Selection-Based Vulnerabilities: Clean-Label Backdoor Attacks in Active Learning

Yuhan Zhi[1], **Longtian Wang**[1], **Xiaofei Xie**[2], **Chao Shen**[1], **Qiang Hu**[3], **Xiaohong Guan**[1]
[1]Xi'an Jiaotong University
[2]Singapore Management University
[3]Tianjin University
zyh1123@stu.xjtu.edu.cn

Abstract

Active learning (AL), which serves as the representative label-efficient learning paradigm, has been widely applied in resource-constrained scenarios. The achievement of AL is attributed to acquisition functions, which are designed for identifying the most important data to label. Despite this success, one question remains unanswered: *is AL safe?* In this work, we introduce *ALA*, a practical and the first framework to utilize the acquisition function as the poisoning attack surface to reveal the weakness of active learning. Specifically, *ALA* optimizes imperceptibly poisoned inputs to exhibit high uncertainty scores, increasing their probability of being selected by acquisition functions. To evaluate *ALA*, we conduct extensive experiments across three datasets, three acquisition functions, and two types of clean-label backdoor triggers. Results show that our attack can achieve high success rates (up to 94%) even under low poisoning budgets (0.5%–1.0%) while preserving model utility and remaining undetectable to human annotators. Our findings remind active learning users: acquisition functions can be easily exploited, and active learning should be deployed with caution in trusted data scenarios.

Introduction

Active learning (AL) is a learning paradigm designed to reduce annotation costs by iteratively selecting and labeling the most informative samples from an unlabeled data pool as training data (Settles 2009; Munjal et al. 2022; Ren et al. 2021). Acquisition function, which identifies the data to be labeled, is the core of AL. Multiple acquisition functions have been proposed (Tong and Koller 2001; Joshi, Porikli, and Papanikolopoulos 2009; Li and Guo 2013; Beluch et al. 2018), where uncertainty based ones are the most representative, such as prediction confidence-based function and entropy score-based function (Wang and Shang 2014) . With the increasingly proposed effective acquisition functions, AL has been more and more useful, especially when labeling is expensive, enabling the model to achieve comparable accuracy with fewer labeled samples.

In real-world applications, AL is frequently deployed in two scenarios: 1) training deep learning (DL) models from scratch with limited data labeling budgets (Sener and Savarese 2018; Wang and Shang 2014), and 2) continual learning for quickly new data distribution adaptation (Kim, Feldt, and Yoo 2019; Hu et al. 2024). For the latter one, when a model has been deployed in the wild, it is common that the new unseen data samples follow different data distributions, i.e., out-of-distribution (OOD) data, from the original training data. In this situation, AL is used to select fine-tuning data for model enhancement. Such settings arise in medical diagnosis, autonomous systems, and data-centric AI applications (Santos and Carvalho 2024; Lu et al. 2024), where models must incrementally learn new concepts or domains from previously unseen data.

Even though AL has achieved great success in multiple domains and scenarios, the safety of AL has rarely been discussed. As the heart of AL, the acquisition function determines which unlabeled samples should be annotated. In this manner, one assumption is – the potential selection bias in acquisition function has the potential to be exploited to inject poisoned samples into the training set, thus, serves as the attack surface.

To this end, in this paper, we introduce *ALA*, the first framework to utilize acquisition functions to reveal the weakness of active learning from the poisoning attack perspective. We consider a white-box threat model where attackers can access the model parameters. *ALA* strategically injects clean-label poisoned samples and exploits the construction design of acquisition functions (selecting data samples with high uncertainty) to increase their probability of being included in the training set. Specifically, *ALA* first selects samples that are near the decision boundaries (quantified by the high uncertainty scores) as seed data. Then, *ALA* applies poisoning attack methods with a selection-aware optimization algorithm to generate poisoned samples while further increase the uncertainty scores of these generated samples. In this way, the poisoned samples are more likely to be selected by acquisition functions and therefore, injected into the training set.

To evaluate *ALA*, we conduct comprehensive experiments on three datasets (Fashion-MNIST, CIFAR-10, and SVHN) with two clean-label backdoor triggers (CL (Turner, Tsipras, and Madry 2019) and SIG (Barni, Kallas, and Tondi 2019)) and three acquisition functions (Entropy, Margin, and Least Confidence (Wang and Shang 2014)). Our results show that poisoned samples injected by *ALA* can lead to attack success rates of up to 94% on the AL-trained model under uncertainty-based acquisition functions, even with a low poisoning budget, achieving a 43% improvement over the Random selection baseline, and highlighting the potential vulnerability of AL in practical scenarios. Moreover, we find that in AL scenarios, the SIG trigger is substantially more effective and robust than CL for clean-label backdoor injection, consistently achieving higher ASRs across datasets and acquisition functions.

Our Contributions.

- We identify the acquisition function in AL as a new and realistic attack surface for clean-label backdoor injection.
- We propose *ALA*, the first framework to attack active learning with a selection-aware optimization strategy that aligns poisoned inputs with the model's selection preferences to enhance attack success.
- We conduct extensive experiments across three datasets, three acquisition functions, and two clean-label attack types, demonstrating strong attack effectiveness under multiple common AL configurations, revealing a potential real-world vulnerability of active learning.

Our results raise important concerns about the security of AL pipelines, and highlight the need to incorporate selection-aware threat models into the design of robust active learning systems.

Background

Active Learning

Active learning is a learning paradigm that aims to reduce annotation costs by selectively querying labels for only the most informative samples (Settles 2009; Munjal et al. 2022). Rather than training on a large fully labeled dataset, an AL algorithm maintains a small labeled set $\mathcal{L}$ and a large unlabeled pool $\mathcal{U}$, and iteratively selects samples from $\mathcal{U}$ to be labeled by an oracle (e.g., a human annotator) (Ren et al. 2021). The newly labeled samples are added to $\mathcal{L}$, and the model is retrained. This loop continues until a predefined labeling budget is exhausted or model performance converges.

The AL framework typically includes three components: (1) a model or learner, (2) an acquisition function for determining which samples to label next, and (3) a labeling oracle. This iterative loop has been widely applied in scenarios such as medical imaging, autonomous driving, and real-world systems where labeling is expensive and continuous adaptation is necessary.

Acquisition Functions in Active Learning

The effectiveness of AL heavily depends on the design of its acquisition function, which determines which unlabeled samples are most beneficial to label at each epoch. Among various approaches, uncertainty-based acquisition functions (Tong and Koller 2001; Joshi, Porikli, and Papanikolopoulos 2009; Li and Guo 2013; Beluch et al. 2018) are the most widely adopted due to their simplicity and effectiveness. These methods prioritize samples on which the current model is most uncertain.

Common uncertainty-based acquisition functions include:

- **Entropy**: Selects samples with the highest entropy in the predicted class distribution. For a sample x, let $p(y \mid x)$ be the model's predicted probability distribution over K classes. The entropy (Shannon 1948) is computed as:

$$\mathcal{H}(x) = -\sum_{k=1}^{K} p(y_k \mid x) \log p(y_k \mid x).$$

  Samples with higher entropy indicate greater prediction uncertainty.
- **Margin**: Selects samples with the smallest margin between the top two class probabilities (Roy and McCallum 2001).
- **Least Confidence**: Selects samples with the lowest maximum predicted probability (Wang and Shang 2014).

Other acquisition functions target alternative objectives beyond uncertainty. Diversity-based methods aim to select samples that are both representative of the input distribution and mutually diverse, in order to cover different regions of the feature space (Sener and Savarese 2017; Bilgic and Getoor 2009; Gal, Islam, and Ghahramani 2017). Hybrid methods combine multiple selection signals—typically blending uncertainty with diversity—to improve robustness and sample efficiency (Ash et al. 2019; Shui et al. 2020; Yin et al. 2017).

In this work, we focus on **uncertainty-based acquisition functions**, as they are generally effective in practice and are commonly used in AL (Li et al. 2022, 2024).

Backdoor Attacks and Clean-Label Threats

Backdoor attacks (Gu, Dolan-Gavitt, and Garg 2017) are a class of data poisoning attacks that aim to implant hidden malicious behavior into machine learning models. Typically, the attacker injects a small number of specially crafted training samples—known as poisoned samples—that contain a specific trigger pattern. These samples are labeled as the attacker's desired target class. At inference time, any input containing the same trigger will be misclassified into the target class, while the model maintains high accuracy on clean inputs.

Traditional backdoor attacks often assume that the attacker can manipulate both the training data and the associated labels (Liu et al. 2018; Zhong et al. 2020; Ji, Zhang, and Wang 2017). However, this assumption may not hold in many real-world scenarios, especially when labels are provided by human annotators or trusted pipelines (Turner, Tsipras, and Madry 2019). To address this limitation, clean-label backdoor attacks have emerged as a more realistic and stealthy threat model by preserving the ground-truth labels of poisoned samples (Saha, Subramanya, and Pirsiavash 2020).

Clean-label backdoor attacks exploit the mismatch between human and model perception by embedding imperceptible triggers into inputs without altering their ground-truth labels. While human annotators assign correct labels based on visual semantics, the model may learn to associate subtle patterns—such as high-frequency or localized signals—with a specific target class. As a result, inputs containing the same trigger at test time can be misclassified into the target class, even if they are semantically unrelated. Representative techniques include:

- **CL**: Adds localized, low-visibility triggers (e.g., small patches) without altering the sample's semantics (Turner, Tsipras, and Madry 2019).
- **SIG**: Applies imperceptible, frequency-domain perturbations as triggers (Barni, Kallas, and Tondi 2019).

These techniques are particularly effective in settings where the labeling process is external and cannot be manipulated. In our work, we adopt and adapt these clean-label backdoor methods to the context of active learning, where the attacker cannot flip labels and must remain undetected by human annotators.

