.07
TAP-class	-0.40	68.89	-0.58	< 0.001	0.07

B.4 Analysis and Enhancement of Bi-level Methods

To investigate the changes in the robustness of the surrogate model in bi-level methods, we test the EM algorithm with perturbation budgets $\epsilon = 1, 2, 4, 8$. To account for randomness, we generate perturbations eight times for each budget. As shown in Figure 8, the x-axis represents the number of iterations, and the y-axis represents the stability metric of the surrogate model at each iteration. Overall, the surrogate model tends to become unstable as the iteration proceeds, and the algorithm stops after reaching its most unstable state. This reveals that bi-level methods implicitly exploit unstable surrogate models to generate poisoning perturbations. From the y-axis range, we observe that a smaller perturbation budget leads to a more unstable surrogate model. For example, at $\epsilon = 1$, the stability value exceeds 80,000, and it gradually decreases as the budget increases. At $\epsilon = 8$, the stability value exceeds 14,000 in the first round, which may be caused by random model initialization rather than the perturbation itself.

Due to the influence of various stochastic factors—including random initialization of model parameters and gradient noise—the stability results of the surrogate model exhibit significant variability. As shown in Figure 8, the stability of the surrogate model is inconsistent across repeated experiments. We propose a heuristic surrogate model selection strategy: among multiple repeated trials, we select the perturbation for which the surrogate model remains consistently unstable throughout the iterative process, i.e., the perturbation that maximizes $\bar{\mathcal{I}}(h)$ in Equation (6). We evaluate the poisoning effectiveness of the selected perturbations generated by the EM algorithm with budgets $\epsilon = 1, 2, 4, 8$. The experimental results are presented in Table 7. Across all budgets, a larger $\bar{\mathcal{I}}(h)$ corresponds to stronger poisoning performance. Notably, under $\epsilon = 1$, the perturbation achieves a poisoning accuracy of 10.67%—a result achieved for the first time. Without selection, the poisoning accuracy of the perturbation is approximately 94%, which is nearly ineffective. Remarkably, we achieve a substantial improvement solely by filtering the surrogate models, without making any modifications



(a) budget $\epsilon = 1$

(b) budget $\epsilon = 2$

(c) budget $\epsilon = 4$

(d) budget $\epsilon = 8$

Figure 8: Variation of the surrogate model's stability during perturbation generation in the bi-level optimization method.

to the internal workings of the EM algorithm. Moreover, as the budget increases, the poisoning performance further improves, reaching 9.61% at $\epsilon = 8$.

B.5 Additional Results on Modern Pretrained Architectures

Sample-wise TAP under Different Adaptation Strategies. We report the detailed results of sample-wise TAP on four pretrained target architectures under full fine-tuning and linear probing in Table 8. The stability-guided checkpoint (epoch 25, with a stability score of 104.47) exactly matches the empirically optimal checkpoint under both adaptation strategies, consistently outperforming the well-trained checkpoint (epoch 60, with a stability score of 20.15).

EM under Linear Probing. Table 9 reports the performance of eight EM-generated perturbation sets with different stability scores. The clean target models achieve a mean accuracy of 89.97%. The perturbation set with the largest stability score of 425.19 achieves the lowest mean accuracy of 77.41%, outperforming the other seven perturbation sets.

EM under Full Fine-tuning. The corresponding results under full fine-tuning are presented in Table 10. Compared with the clean baseline of 97.03%, the perturbation set with the largest stability score reduces the mean accuracy to 15.47%. The substantial degradation across all four architectures further demonstrates the effectiveness of stability-guided run selection in the pretrained fine-tuning scenario.

Table 7: Performance Improvement on Bi-level Methods.