Methodology

Problem setting

We consider a realistic and common deployment scenario: a deployed model has already been well trained on an initial labeled dataset and achieves strong performance. Given newly collected unlabeled data that follow a different distribution (OOD data) from the original training data, we employ AL to conduct distribution adaptation for the pre-trained model under a specific labeling budget.

We focus on this setting for both technical and practical reasons. From a technical standpoint, poisoning the initial training set is often infeasible, as it typically requires insider access to the original data pipeline. In contrast, newly arriving OOD data is more accessible and provides a natural point of interaction for external attackers. In practice, attacking a poorly trained model is of limited value, as its predictions are already unreliable. In contrast, attacking a deployed, well-performed model makes real-world impacts and therefore should be more carefully considered.

In our setting, the AL model follows a standard loop: at each epoch, the model selects a batch of unlabeled OOD samples using a predefined acquisition function (e.g., Entropy, Margin, Least Confidence), obtains their labels through human annotation (i.e., oracle in our experiments), and retrains on the updated labeled set. This process repeats until the labeling budget is exhausted or target performance is reached.

Our attack aims to inject a clean-label backdoor during this OOD adaptation process. Specifically, we poison a small number of OOD samples by embedding an imperceptible trigger, such that the final model misclassifies any input containing the trigger into a specific target class, while maintaining high accuracy on clean inputs from both ID and OOD distributions.

To make the threat model explicit, we assume the following capabilities and constraints for the attacker:

- **Model access**: The attacker has access to the current AL model before the selection step on incoming OOD data. This includes the architecture and parameters of the model.
- **Data injection**: The attacker can access and modify a subset of the newly arriving OOD samples *before they are injected into the unlabeled pool*. However, the attacker has **no access** to the original ID training data.

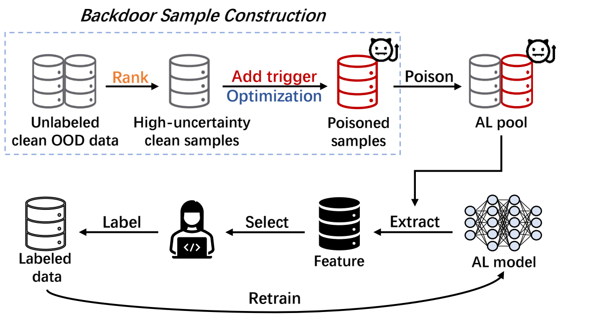

Figure 1: Overview of *ALA*.

- **Label integrity**: The attacker cannot alter the ground-truth labels, nor interfere with the annotation process, which is assumed to be performed by human annotators.

These assumptions reflect realistic attack surfaces in deployed AL systems. Meanwhile, this setting introduces unique challenges: the attacker must craft poisoned samples that satisfy the clean-label constraint, and yet ensure that they are selected by the AL agent despite being rare in a large, unfamiliar OOD pool.

Overview of *ALA*

Figure 1 illustrates the overall workflow of *ALA*. It first uses the current-epoch AL model (or pre-trained model) to rank the incoming OOD data based on their uncertainty scores. We then identify high-uncertainty samples from the attack target class. After that, according to a predefined poisoning ratio, *ALA* embeds an invisible trigger to the top uncertain samples. Since adding the trigger may change the prediction confidence of the model on the data, in turn, affects the uncertainty scores, we design a selection-aware optimization algorithm to increase the attack success rate and the uncertainty of the poisoned data.

The optimized poisoned samples are then injected into the candidate pool under acquisition. The AL agent selects samples based on their extracted features, obtains labels from the oracle, and retrains the AL model. This process is repeated throughout the AL loop, allowing poisoned samples to be gradually incorporated and enabling the model to learn the backdoor behavior over time.

Backdoor Sample Construction

The backdoor sample construction process is the core of *ALA*, which contains two key steps: target-class candidate selection and selection-aware optimization. Algorithm 1 shows the detailed process.

Target-class candidate selection. Clean-label backdoor attacks target a specific class y_t, and poisoning is only applied to samples belonging to this class. Since the incoming OOD samples are unlabeled, the attacker must manually identify which samples belong to y_t. Given this setting, we aim to minimize labeling cost by first ranking the unlabeled OOD pool $\mathcal{U}$ using the current AL model's uncertainty score, in this work we use entropy, though any uncertainty-based metric could be used (Line 2), and then sequentially

```
Algorithm 1: Backdoor Sample Construction
Require: Unlabeled pool $\mathcal{U}$, target class $y_t$, poisoning ratio
    $\rho$, AL model $M$, trigger $T$, GA iterations $G$, population
    size $P$, Entropy calculation $H$
Ensure: Poisoned sample set $\mathcal{P}$
 1: $\mathcal{P} \leftarrow \emptyset, \mathcal{C} \leftarrow \emptyset$                ▷ Initialize
 2: Rank $\mathcal{U}$ by entropy $H(M(x))$ in descending order
 3: for top-ranked $x \in \mathcal{U}$ do
 4:     $y \leftarrow$ QueryLabel$(x)$             ▷ Human annotation
 5:     if $y = y_t$ then
 6:         $\mathcal{C} \leftarrow \mathcal{C} \cup \{x\}$
 7:         if $|\mathcal{C}| \geq \rho \cdot |\mathcal{U}|$ then break
 8:         end if
 9:     end if
10: end for
11: for $x \in \mathcal{C}$ do
12:     $x_{\text{best}} \leftarrow x$, $H_{\text{max}} \leftarrow -\infty$
13:     for $g = 1$ to $G$ do
14:         Generate mutants $\{m_1, \ldots, m_P\}$ from $x_{\text{best}}$ via
    augmentation
15:         $\mathcal{P}_{\text{cand}} \leftarrow \{\text{ApplyTrigger}(m_i, T)\}_{i=1}^{P}$
16:         $index \leftarrow \arg\max_{p \in \mathcal{P}_{\text{cand}}} H(M(p))$
17:         $p^* \leftarrow m_{index}$
18:         if $H(M(p^*)) > H_{\text{max}}$ then
19:             $x_{\text{best}} \leftarrow p^*$, $H_{\text{max}} \leftarrow H(M(p^*))$
20:         end if
21:         $x \leftarrow$ RemoveTrigger$(p^*, T)$      ▷ Clean seed for
    next iter
22:     end for
23:     $\mathcal{P} \leftarrow \mathcal{P} \cup \{x_{\text{best}}\}$
24: end for
25: return $\mathcal{P}$
```

querying the ground-truth labels of the top-ranked samples, which in practice would require human annotation (Line 3-6). The process stops once $\rho \cdot |\mathcal{U}|$ samples from class y_t are found (Line 7), where ρ is the predefined poisoning ratio. This strategy ensures that the selected samples are both label-efficient and more likely to be chosen by the AL agent.

Selection-aware optimization. To perform the poisoning attack, a trigger needs to be embedded in the data. However, adding a trigger can affect the uncertainty of a sample, potentially reducing the chance that it will be queried by the uncertainty-based acquisition function. To mitigate this, we design a selection-aware optimization procedure based on a genetic algorithm (GA) to increase each poisoned sample's uncertainty score (i.e., entropy). Specifically, given a clean sample as a seed, *ALA* first mutates it and generates multiple mutants (Line 14). Then, the poisoning attack is used to inject triggers into these mutants (Line 15). After that, we calculate the uncertainty score of each poisoned sample and keep the one with the highest uncertainty (Line 16-19). After removing its trigger, this sample will be used as the seed in the next iteration (Line 20). Finally, the poisoned sample with the highest entropy in the final population is selected as the optimized poisoned sample (Line 22).

We adopt existing clean-label attack methods—CL and SIG—for trigger construction. To remain stealthy, the trigger is designed to be visually imperceptible and does not alter the semantic content of the image. The ground-truth label remains unchanged, which satisfies the clean-label constraint and avoids suspicion during the annotation process.

Experiments

Datasets and Models

We conduct experiments on three widely used benchmark datasets: Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017), a grayscale image dataset of clothing items with 10 classes; CIFAR-10 (Krizhevsky, Hinton et al. 2009), a 10-class natural image dataset with low-resolution color images; and SVHN (Netzer et al. 2011), a real-world digit classification dataset derived from street view house numbers. These datasets cover a range of visual domains and classification difficulties.

For each dataset, we use a commonly adopted architecture with sufficient capacity for the corresponding task. Specifically, we use LeNet-5 (LeCun et al. 1998) for Fashion-MNIST, ResNet-18 (He et al. 2016) for CIFAR-10, and MobileNetV2 (Sandler et al. 2018) for SVHN.

Setup

AL Setting. Following previous work (Hu et al. 2024), we simulate an iterative AL loop with a total labeling budget of 10% of the unlabeled data. At each epoch, the model selects 1% of samples from the unlabeled pool using a predefined acquisition function. We evaluate three widely used uncertainty-based acquisition functions: Entropy, Margin, and Least Confidence. Following standard AL practice, the model is incrementally trained at each epoch using the updated labeled set, continuing from the model parameters obtained in the previous epoch.

Data Distribution. To simulate continuous learning in deployment time on unfamiliar data, we construct an unlabeled OOD pool by applying a randomly sampled corruption from CIFAR-10-C (Hendrycks and Dietterich 2019) to each test image, with a randomly chosen severity level. CIFAR-10-C provides 19 types of common visual corruptions that can significantly degrade model performance, each with five severity levels. A corrupted sample is added to the OOD pool if it is misclassified by the pretrained AL model. Otherwise, the corruption process is repeated until a misclassified variant is found. This procedure ensures that the OOD pool consists of visually plausible samples that the model fails to recognize, reflecting realistic data drift during post-deployment learning. The ID data used for pretraining the AL model corresponds to the original training split of each dataset.

Backdoor Attack Configuration. The current version of *ALA* integrates two representative clean-label backdoor attack methods. *ALA* is extensible, and any new attack methods can be easily employed in it.

- **Clean-Label (CL)** (Turner, Tsipras, and Madry 2019): Adversarial samples are generated via projected gradient descent (PGD) attack. The perturbation is constrained under an ℓ_∞ bound of $\epsilon = 32$.

- **SIG** (Barni, Kallas, and Tondi 2019): An imperceptible sinusoidal signal trigger is overlaid onto each poisoned sample. The signal is defined by a spatial frequency $f = 6$ and amplitude $\delta = 50$, resulting in a smooth, high-frequency pattern that is visually indistinguishable to humans but learnable by the model.

All poisoned samples are assigned their correct ground-truth labels and visually resemble benign data. We evaluate two poisoning ratios: 0.5% and 1.0% of the OOD pool. Poisoning is applied class-wise for each of the 10 classes, and results are averaged over classes.

Selection-Aware Optimization. To increase the likelihood that poisoned samples are selected by the AL agent, we apply a GA to optimize their uncertainty (measured via entropy). For each poisoned sample, the GA is run independently with a population size of 100, tournament size of 5, and mutation rate of 0.5. Crossover is performed by randomly inheriting from one parent. The mutation operations include: (1) pixel-level noise (Gaussian, salt-and-pepper, multiplicative), (2) blurring (Gaussian, uniform, median, bilateral), and (3) global transformations (brightness and contrast adjustments). We evaluate the effect of optimization at multiple checkpoints: 0, 5, 10, and 15 iterations.

Experiment Configurations. Each active learning process is repeated three times with different random seeds, and we report the average value of all evaluation metrics. All experiments are conducted on a server equipped with an Intel(R) Xeon(R) Gold 6226R CPU (64 cores, 2.90GHz) and eight NVIDIA RTX 3090 GPUs.

Evaluation Metrics

We evaluate our attack effectiveness and its impact on model performance using the following metrics:

- **Poisoned Selection Rate (R_{select}).** We report the percentage of poisoned samples that are selected by the AL agent during the entire active learning process. A higher R_{select} under uncertainty-based acquisition functions—compared to Random—indicates a more effective attack This metric directly quantifies how well the attack succeeds in manipulating the acquisition function to prioritize poisoned inputs.
- **Attack Success Rate (ASR).** ASR measures the fraction of test-time inputs embedded with the backdoor trigger that are misclassified into the attacker-specified target class. A higher ASR indicates a more effective backdoor attack.
- **ID Accuracy (Acc_{ID}).** To assess whether the backdoor affects the model's performance on previously learned data, we report the classification accuracy on clean, ID test data. High Acc_{ID} indicates that the attack preserves performance on the original distribution.
- **OOD Accuracy (Acc_{OOD}).** We also measure the model's accuracy on clean (i.e., non-poisoned) OOD test data to assess whether the attack compromises generalization to newly learned classes. Maintaining high Acc_{OOD} is important for stealthiness, as a noticeable drop may expose the presence of poisoned data.

Table 1: Poisoned Selection Rate (%) at the First and Last Training Epochs on CIFAR-10 under the SIG Attack (0.5% Poisoning), across different acquisition functions and optimization iterations. Iter refers to optimization iteration.

Acquisition	Epoch	Iter 0	Iter 5	Iter 10	Iter 15
Random	0	1.6	0.6	1.4	0.8
	9	7.6	10.0	9.4	8.6
Entropy	0	23.8	75.4	84.6	89.4
	9	29.1	75.9	84.8	89.4
Margin	0	4.6	18.0	26.2	32.4
	9	14.6	24.3	30.1	35.5
Least Confidence	0	23.8	78.8	89.4	95.8
	9	28.5	79.7	89.5	95.8

Results

Effectiveness of Selection-Aware Optimization. Table 1 summarizes the Poisoned Selection Rate (R_{select}) at the first and last training epochs across different acquisition functions on the CIFAR-10 dataset using the SIG trigger with a poisoning ratio of 0.5%. The results of other settings can be found in the Appendix.

Comparing the results of iteration 0 (no optimization) with iterations 5, 10, and 15. We observe that *ALA* substantially increases R_{select} when using uncertainty-based acquisition functions (e.g., from 29.1% to 89.4% under Entropy-based acquisition), while no significant change is observed under Random selection. By computing the correlation between R_{select} and ASR scores (shown in the Appendix) across training epochs for iterations 5, 10, and 15, we find a positive correlation of 0.698 ($p < 0.001$), indicating a strong relationship between R_{select} and attack success.

Considering the R_{select} under different training epochs, the results show that the difference between these two epochs is negligible. For instance, across all uncertainty-based acquisition functions and optimization settings, the average change in R_{select} between epoch 0 and epoch 9 is less than 2.9%. We attribute this to the shift in model decision boundaries after the first epoch of training, which may reduce the uncertainty of poisoned samples that were originally optimized based on the previous model. These findings suggest that, if the attacker can only manipulate samples before injecting them into the AL pool, it is critical to ensure that as many poisoned samples as possible are selected in the very first epoch.

Figure 2 depicts the ASR scores achieved by *ALA* over different iterations. The dashed line represents a control experiment in which all poisoned samples are manually forced to be selected in the first active learning epoch. In this setting, the poisoned samples participate in training from the beginning of the loop, providing an estimated upper bound for ASR under full exposure. The estimated ASR values are also averaged over attacks targeting all classes. Table 2 reports the final ASR achieved by each acquisition function under different optimization iterations, including the estimated upper bound for reference.

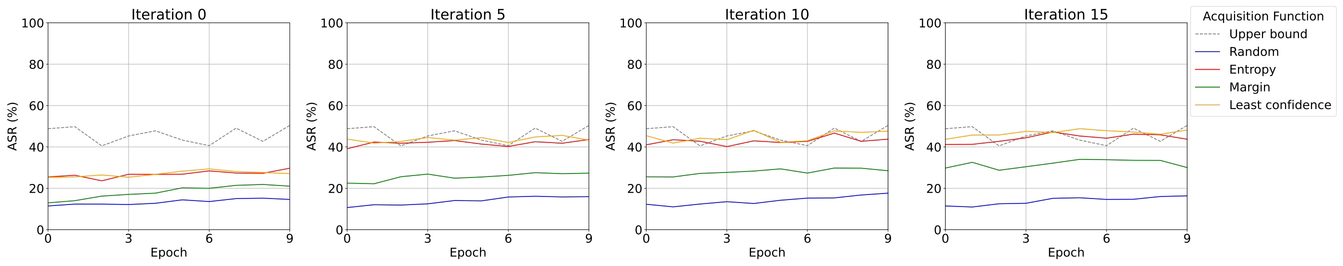

Figure 2: ASR (%) over training epochs on CIFAR-10 under the SIG trigger with 0.5% poisoning. Colored lines represent different acquisition functions, while the dashed line denotes the estimated upper bound achieved when all poisoned samples are selected in the first AL epoch.

Table 2: Final ASR (%) on CIFAR-10 under the SIG attack with 0.5% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where all poisoned samples are selected in the first AL epoch. Iter refers to optimization iteration.

ASR (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	46.0	46.0	46.0	46.0
Random	14.6	16.0	17.7	16.4
Entropy	29.7	43.6	43.8	43.7
Margin	21.0	27.3	28.5	30.0
Least Confidence	27.1	43.3	47.7	48.2

We observe that *ALA* significantly improves ASR under Entropy and Least Confidence selection, approaching or even surpassing the estimated upper bound. For example, *ALA* achieves 47.7% and 48.2% ASRs on Least Confidence with 10 and 15 optimization iterations, respectively—exceeding the estimated upper bound of 46.0%. Margin also benefits from optimization. Though its gains are more modest, it still maintains a clear advantage over Random. As expected, Random selection shows no substantial improvement with additional optimization.

Notably, the differences in ASR across different optimization iterations are relatively small, which aligns with our observations from Table 1: when R_{select} ceases to increase significantly, the ASR also plateaus. This suggests that five iterations of optimization are sufficient to achieve stable and effective results. Nevertheless, although additional optimization iterations yield diminishing returns, we still observe slight ASR improvements under the Margin and Least Confidence acquisition functions, as their corresponding R_{select} values continue to grow slightly.

We further report both Acc_{ID} and Acc_{OOD} results in the Appendix. To investigate whether *ALA* impacts the model's performance on the original ID data, we plot the Acc_{ID} curve over training epochs under the poisoned setting. The results show that Acc_{ID} remains stable throughout training, with no noticeable degradation and even slight improvements in some cases. To assess whether the attack degrades generalization on clean OOD data, we conduct a control experiment where the AL model is trained using the original,

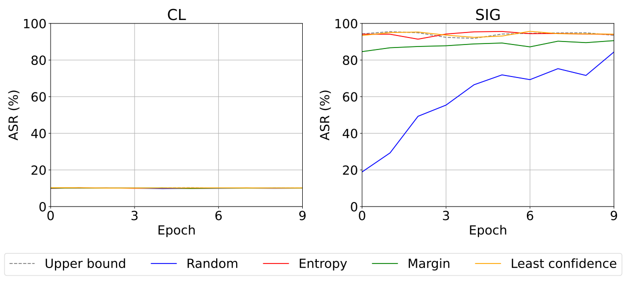

Figure 3: ASR (%) over training epochs on SVHN under the CL and SIG trigger with 1.0% poisoning and 15 optimization iterations. Colored lines represent different acquisition functions, while the dashed line denotes the estimated upper bound achieved when all poisoned samples are selected in the first AL epoch.

unpoisoned OOD samples. The resulting accuracy curve is compared against the poisoned training curve. We observe no significant difference between the two. These findings suggest that our clean-label attack does not introduce abnormal accuracy drops or fluctuations in either Acc_{ID} or Acc_{OOD}, making it difficult to detect through standard performance monitoring.