Method	ϵ	$\bar{\mathcal{I}}(h)$	ResNet18	MobileNet	GoogleNet	DenseNet	Avg
EM [19]	1	1398.99	94.15	92.90	94.61	95.14	94.20
		1756.68	94.24	92.86	94.46	94.89	94.11
		2005.78	94.79	92.97	94.50	94.91	94.29
		2583.57	94.41	92.40	94.53	95.25	94.15
		2848.87	40.18	29.85	33.66	95.23	49.73
		4020.47	12.95	15.69	10.00	10.00	12.16
		7047.32	15.57	12.39	11.99	10.11	12.49
		10989.26	**10.36**	**12.37**	**9.99**	**9.96**	**10.67**
	2	377.00	28.97	39.70	22.92	45.56	34.29
		496.05	29.16	19.81	41.56	25.06	28.90
		520.34	34.47	89.35	**10.01**	10.04	34.47
		638.56	38.20	18.91	16.87	26.54	25.13
		763.90	11.42	10.26	14.41	15.17	12.82
		1270.27	10.64	10.79	15.66	11.97	12.27
		1977.59	**10.11**	11.17	10.61	11.33	10.80
		5082.22	**10.11**	**10.29**	10.10	**10.34**	**10.21**
	4	166.62	14.80	13.60	12.50	13.10	13.50
		178.47	14.18	12.30	10.37	13.65	12.62
		212.36	12.35	11.38	13.39	13.74	12.72
		229.94	33.74	9.57	12.43	10.00	16.44
		324.59	11.11	14.14	10.28	15.81	12.84
		329.91	14.21	15.08	12.36	17.40	14.76
		425.19	11.42	10.13	10.03	10.65	10.56
		596.73	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**
	8	44.97	12.21	12.00	11.02	11.02	11.56
		119.19	13.47	10.97	11.70	10.45	11.65
		189.61	10.18	10.00	10.00	16.94	11.78
		219.57	10.00	10.00	10.00	14.75	11.19
		268.36	10.00	10.00	10.09	**9.99**	10.02
		781.71	**9.99**	10.00	9.99	10.08	10.02
		861.33	10.00	10.01	10.02	**9.99**	10.00
		1679.94	10.00	**9.97**	**8.16**	10.32	**9.61**

Table 8: Sample-wise TAP performance on pretrained target architectures under full fine-tuning and linear probing ($\epsilon = 4$).

Adaptation	Selection	Epoch	$\bar{\mathcal{I}}(h)$	ResNet-18	ConvNeXt-T	ViT-B/16	Swin-T	Avg.
Full	Well-trained	60	20.15	72.78	96.26	97.30	93.38	89.93
Full	Stability-guided	25	**104.47**	**52.90**	**80.32**	**95.15**	**75.73**	**76.03**
Full	Optimal	25	**104.47**	**52.90**	**80.32**	**95.15**	**75.73**	**76.03**
Linear	Well-trained	60	20.15	72.07	91.35	94.21	89.54	86.79
Linear	Stability-guided	25	**104.47**	**67.70**	**89.06**	**93.12**	**87.58**	**84.36**
Linear	Optimal	25	**104.47**	**67.70**	**89.06**	**93.12**	**87.58**	**84.36**

B.6 Generalization to Additional Surrogate Architectures

We conduct additional experiments on CIFAR-10 and ImageNet-10 using three surrogate architectures: MobileNetV2, DenseNet121, and GoogleNet. We employ TAP to generate class-wise perturbations and evaluate the resulting poisoned datasets on four target architectures: ResNet-18, MobileNetV2, GoogleNet, and DenseNet121. For each surrogate architecture, we compare three checkpoint selection strategies: (1) *Well-trained*, which selects the final checkpoint of the training trajectory; (2) *Stability-guided*, which selects the checkpoint with the highest stability metric $\bar{\mathcal{I}}(h)$; and (3) *Optimal*, which selects the empirically optimal checkpoint based on poisoning effectiveness. We report the target test accuracy for each architecture and the average across the four target models. Lower accuracy indicates stronger poisoning effectiveness. Importantly, since the scale of the stability metric may vary across architectures and parameterizations, checkpoint selection is performed independently within each surrogate's training trajectory.

Table 9: EM performance on pretrained target architectures under linear probing with $\epsilon = 4$.

ϵ	$\bar{\mathcal{I}}(h)$	ResNet-18	ConvNeXt-T	ViT-B/16	Swin-T	Avg.
Clean	–	78.72	94.13	95.31	91.71	89.97
4	166.62	71.52	93.20	94.49	90.42	87.41
4	173.99	71.10	92.58	93.95	90.32	86.99
4	178.47	68.87	91.63	93.43	89.67	85.90
4	212.36	69.84	92.49	94.25	90.34	86.73
4	227.38	72.27	86.73	93.62	88.50	85.28
4	229.94	72.95	90.84	91.37	90.09	86.31
4	324.59	67.04	90.77	90.43	87.54	83.95
4	**425.19**	**66.70**	**74.35**	**90.81**	**77.77**	**77.41**

Table 10: EM performance on pretrained target architectures under full fine-tuning with $\epsilon = 4$.