Comparison Across Backdoor Attack Methods. We compare two clean-label backdoor attack methods: CL and SIG. While SIG consistently achieves reasonably strong ASR across all datasets, CL proves ineffective as an attack method under the active learning setting.

Figure 3 illustrates the ASR over training epochs on the CIFAR-10 dataset for both CL and SIG triggers, with a 1% poisoning ratio and 15 optimization iterations. For reference, we also include the estimated upper bound—obtained by assuming all poisoned samples are selected in the first AL epoch. The results reveal a stark contrast between the two methods. Under CL, the ASR remains consistently around 10% throughout training. Even under the estimated upper bound condition, the ASR fails to exceed this level, indicating that CL cannot induce misclassification beyond the poisoned sample's original class. In contrast, SIG achieves significantly higher ASRs, with attacks optimized by *ALA* approaching the estimated upper bound, which itself is close

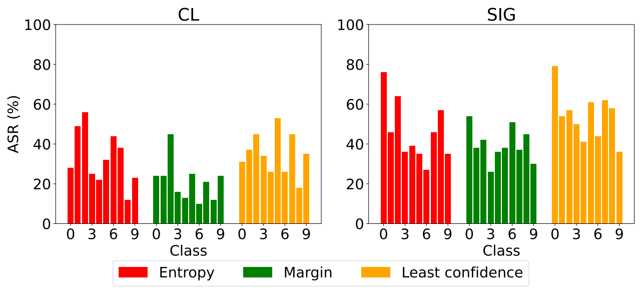

Figure 4: ASR (%) of epoch 9 over different classes on CIFAR-10 under the CL and SIG trigger with 1.0% poisoning and 15 optimization iterations. Colored lines represent different acquisition functions, while the dashed line denotes the estimated upper bound achieved when all poisoned samples are selected in the first AL epoch.

to 100%. More results are put in the Appendix.

These findings demonstrate that in active learning scenarios, SIG is substantially more effective and robust than CL for clean-label backdoor injection—likely due to its stronger and more learnable trigger design.

ASR Difference Across Target Classes. Figure 4 presents the class-wise ASR on training epoch 9 of CIFAR-10 under the CL and SIG triggers with a 1.0% poisoning ratio. Each bar corresponds to the ASR for a specific target class when that class is selected as the attack target.

We observe class-dependent differences in attack difficulty across methods. For example, under all three uncertainty-based acquisition functions, CL consistently yields high ASR on class label 2, while SIG achieves its highest ASR on class label 0, showing consistent behavior within each method. Interestingly, SIG performs well on class label 8, where CL fails to achieve high ASR, further highlighting the differing effectiveness of the two triggers across target classes.

The results reveal a clear contrast in attack generality between the two trigger types. Compared to CL, the SIG trigger achieves consistently higher ASRs across nearly all classes and acquisition functions, demonstrating stronger and more stable attack performance. In particular, under the SIG trigger, most target classes achieve ASRs above 30%, with more than half exceeding 50%, and some reaching over 70%. In contrast, under the CL trigger, ASRs for more than half of the classes fail to surpass 30%, indicating that SIG generalizes better across different semantic categories. More results can be found in the Appendix.

Discussion

Broader Implications of Selection-Aware Attacks. Our work highlights a new and underexplored attack surface: the acquisition function itself. While our study focuses on AL, the vulnerability is not confined to AL frameworks. Any machine learning pipeline that leverages acquisition function—such as test case selection, data valuation, repair, and retraining strategies—could be similarly exploited. This calls for a re-evaluation of the trust placed in selection mechanisms, which have traditionally been seen as purely efficiency-enhancing components.

Extension to Other Acquisition Functions. In this work, we focus on uncertainty-based acquisition functions, which are the most widely used category in AL literature. However, other acquisition functions—such as diversity-based (Sener and Savarese 2017; Gal, Islam, and Ghahramani 2017) or hybrid methods (Ash et al. 2019)—may also be vulnerable. Attacking such methods would require new criteria for identifying and optimizing poisoning samples, potentially relying on feature space distances or clustering behavior. We leave this as an important direction for future work.

Transferability and Black-box Potential. To explore the black-box applicability of our attack, we evaluated its transferability across models with different architectures or random initializations. We observed that both the poisoned selection rate and ASR under uncertainty-based acquisition functions become comparable to Random selection in the transfer setting. This holds even when adopting common transfer techniques based on ensembling, gradients, or input transformations. Notably, high-entropy samples are not consistent across models, highlighting a key limitation: uncertainty-based selection is highly sensitive to model-specific decision boundaries, even when performance is similar. These findings suggest that effective selection-aware attacks in the black-box setting remain a challenging open problem, and further research is needed to design transferable or model-agnostic poisoning techniques.

Toward Possible Defenses. Our findings underscore the need for defenses that explicitly consider vulnerabilities arising from the selection process. Potential countermeasures include entropy regularization (Pereyra et al. 2017) to mitigate model overconfidence, and anomaly detection mechanisms to identify suspicious samples selected during AL epochs. Another promising direction is to randomize or ensemble multiple acquisition functions, thereby limiting the attacker's ability to optimize against any fixed selection rule.

Overall, our study issues a new warning for the security of AI systems: acquisition functions, while intended to improve efficiency, can inadvertently introduce exploitable attack surfaces. To build truly robust AL systems, it is essential to integrate selection-aware threat modeling into the design of secure learning pipelines.

Conclusion

In this paper, we introduced a new type of attack for deep learning – poisoning attack for active learning. Leveraging the acquisition function as the attack surface, we proposed a novel framework, *ALA*, to inject poisoned samples into the training set of active learning models. To increase the attack efficiency, we designed a selection-aware optimization algorithm to maximize the probability of poisoned samples being selected. Evaluation results demonstrated that *ALA* is effective in attacking active learning, highlighting the need for secure active learning in the future.

References

Ash, J. T.; Zhang, C.; Krishnamurthy, A.; Langford, J.; and Agarwal, A. 2019. Deep batch active learning by diverse, uncertain gradient lower bounds. *arXiv preprint arXiv:1906.03671*.

Barni, M.; Kallas, K.; and Tondi, B. 2019. A new backdoor attack in cnns by training set corruption without label poisoning. In *2019 IEEE International Conference on Image Processing (ICIP)*, 101–105. IEEE.

Beluch, W. H.; Genewein, T.; Nürnberger, A.; and Köhler, J. M. 2018. The power of ensembles for active learning in image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 9368–9377.

Bilgic, M.; and Getoor, L. 2009. Link-based active learning. In *NIPS workshop on analyzing networks and learning with graphs*, volume 4, 9. Vancouver, BC, Canada.

Gal, Y.; Islam, R.; and Ghahramani, Z. 2017. Deep bayesian active learning with image data. In *International conference on machine learning*, 1183–1192. PMLR.

Gu, T.; Dolan-Gavitt, B.; and Garg, S. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hendrycks, D.; and Dietterich, T. 2019. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. *Proceedings of the International Conference on Learning Representations*.

Hu, Q.; Guo, Y.; Xie, X.; Cordy, M.; Ma, L.; Papadakis, M.; and Le Traon, Y. 2024. Active code learning: Benchmarking sample-efficient training of code models. *IEEE Transactions on Software Engineering*, 50(5): 1080–1095.

Ji, Y.; Zhang, X.; and Wang, T. 2017. Backdoor attacks against learning systems. In *2017 IEEE Conference on Communications and Network Security (CNS)*, 1–9. IEEE.

Joshi, A. J.; Porikli, F.; and Papanikolopoulos, N. 2009. Multi-class active learning for image classification. In *2009 ieee conference on computer vision and pattern recognition*, 2372–2379. IEEE.

Kim, J.; Feldt, R.; and Yoo, S. 2019. Guiding deep learning system testing using surprise adequacy. In Atlee, J. M.; Bultan, T.; and Whittle, J., eds., *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, 1039–1049. IEEE / ACM.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

LeCun, Y.; eon Bottou, L.; Bengio, Y.; et al. 1998. Gradient-Based Learning Applied to Document Recognition. *PROC. OF THE IEEE*, 1.

Li, D.; Wang, Z.; Chen, Y.; Jiang, R.; Ding, W.; and Okumura, M. 2024. A survey on deep active learning: Recent advances and new frontiers. *IEEE Transactions on Neural Networks and Learning Systems*, 36(4): 5879–5899.

Li, X.; and Guo, Y. 2013. Adaptive active learning for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 859–866.

Li, Y.; Chen, M.; Liu, Y.; He, D.; and Xu, Q. 2022. An empirical study on the efficacy of deep active learning for image classification. *arXiv preprint arXiv:2212.03088*.

Liu, Y.; Ma, S.; Aafer, Y.; Lee, W.-C.; Zhai, J.; Wang, W.; and Zhang, X. 2018. Trojaning attack on neural networks. In *25th Annual Network And Distributed System Security Symposium (NDSS 2018)*. Internet Soc.

Lu, H.; Jia, X.; Xie, Y.; Liao, W.; Yang, X.; and Yan, J. 2024. ActiveAD: Planning-Oriented Active Learning for End-to-End Autonomous Driving. *CoRR*, abs/2403.02877.

Munjal, P.; Hayat, N.; Hayat, M.; Sourati, J.; and Khan, S. 2022. Towards robust and reproducible active learning using neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 223–232.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; Ng, A. Y.; et al. 2011. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, 7. Granada.

Pereyra, G.; Tucker, G.; Chorowski, J.; Kaiser, Ł.; and Hinton, G. 2017. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*.

Ren, P.; Xiao, Y.; Chang, X.; Huang, P.-Y.; Li, Z.; Gupta, B. B.; Chen, X.; and Wang, X. 2021. A survey of deep active learning. *ACM computing surveys (CSUR)*, 54(9): 1–40.

Roy, N.; and McCallum, A. 2001. Toward optimal active learning through sampling estimation of error reduction. In *In Proc. 18th International Conf. on Machine Learning*.

Saha, A.; Subramanya, A.; and Pirsiavash, H. 2020. Hidden trigger backdoor attacks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 11957–11965.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4510–4520.

Santos, I.; and Carvalho, A. 2024. ProtoAL: Interpretable deep active learning with prototypes for medical imaging. In Zaza, G.; Casalino, G.; and Castellano, G., eds., *Proceedings of the First Workshop on Explainable Artificial Intelligence for the Medical Domain (EXPLIMED 2024) co-located with 27th European Conference on Artificial Intelligence (ECAI 2024), Santiago de Compostela, Spain, October 20, 2024*, volume 3831 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Sener, O.; and Savarese, S. 2017. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*.

Sener, O.; and Savarese, S. 2018. Active Learning for Convolutional Neural Networks: A Core-Set Approach. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Settles, B. 2009. Active learning literature survey.

Shannon, C. E. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3): 379–423.

Shui, C.; Zhou, F.; Gagné, C.; and Wang, B. 2020. Deep active learning: Unified and principled method for query and training. In *International conference on artificial intelligence and statistics*, 1308–1318. PMLR.

Tong, S.; and Koller, D. 2001. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov): 45–66.

Turner, A.; Tsipras, D.; and Madry, A. 2019. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*.

Wang, D.; and Shang, Y. 2014. A new active labeling method for deep learning. In *2014 International joint conference on neural networks (IJCNN)*, 112–119. IEEE.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747*.

Yin, C.; Qian, B.; Cao, S.; Li, X.; Wei, J.; Zheng, Q.; and Davidson, I. 2017. Deep similarity-based batch mode active learning with exploration-exploitation. In *2017 IEEE international conference on data mining (ICDM)*, 575–584. IEEE.

Zhong, H.; Liao, C.; Squicciarini, A. C.; Zhu, S.; and Miller, D. 2020. Backdoor embedding in convolutional neural network models via invisible perturbation. In *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*, 97–108.

Appendix

Poisoned Selection Rate

Table 3–14 present the Poisoned Selection Rate at the First and Last Training Epochs across all settings.

Table 3: Poisoned Selection Rate (%) on CIFAR-10 under the CL Attack (0.5% Poisoning)

Acquisition	Epoch	Iter 0	Iter 5	Iter 10	Iter 15
Random	0	1.0	1.2	0.6	1.2
	9	11.2	7.2	8.0	11.6
Entropy	0	5.8	46.0	71.4	80.8
	9	14.9	47.1	71.7	81.0
Margin	0	6.2	12.6	22.8	25.6
	9	14.2	21.5	28.4	31.3
Least Confidence	0	7.2	49.0	72.4	88.6
	9	15.5	51.3	73.1	88.8

Table 4: Poisoned Selection Rate (%) on Fashion-MNIST under the CL Attack (0.5% Poisoning)

Acquisition	Epoch	Iter 0	Iter 5	Iter 10	Iter 15
Random	0	1.8	1.4	1.2	1.6
	9	12.8	8.6	10.6	9.6
Entropy	0	0.0	18.2	40.6	60.6
	9	2.0	25.0	44.6	62.2
Margin	0	0.0	3.4	7.2	8.2
	9	6.2	11.0	14.0	13.4
Least Confidence	0	0.0	23.0	47.4	67.6
	9	3.2	28.0	49.6	68.4

Table 5: Poisoned Selection Rate (%) on SVHN under the CL Attack (0.5% Poisoning)

Acquisition	Epoch	Iter 0	Iter 5	Iter 10	Iter 15
Random	0	1.4	0.2	0.6	1.0
	9	13.0	8.4	8.6	11.6
Entropy	0	85.2	100.0	100.0	100.0
	9	85.4	100.0	100.0	100.0
Margin	0	10.2	14.4	19.4	25.4
	9	18.2	21.2	24.8	31.0
Least Confidence	0	61.0	95.0	98.4	99.8
	9	61.8	95.0	98.4	99.8

Table 6: Poisoned Selection Rate (%) on CIFAR-10 under the SIG Attack (0.5% Poisoning)

Acquisition	Epoch	Iter 0	Iter 5	Iter 10	Iter 15
Random	0	1.6	0.6	1.4	0.8
	9	7.6	10.0	9.4	8.6
Entropy	0	23.8	75.4	84.6	89.4
	9	29.1	75.9	84.8	89.4
Margin	0	4.6	18.0	26.2	32.4
	9	14.6	24.3	30.1	35.5
Least Confidence	0	23.8	78.8	89.4	95.8
	9	28.5	79.7	89.5	95.8

Table 7: Poisoned Selection Rate (%) on Fashion-MNIST under the SIG Attack (0.5% Poisoning)

Acquisition	Epoch	Iter 0	Iter 5	Iter 10	Iter 15
Random	0	1.4	1.8	0.4	0.2
	9	9.6	12.4	7.0	9.4
Entropy	0	15.2	53.2	72.0	86.0
	9	20.6	54.2	72.2	86.2
Margin	0	2.6	4.6	5.2	7.2
	9	8.4	11.4	10.8	11.8
Least Confidence	0	9.6	47.6	69.8	85.4
	9	16.2	48.4	70.4	85.4

Table 8: Poisoned Selection Rate (%) on SVHN under the SIG Attack (0.5% Poisoning)

Acquisition	Epoch	Iter 0	Iter 5	Iter 10	Iter 15
Random	0	1.4	0.2	0.6	1.0
	9	13.0	8.4	8.6	11.6
Entropy	0	85.2	100.0	100.0	100.0
	9	85.4	100.0	100.0	100.0
Margin	0	10.2	14.4	19.4	25.4
	9	18.2	21.2	24.8	31.0
Least Confidence	0	61.0	95.0	98.4	99.8
	9	61.8	95.0	98.4	99.8

Table 9: Poisoned Selection Rate (%) on CIFAR-10 under the CL Attack (1.0% Poisoning)

Acquisition	Epoch	Iter 0	Iter 5	Iter 10	Iter 15
Random	0	1.2	1.0	1.0	0.5
	9	9.5	11.0	10.8	8.8
Entropy	0	3.0	45.6	65.1	73.9
	9	11.5	46.0	65.4	74.0
Margin	0	3.7	14.3	19.7	23.9
	9	10.9	19.4	23.3	26.6
Least Confidence	0	3.6	44.0	67.9	77.4
	9	11.3	44.6	68.0	77.5

Table 10: Poisoned Selection Rate (%) on Fashion-MNIST under the CL Attack (1.0% Poisoning)

Acquisition	Epoch	Iter 0	Iter 5	Iter 10	Iter 15
Random	0	1.0	1.1	0.9	0.5
	9	11.0	10.6	10.7	8.4
Entropy	0	0.0	19.4	42.3	61.2
	9	3.0	23.6	43.8	61.8
Margin	0	0.0	4.1	6.0	8.4
	9	4.3	10.3	12.3	13.9
Least Confidence	0	0.0	24.8	47.9	64.5
	9	3.1	26.6	48.9	64.9

Table 11: Poisoned Selection Rate (%) on SVHN under the CL Attack (1.0% Poisoning)

Acquisition	Epoch	Iter 0	Iter 5	Iter 10	Iter 15
Random	0	0.9	0.7	0.8	0.8
	9	10.3	8.4	10.0	11.1
Entropy	0	0.0	31.5	51.2	62.3
	9	10.7	42.5	57.4	67.3
Margin	0	0.0	7.6	10.0	12.3
	9	8.3	20.6	22.7	25.6
Least Confidence	0	0.0	33.1	50.9	64.9
	9	10.4	41.8	57.9	69.3

Table 12: Poisoned Selection Rate (%) on CIFAR-10 under the SIG Attack (1.0% Poisoning)

Acquisition	Epoch	Iter 0	Iter 5	Iter 10	Iter 15
Random	0	1.2	1.0	1.4	0.8
	9	9.3	8.4	10.1	9.4
Entropy	0	11.9	58.2	69.4	73.4
	9	16.1	58.4	69.6	73.6
Margin	0	4.5	18.3	23.2	24.7
	9	12.1	21.7	25.9	27.0
Least Confidence	0	12.6	62.7	76.0	81.9
	9	16.7	63.2	76.1	82.0

Table 13: Poisoned Selection Rate (%) on Fashion-MNIST under the SIG Attack (1.0% Poisoning)

Acquisition	Epoch	Iter 0	Iter 5	Iter 10	Iter 15
Random	0	1.5	1.4	1.3	1.0
	9	9.1	10.3	9.5	10.1
Entropy	0	7.8	44.6	65.7	79.4
	9	12.7	45.3	65.9	79.7
Margin	0	2.0	5.5	6.6	9.8
	9	8.2	7.5	9.8	11.5
Least Confidence	0	5.4	41.6	61.1	76.0
	9	10.1	41.7	61.2	76.0

Table 14: Poisoned Selection Rate (%) on SVHN under the SIG Attack (1.0% Poisoning)

Acquisition	Epoch	Iter 0	Iter 5	Iter 10	Iter 15
Random	0	0.6	1.6	0.9	0.9
	9	10.4	9.3	9.0	11.0
Entropy	0	43.6	74.8	82.5	86.1
	9	44.1	75.0	82.7	86.2
Margin	0	7.2	14.7	17.0	21.1
	9	13.3	17.7	19.6	23.6
Least Confidence	0	37.1	74.9	84.3	88.4
	9	38.5	75.1	84.3	88.6

Attack success rate

Table 15–26 present the Final ASR (%) for different acquisition functions and optimization rounds. Figure 5-16 show the ASR (%) across epochs. Table 39 shows the Poisoned Selection Rate and the corresponding ASR on the CIFAR-10 dataset using the SIG trigger with a poisoning ratio of 0.5%.