ϵ	$\bar{\mathcal{I}}(h)$	ResNet-18	ConvNeXt-T	ViT-B/16	Swin-T	Avg.
Clean	–	94.83	97.99	97.71	97.59	97.03
4	166.62	41.45	67.01	89.68	58.78	64.23
4	173.99	25.79	44.13	86.13	31.71	46.94
4	178.47	37.96	45.35	75.91	42.55	50.44
4	212.36	34.99	60.66	84.92	42.77	55.84
4	227.38	18.11	20.87	73.34	27.87	35.05
4	229.94	19.25	34.72	70.31	28.24	38.13
4	324.59	21.36	67.96	79.00	46.58	53.72
4	**425.19**	**12.51**	**11.38**	**26.49**	**11.52**	**15.47**

Results on CIFAR-10. We first evaluate the three surrogate architectures on CIFAR-10 with an ℓ_∞ perturbation budget of $\epsilon = 2$. Each surrogate is trained for 60 epochs, and the final checkpoint is used as the well-trained baseline. The results are summarized in Table 11. Stability-guided selection consistently improves poisoning effectiveness across all three surrogate architectures and all four target models. Specifically, for MobileNetV2, DenseNet121, and GoogleNet surrogates, the average target accuracy decreases from 63.98% to 41.39%, from 39.39% to 31.74%, and from 82.09% to 69.45%, respectively. Averaged across the three surrogate architectures, stability-guided selection reduces the target accuracy from 61.82% to 47.52%, corresponding to an improvement of 14.30 percentage points over the conventional well-trained checkpoint selection.

Table 11: Generalization of stability-guided surrogate selection to additional surrogate architectures on CIFAR-10 with an ℓ_∞ perturbation budget of $\epsilon = 2$.

Surrogate Architecture	Strategy	Epoch	$\bar{\mathcal{I}}(h)$	RN	MN	GN	DN	Avg.
MobileNetV2	Well-trained	60	78.01	67.54	56.40	69.55	62.44	63.98
	Stability-guided	13	**159.27**	53.68	43.68	34.81	33.41	41.39
	Optimal	33	115.11	**27.32**	**23.68**	**28.29**	**31.15**	**27.61**
DenseNet121	Well-trained	60	6.35	52.91	27.51	44.89	32.26	39.39
	Stability-guided	37	**117.82**	44.08	**22.47**	**28.69**	31.71	31.74
	Optimal	41	91.47	**29.32**	22.55	29.40	**25.87**	**26.79**
GoogleNet	Well-trained	60	5.25	87.20	82.27	85.76	73.13	82.09
	Stability-guided	21	**184.12**	78.25	62.98	77.60	**58.99**	69.45
	Optimal	37	142.39	**66.10**	**53.11**	**65.15**	62.25	**61.65**

Results on ImageNet-10. We further evaluate the three surrogate architectures on ImageNet-10 with an ℓ_∞ perturbation budget of $\epsilon = 4$. Each surrogate is trained for 100 epochs, and the intermediate checkpoints for selection. The results are reported in Table 12. Similar to the CIFAR-10 experiments, stability-guided selection improves the average poisoning effectiveness across all three surrogate architectures. Specifically, it reduces the average target accuracy from 68.05% to 56.80% for DenseNet121, from 68.60% to 67.20% for GoogleNet, and from 69.30% to 68.00% for

MobileNetV2. Overall, the average target accuracy decreases from 68.65% to 64.00%, corresponding to an improvement of 4.65 percentage points over the well-trained checkpoints.

Although the improvements on ImageNet-10 are smaller than those on CIFAR-10, the results show that stability-guided selection remains effective across different surrogate architectures and datasets. We note that the improvement is not uniform across every individual surrogate-target pair; for example, the accuracy on MobileNetV2 as the target model slightly increases from 64.0% to 64.8% when using GoogleNet as the surrogate. Nevertheless, the average target accuracy improves for all three surrogate architectures, providing further evidence of the cross-dataset generalizability of the proposed selection strategy.