Table 15: Final ASR (%) on CIFAR-10 under the CL attack with 0.5% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where all poisoned samples are selected in the first AL epoch. Iter refers to optimization iteration.

ASR (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	55.1	55.1	55.1	55.1
Random	11.6	10.4	10.4	10.8
Entropy	12.0	16.7	20.9	25.0
Margin	13.1	11.9	12.5	13.7
Least Confidence	11.9	16.2	22.0	26.5

Table 16: Final ASR (%) on Fashion-MNIST under the CL attack with 0.5% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where all poisoned samples are selected in the first AL epoch. Iter refers to optimization iteration.

ASR (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	35.8	35.8	35.8	35.8
Random	16.9	12.5	11.0	13.3
Entropy	11.3	13.0	15.7	20.2
Margin	11.9	12.7	14.2	13.7
Least Confidence	10.5	14.7	17.4	24.4

Table 17: Final ASR (%) on SVHN under the CL attack with 0.5% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where all poisoned samples are selected in the first AL epoch. Iter refers to optimization iteration.

ASR (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	10.1	10.1	10.1	10.1
Random	10.1	10.2	10.0	10.0
Entropy	10.1	10.0	10.1	10.0
Margin	10.1	10.1	10.0	10.2
Least Confidence	10.1	10.1	10.1	10.1

Table 18: Final ASR (%) on CIFAR-10 under the SIG attack with 0.5% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where all poisoned samples are selected in the first AL epoch. Iter refers to optimization iteration.

ASR (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	46.0	46.0	46.0	46.0
Random	14.6	16.0	17.7	16.4
Entropy	29.7	43.6	43.8	43.7
Margin	21.0	27.3	28.5	30.0
Least Confidence	27.1	43.3	47.7	48.2

Table 19: Final ASR (%) on Fashion-MNIST under the SIG attack with 0.5% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where all poisoned samples are selected in the first AL epoch. Iter refers to optimization iteration.

ASR (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	33.3	33.3	33.3	33.3
Random	17.0	19.3	14.9	18.1
Entropy	20.5	25.9	27.8	29.0
Margin	17.5	18.5	19.4	21.7
Least Confidence	21.3	27.8	27.2	33.6

Table 20: Final ASR (%) on SVHN under the SIG attack with 0.5% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where all poisoned samples are selected in the first AL epoch. Iter refers to optimization iteration.

ASR (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	91.6	91.6	91.6	91.6
Random	74.0	51.1	65.7	64.6
Entropy	90.5	90.9	93.2	93.0
Margin	80.9	83.8	81.5	77.3
Least Confidence	87.6	92.6	92.5	87.8

Table 21: Final ASR (%) on CIFAR-10 under the CL attack with 1.0% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where all poisoned samples are selected in the first AL epoch. Iter refers to optimization iteration.

ASR (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	66.6	66.6	66.6	66.6
Random	12.3	12.2	11.6	11.5
Entropy	15.1	27.4	33.3	33.8
Margin	16.3	19.2	17.6	20.5
Least Confidence	13.5	27.5	33.2	34.9

Table 22: Final ASR (%) on Fashion-MNIST under the CL attack with 1.0% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where all poisoned samples are selected in the first AL epoch. Iter refers to optimization iteration.

ASR (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	47.6	47.6	47.6	47.6
Random	17.4	13.4	13.3	13.3
Entropy	10.7	16.4	20.3	22.4
Margin	14.8	15.1	15.3	15.0
Least Confidence	11.5	17.0	20.4	23.8

Table 23: Final ASR (%) on SVHN under the CL attack with 1.0% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where all poisoned samples are selected in the first AL epoch. Iter refers to optimization iteration.

ASR (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	10.1	10.1	10.1	10.1
Random	10.1	10.2	10.0	10.1
Entropy	10.0	10.1	9.9	10.1
Margin	10.1	10.0	10.1	10.1
Least Confidence	10.1	10.2	10.1	10.2

Table 24: Final ASR (%) on CIFAR-10 under the SIG attack with 1.0% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where all poisoned samples are selected in the first AL epoch. Iter refers to optimization iteration.

ASR (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	60.3	60.3	60.3	60.3
Random	24.1	20.8	23.8	19.9
Entropy	30.5	50.3	48.3	48.1
Margin	30.2	36.6	38.3	38.1
Least Confidence	34.4	50.6	53.5	52.2

Table 25: Final ASR (%) on Fashion-MNIST under the SIG attack with 1.0% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where all poisoned samples are selected in the first AL epoch. Iter refers to optimization iteration.

ASR (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	31.5	31.5	31.5	31.5
Random	19.5	24.0	23.1	21.6
Entropy	24.0	25.5	31.9	34.2
Margin	21.3	24.9	25.8	27.0
Least Confidence	20.9	33.5	30.4	34.8

Table 26: Final ASR (%) on SVHN under the SIG attack with 1.0% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where all poisoned samples are selected in the first AL epoch. Iter refers to optimization iteration.

ASR (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	94.3	94.3	94.3	94.3
Random	77.1	74.5	80.0	84.4
Entropy	86.9	93.3	91.1	94.0
Margin	88.0	88.8	85.7	90.6
Least Confidence	91.6	94.0	93.3	94.1

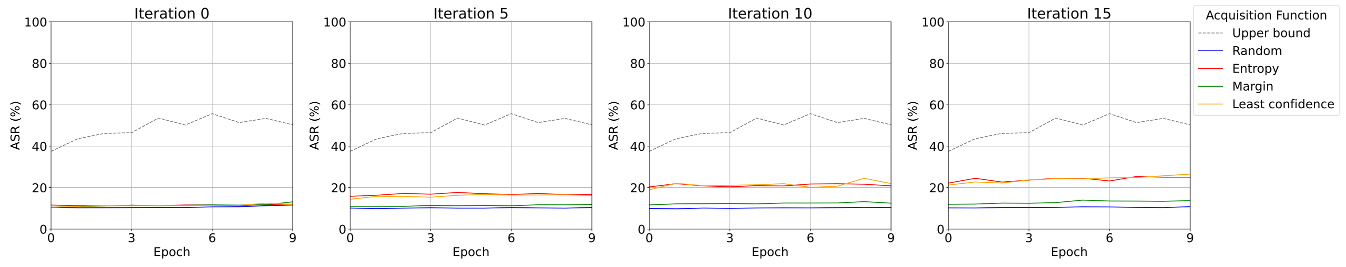

Figure 5: ASR (%) over training epochs on CIFAR-10 under the CL trigger with 0.5% poisoning. Colored lines represent different acquisition functions, while the dashed line denotes the estimated upper bound achieved when all poisoned samples are selected in the first AL epoch.

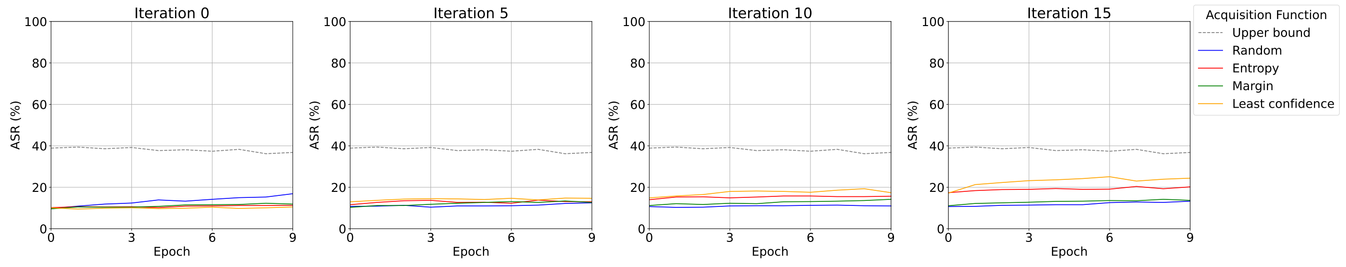

Figure 6: ASR (%) over training epochs on Fashion-MNIST under the CL trigger with 0.5% poisoning.

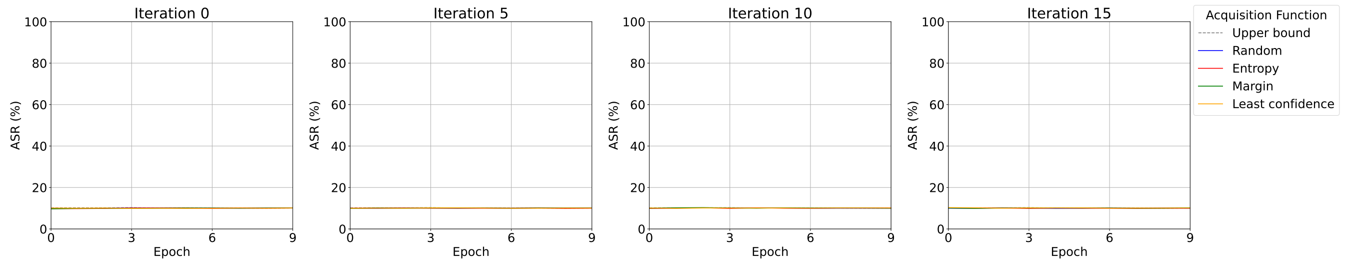

Figure 7: ASR (%) over training epochs on SVHN under the CL trigger with 0.5% poisoning.