Table 12: Generalization of stability-guided surrogate selection to additional surrogate architectures on ImageNet-10 with an ℓ_∞ perturbation budget of $\epsilon = 4$.

Surrogate Architecture	Strategy	Epoch	$\bar{\mathcal{I}}(h)$	RN	MN	GN	DN	Avg.
DenseNet121	Well-trained	100	0.12	66.8	61.0	70.8	73.6	68.05
	Stability-guided	50	**2.21**	**47.8**	47.8	65.2	66.4	56.80
	Optimal	35	1.22	52.0	**46.4**	**64.8**	**54.8**	**54.50**
GoogleNet	Well-trained	100	1.06	69.0	**64.0**	68.2	73.2	68.60
	Stability-guided	35	**7.84**	68.6	64.8	64.6	70.8	67.20
	Optimal	25	3.11	**66.0**	64.4	**59.6**	**67.0**	**64.25**
MobileNetV2	Well-trained	100	3.60	70.2	70.2	63.8	73.0	69.30
	Stability-guided	60	**21.74**	**67.0**	69.4	**63.4**	72.3	68.00
	Optimal	35	16.04	69.0	**55.8**	65.4	**72.2**	**65.60**

B.7 Generalization to Additional UE Generation Objectives

To further examine whether stability-guided surrogate selection generalizes to different perturbation generation objectives, we conduct additional experiments using two UE methods: Unlearnable Clusters (UC), a label-agnostic method, and EMax, which maximizes the classification loss with respect to the true label. These methods employ different optimization objectives from TAP and HYPO, allowing us to evaluate the applicability of our selection strategy beyond the objectives considered in the main experiments. We compare three selection strategies: *Well-trained*, *Stability-guided*, and *Optimal*. The resulting perturbations are evaluated on four target architectures: ResNet-18 (RN), MobileNetV2 (MN), GoogleNet (GN), and DenseNet121 (DN). Lower target test accuracy indicates stronger poisoning effectiveness.

Unlearnable Clusters (UC). UC is a label-agnostic method that extracts representations using a well-trained model, clusters the representations to obtain pseudo-labels, and optimizes perturbations to shift the representations. To isolate the effect of surrogate selection, we fix the pseudo-labels and vary only the surrogate checkpoint used for perturbation generation. Specifically, we train a ResNet-18 surrogate on ImageNet for 90 epochs and generate UC perturbations on Pets with an ℓ_∞ perturbation budget of $\epsilon = 4$.

As reported in Table 13, stability-guided selection identifies the checkpoint at epoch 17, reducing the average target accuracy from 57.23% to 33.99%, an improvement of 23.24 percentage points over the well-trained checkpoint. The selected checkpoint also achieves lower accuracy across all four target architectures. Although the optimal checkpoint achieves a slightly lower average accuracy of 32.15%, the stability-guided checkpoint provides comparable poisoning effectiveness without requiring exhaustive evaluation of all candidate checkpoints.

Table 13: Performance of UC on Pets with an ℓ_∞ perturbation budget of $\epsilon = 4$.

Strategy	Epoch	Stability	RN	MN	GN	DN	Avg.
Well-trained	90	1.28	57.74	50.57	73.97	46.65	57.23
Stability-guided	17	**1.74**	38.74	24.07	**46.45**	26.71	33.99
Optimal	30	1.54	**22.24**	**21.23**	60.18	**24.95**	**32.15**

EMax. We further evaluate EMax on CIFAR-10 with an ℓ_∞ perturbation budget of $\epsilon = 1$. Unlike TAP, which optimizes perturbations toward an incorrect target label, EMax maximizes the classification loss with respect to the true label. We use ResNet-18 as the surrogate architecture and compare checkpoints from its 60-epoch training trajectory.

The results are summarized in Table 14. The stability-guided checkpoint at epoch 32 achieves a stability score of 107.35, substantially higher than that of the well-trained checkpoint at epoch 60 (20.15). Using this checkpoint reduces the average target accuracy from 59.42% to 42.33%, corresponding to an improvement of 17.09 percentage points. The optimal checkpoint at epoch 38 further reduces the average accuracy to 35.02%.

Table 14: Performance of EMax on CIFAR-10 with an ℓ_∞ perturbation budget of $\epsilon = 1$.