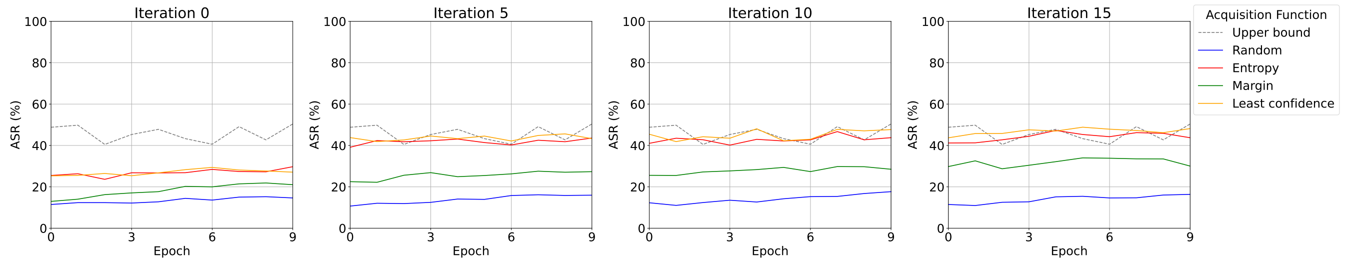

Figure 8: ASR (%) over training epochs on CIFAR-10 under the SIG trigger with 0.5% poisoning.

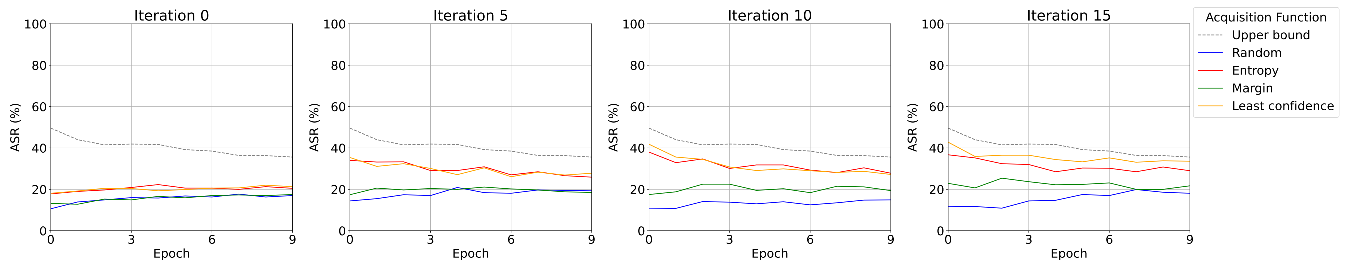

Figure 9: ASR (%) over training epochs on Fashion-MNIST under the SIG trigger with 0.5% poisoning.

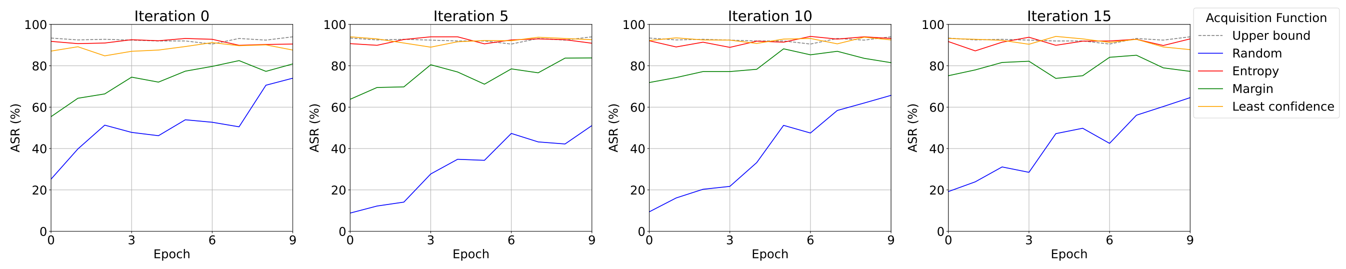

Figure 10: ASR (%) over training epochs on SVHN under the SIG trigger with 0.5% poisoning.

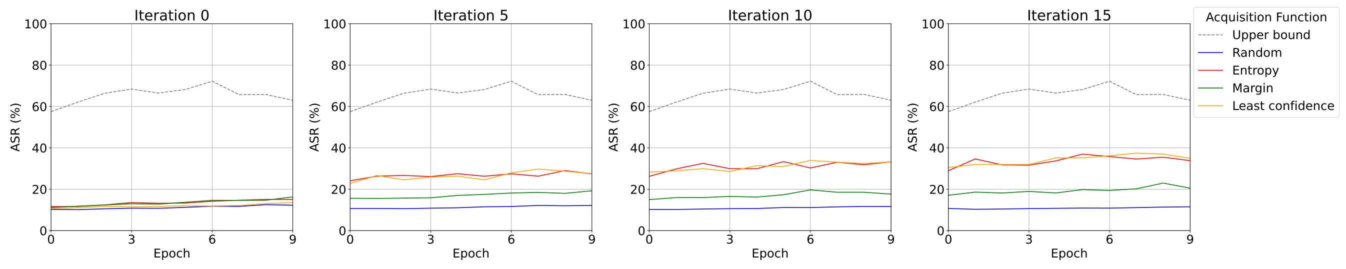

Figure 11: ASR (%) over training epochs on CIFAR-10 under the CL trigger with 1.0% poisoning.

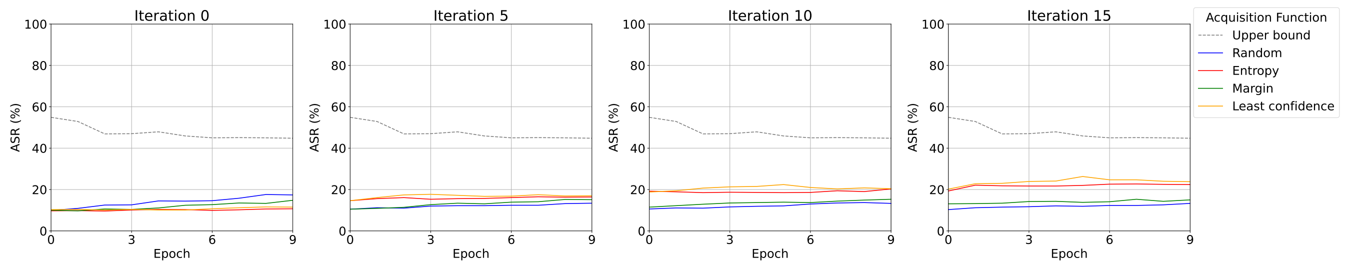

Figure 12: ASR (%) over training epochs on Fashion-MNIST under the CL trigger with 1.0% poisoning.

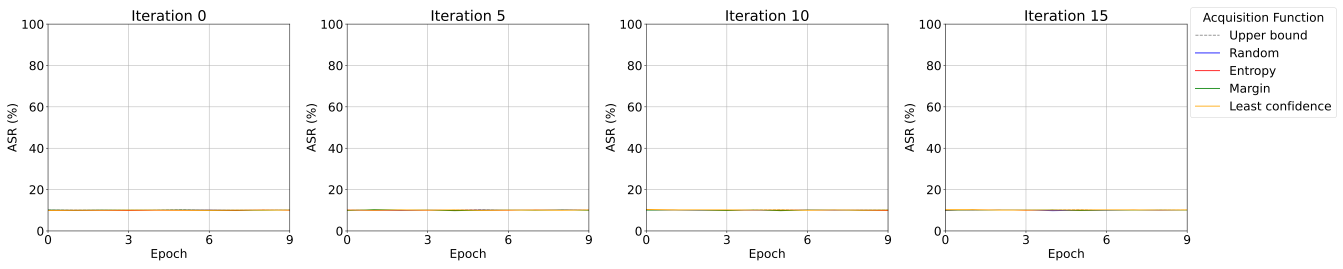

Figure 13: ASR (%) over training epochs on SVHN under the CL trigger with 1.0% poisoning.

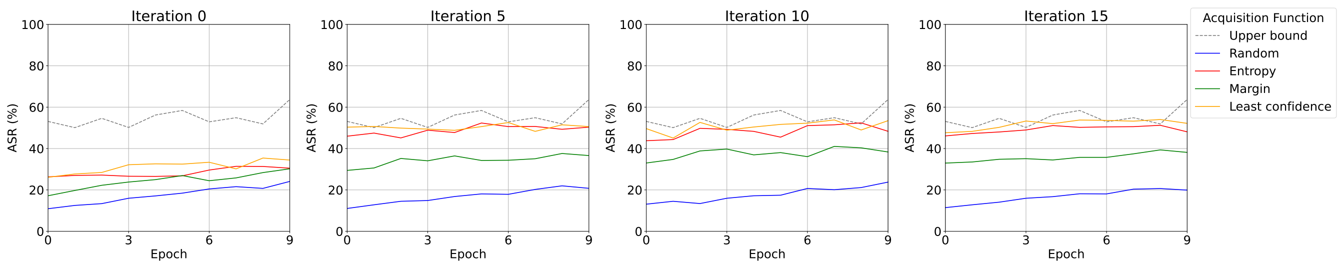

Figure 14: ASR (%) over training epochs on CIFAR-10 under the SIG trigger with 1.0% poisoning.