Strategy	Epoch	Stability	RN	MN	GN	DN	Avg.
Well-trained	60	20.15	60.83	54.64	63.50	58.69	59.42
Stability-guided	32	**107.35**	43.09	39.10	45.39	41.75	42.33
Optimal	38	90.79	**41.80**	**29.85**	**36.97**	**31.46**	**35.02**

B.8 Online Early Stopping for Bi-level Optimization

In addition to offline checkpoint and run selection, we further explore whether the proposed surrogate stability metric can support online early stopping during bi-level optimization. The motivation is to identify potentially effective perturbations before completing the entire optimization trajectory, thereby avoiding unnecessary optimization iterations. Specifically, at each optimization iteration t, we compute the increment of the surrogate stability metric:

$$\Delta\mathcal{I}_t = \mathcal{I}(h_S^{t+1}) - \mathcal{I}(h_S^t), \tag{40}$$

where $\mathcal{I}(h_S^t)$ denotes the stability metric evaluated at the current surrogate state. We adopt a threshold-based stopping strategy: once $\Delta\mathcal{I}_t$ exceeds a predefined threshold τ, the optimization process is terminated and the current perturbation is returned. This strategy uses only the surrogate stability metric to determine when to stop, without requiring the evaluation of poisoning effectiveness on target models during optimization.

We evaluate this strategy on EM using CIFAR-10 with a perturbation budget of $\epsilon = 1$. To investigate its behavior under different optimization trajectories, we independently repeat the bi-level optimization experiment eight times. For each run, we record the stopping iteration, the corresponding stability increment, and the resulting target test accuracy across four architectures: ResNet-18, MobileNetV2, GoogleNet, and DenseNet121. The results are reported in Table 15. The results reveal an overall association between larger abrupt increases in the stability metric and stronger poisoning effectiveness. In particular, runs with $\Delta\mathcal{I}_t < 8{,}000$ generally exhibit relatively weak poisoning performance, with average target accuracies ranging from 60.91% to 81.19%. As the stability increment becomes larger, the poisoning effectiveness improves substantially. For example, a stability increment of 14,116.17 reduces the average target accuracy to 14.18%. The two runs with even larger increments of 29,860.30 and 66,882.51 achieve average accuracies of 10.51% and 9.79%, respectively.

These observations suggest that abrupt increases in surrogate instability can serve as an informative signal for identifying promising perturbations during bi-level optimization. This experiment is an initial exploration of metric-guided online early stopping. Its effectiveness may depend on the stopping threshold and perturbation budget, and a more principled stopping criterion remains an important direction for future work.

B.9 Ineffectiveness of Robustness Surrogate Models

To further validate Theorem 3.1, we now prove that its contrapositive also holds: the more robust the surrogate model, the less effective the poisoning perturbation. We train ResNet18 on CIFAR-10 via both standard training and adversarial training. The adversarially trained model is obtained using PGD-50 with a perturbation budget of 8 over 200 epochs. We then generate perturbations using the TAP with $\epsilon = 1$ from surrogates sampled at different training stages and evaluate their effectiveness. The results are visualized in Figure 9. The dashed lines represent the standard training:

Table 15: Online early stopping results for EM-based bi-level optimization on CIFAR-10 with $\epsilon = 1$.

Run	Stop Iter.	$\Delta\mathcal{I}_t$	ResNet18	MobileNetV2	GoogleNet	DenseNet121	Avg.
1	25	2,130.92	81.92	72.28	86.90	82.76	80.97
2	26	3,767.38	94.48	78.84	56.40	95.04	81.19
3	27	7,096.41	94.35	32.86	21.46	94.95	60.91
4	26	7,289.93	94.60	80.04	33.96	95.03	75.90
5	30	9,094.15	56.90	33.84	42.70	39.48	43.23
6	28	14,116.17	13.63	16.62	16.16	10.31	14.18
7	25	29,860.30	**10.00**	9.95	9.99	12.10	10.51
8	22	**66,882.51**	10.10	**9.35**	**9.70**	**10.01**	**9.79**

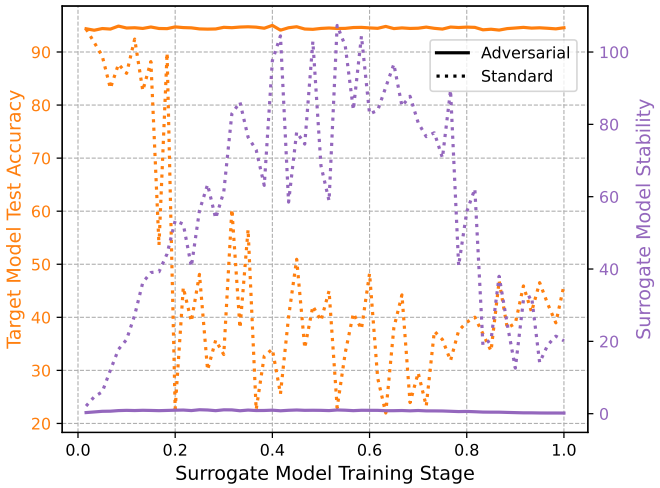


Figure 9: Poisoning using standard and adversarial training surrogate models on CIFAR-10.

the stability metric $\mathcal{I}(h_S^t)$ (purple dashed line) first rises then falls, while the test accuracy (orange dashed line) correspondingly drops then rises, confirming the correlation between robustness and poisoning performance. In contrast, the adversarial training model maintains a near-zero stability metric throughout training (purple solid line), and the resulting perturbations achieve above 90% test accuracies (orange solid line), with virtually no protection effect. This experiment provides clear evidence that the adversarial training enhances the surrogate model's robustness, limiting its utility for generating UEs.

B.10 Empirical Validation of the Stability Metric

Our additional experiments demonstrate that the proposed stability metric is strongly consistent with an empirical measure of model stability. To validate this, we inject Gaussian noise into the training inputs and measure the corresponding change in loss before and after perturbation. As shown in Figure 10, the red curve represents the empirical stability obtained via Gaussian noise, while the blue curve corresponds to the stability metric we propose. Overall, the two curves exhibit a consistent trend: both increase initially, reach a peak in the intermediate stage, and then gradually decline. To objectively evaluate their consistency, we compare the ranking induced by our metric with the empirical ranking. Specifically, we compute the Spearman rank correlation coefficient, which quantifies the agreement between two ranked variables, where values closer to 1 indicate stronger positive correlation. In our experiments, the coefficient reaches 0.814, indicating a strong alignment between the two rankings. These results suggest that the observed loss variations are highly consistent with the trend captured by our proposed metric.

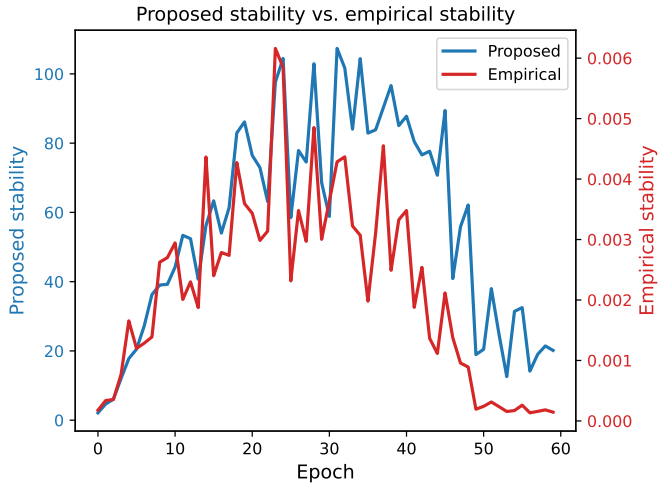


Figure 10: Proposed stability vs. empirical stability.

B.11 Non-linear Degradation under Partial Poisoning

Data protection through poisoning remains a significant challenge, particularly in scenarios involving mixed datasets where only a portion of the data is perturbed, while the remainder stays clean. On CIFAR-10, we generate poisoned data using 5 methods: TAP [10], EM [19], REM [12], LSP [45], and classwise EM [19]. For each method, we create partially poisoned datasets by randomly selecting $(1-\beta)$ samples from the poisoning training set and β samples from the clean training set. We train a ResNet18 on each mixed dataset and record its test accuracy. We also calculate the perturbation-induced poisoning gap of the target model $\Delta(h_T, \boldsymbol{\eta}(\beta))$. Figure 11 shows the test accuracy (solid lines, left axis) and the corresponding gap values (dashed lines, right axis) against the training dataset proportion β. As β increases from 0 to 20%, the test accuracy of all five methods jumps from about 20% to over 80%, while the $\Delta(h_T, \boldsymbol{\eta}(\beta))$ values drop sharply. Beyond β=20%, the accuracy improves slowly until it plateaus, and $\Delta(h_T, \boldsymbol{\eta}(\beta))$ correspondingly decreases to its minimum.

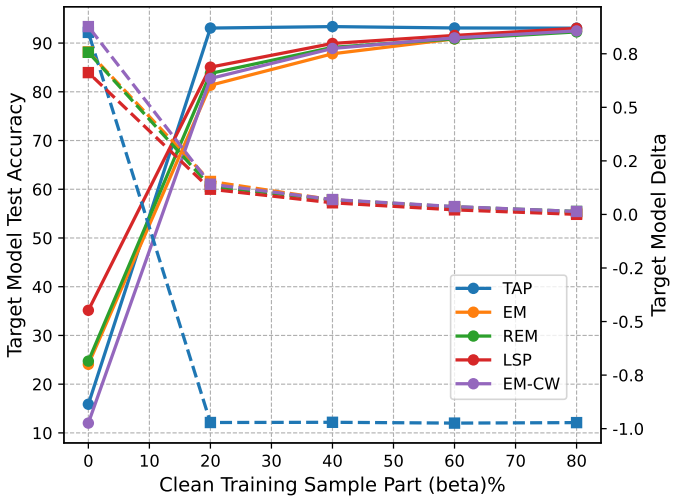


Figure 11: Impact of training dataset proportion β on poisoning effectiveness.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state that well-trained surrogates are suboptimal for generating unlearnable examples, propose stability-based surrogate selection, and provide theoretical analysis (Theorem 3.1) as well as extensive experiments (Section 1, Section 4) that support all claims.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section B.11 discusses partial data protection and the non-linear degradation of poisoning effectiveness.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Theorem 3.1 are stated with clear assumptions. Complete proofs are provided in Appendix A, including all necessary definitions (loss, generalization gap, poisoning gap, deviation gap) and derivations.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 4 and its subsections provide detailed descriptions of datasets (CIFAR-10/100, Pets, Car, ImageNet), surrogate/target models (ResNet18, MobileNetV2, GoogleNet, DenseNet121), training hyperparameters (SGD, learning rate schedule, epochs), perturbation budgets, and evaluation metrics. Section 4.6 adds computational details.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The paper provides anonymized code in the supplemental material, including scripts to generate perturbations (EMax, HYPO, TAP, EM, SEP) and to train target models. All datasets used are publicly available (CIFAR-10/100, Pets, Car, ImageNet) with documented download instructions. The README file will contain exact commands and environment setup.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 (Experimental Setup) specifies the optimizer (SGD with momentum 0.9), learning rate (CosineAnnealingLR from 0.1), training epochs (60/100/300), data splits (e.g., Pets 4:1 train/test, Car 5:5), perturbation norms (ℓ_∞), and the metric (test accuracy on clean data).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: The paper reports single run accuracies for each surrogate epoch and target model (e.g., Tables 1). While multiple target architectures are evaluated, standard deviations or confidence intervals are not provided due to the high computational cost of repeated runs (each run requires training a surrogate and multiple target models). The trends and relative improvements are consistent and statistically clear.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 4.6 reports runtime per checkpoint (30 seconds for ResNet18 on CIFAR-10) and estimates that the full stability calculation is affordable.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research uses public benchmark datasets (CIFAR, Pets, Car, ImageNet) and does not involve human subjects or sensitive data. There is no privacy violation, discrimination, or harmful deployment. The work conforms to the Code of Ethics.

   Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The introduction (Section 1) discusses the positive impact of protecting data privacy and intellectual property against unauthorized model training.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release new datasets or pre-trained models. It only uses existing public datasets and generates perturbations as a defense mechanism. No high-risk assets are introduced.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the original papers for all datasets and models. The accompanying code repository will include a LICENSE file and properly attribute each asset (e.g., CIFAR-10/100 under CC BY-SA 4.0, ImageNet under custom non-commercial license)

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not introduce new datasets, code packages, or models.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human subjects research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [No]

    Justification: LLMs are not used as a core component of the methodology.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.