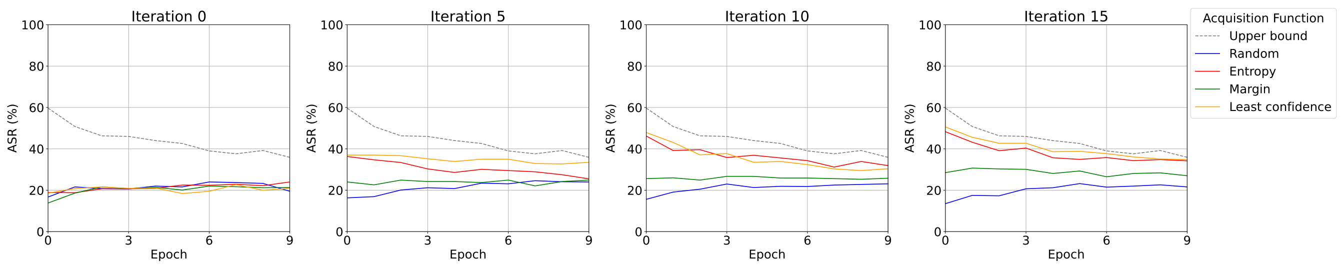

Figure 15: ASR (%) over training epochs on Fashion-MNIST under the SIG trigger with 1.0% poisoning.

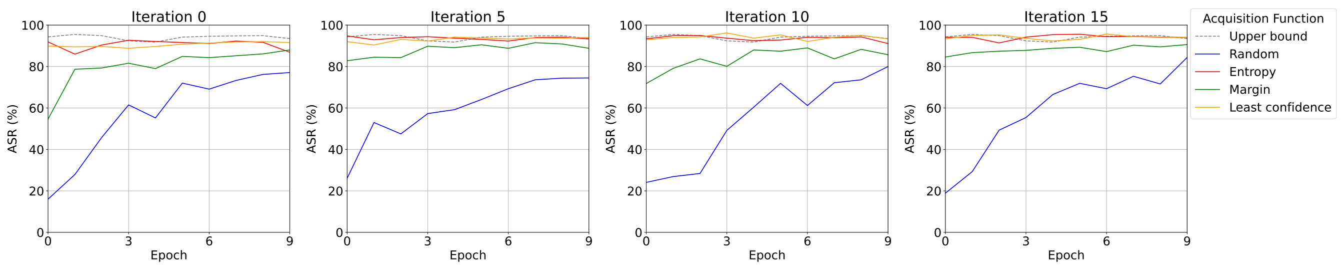

Figure 16: ASR (%) over training epochs on SVHN under the SIG trigger with 1.0% poisoning.

ASR over different classes

Figure 17 and Figure 18 show the ASR (%) of epoch 9 over different classes on Fashion-MNIST and SVHN dataset.

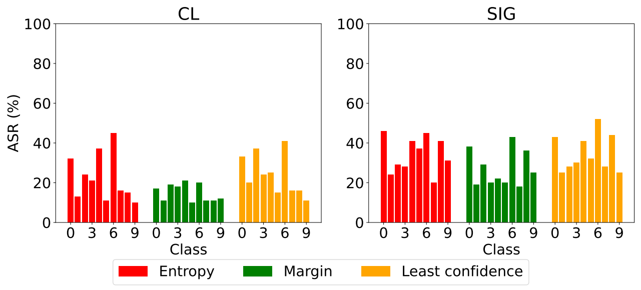

Figure 17: ASR (%) of epoch 9 over different classes on Fashion-MNIST under the CL and SIG trigger with 1.0% poisoning and 15 optimization iterations. Colored lines represent different acquisition functions, while the dashed line denotes the estimated upper bound achieved when all poisoned samples are selected in the first AL epoch.

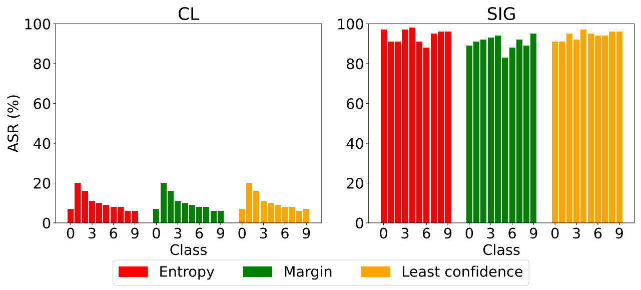

Figure 18: ASR (%) of epoch 9 over different classes on SVHN under the CL and SIG trigger with 1.0% poisoning and 15 optimization iterations. Colored lines represent different acquisition functions, while the dashed line denotes the estimated upper bound achieved when all poisoned samples are selected in the first AL epoch.

OOD data accuracy

Tables 27–38 show the prediction accuracy of the fine-tuned models on OOD data (i.e., all benign unlabeled data). The results indicate that the backdoor injection process does not affect the model's performance on benign samples.

Table 27: OOD Accuray (%) on CIFAR-10 under the CL attack with 0.5% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where no poisoned samples are injected in unlabeled pool. ACC refers to OOD accuracy and Iter refers to optimization iteration.

ACC (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	55.0	55.0	55.0	55.0
Random	56.3	56.5	56.3	57.5
Entropy	55.7	56.0	55.9	55.4
Margin	56.9	57.0	57.9	57.5
Least Confidence	56.3	56.7	56.6	56.0

Table 28: OOD Accuray (%) on Fashion-MNIST under the CL attack with 0.5% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where no poisoned samples are injected in unlabeled pool. ACC refers to OOD accuracy and Iter refers to optimization iteration.

ACC (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	51.0	51.0	51.0	51.0
Random	50.3	50.1	49.6	50.4
Entropy	53.3	52.7	53.2	53.0
Margin	51.5	51.6	52.2	52.1
Least Confidence	52.7	52.9	52.7	53.1

Table 29: OOD Accuray (%) on SVHN under the CL attack with 0.5% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where no poisoned samples are injected in unlabeled pool. ACC refers to OOD accuracy and Iter refers to optimization iteration.

ACC (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	62.0	62.0	62.0	62.0
Random	63.0	62.7	62.6	62.3
Entropy	63.0	62.9	61.5	61.9
Margin	64.4	64.0	63.6	63.9
Least Confidence	64.1	64.0	62.9	63.2

Table 30: OOD Accuray (%) on CIFAR-10 under the SIG attack with 0.5% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where no poisoned samples are injected in unlabeled pool. ACC refers to OOD accuracy and Iter refers to optimization iteration.

ACC (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	55.0	55.0	55.0	55.0
Random	56.5	56.8	57.3	57.8
Entropy	56.3	55.8	56.1	55.2
Margin	57.5	57.4	57.4	57.2
Least Confidence	56.7	57.2	56.5	56.6

Table 31: OOD Accuray (%) on Fashion-MNIST under the SIG attack with 0.5% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where no poisoned samples are injected in unlabeled pool. ACC refers to OOD accuracy and Iter refers to optimization iteration.

ACC (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	51.0	51.0	51.0	51.0
Random	50.2	50.5	50.6	50.1
Entropy	53.6	52.5	52.4	52.5
Margin	52.3	52.1	51.9	52.2
Least Confidence	53.1	52.9	53.0	53.0

Table 32: OOD Accuray (%) on SVHN under the SIG attack with 0.5% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where no poisoned samples are injected in unlabeled pool. ACC refers to OOD accuracy and Iter refers to optimization iteration.

ACC (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	62.0	62.0	62.0	62.0
Random	62.3	61.9	62.3	62.1
Entropy	61.2	61.5	61.3	62.1
Margin	64.4	64.2	63.7	63.5
Least Confidence	62.5	61.9	62.3	62.1

Table 33: OOD Accuray (%) on CIFAR-10 under the CL attack with 1.0% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where no poisoned samples are injected in unlabeled pool. ACC refers to OOD accuracy and Iter refers to optimization iteration.

ACC (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	55.0	55.0	55.0	55.0
Random	56.2	57.2	57.0	56.8
Entropy	56.2	56.2	55.4	55.4
Margin	57.0	57.3	57.4	57.6
Least Confidence	56.1	56.2	55.6	55.7

Table 34: OOD Accuray (%) on Fashion-MNIST under the CL attack with 1.0% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where no poisoned samples are injected in unlabeled pool. ACC refers to OOD accuracy and Iter refers to optimization iteration.

ACC (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	51.0	51.0	51.0	51.0
Random	49.7	50.5	49.9	50.2
Entropy	53.1	53.3	53.4	52.8
Margin	51.3	52.2	52.0	52.0
Least Confidence	52.8	52.7	53.0	52.4

Table 35: OOD Accuray (%) on SVHN under the CL attack with 1.0% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where no poisoned samples are injected in unlabeled pool. ACC refers to OOD accuracy and Iter refers to optimization iteration.

ACC (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	62.0	62.0	62.0	62.0
Random	62.1	61.8	62.1	62.0
Entropy	61.3	62.0	61.9	61.1
Margin	63.2	63.8	63.4	63.3
Least Confidence	62.5	62.9	63.1	62.7

Table 36: OOD Accuray (%) on CIFAR-10 under the SIG attack with 1.0% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where no poisoned samples are injected in unlabeled pool. ACC refers to OOD accuracy and Iter refers to optimization iteration.

ACC (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	55.0	55.0	55.0	55.0
Random	57.4	58.2	57.7	57.5
Entropy	56.6	55.8	55.4	55.3
Margin	57.6	57.6	57.5	57.5
Least Confidence	56.9	56.0	56.0	56.1

Table 37: OOD Accuray (%) on Fashion-MNIST under the SIG attack with 1.0% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where no poisoned samples are injected in unlabeled pool. ACC refers to OOD accuracy and Iter refers to optimization iteration.

ACC (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	51.0	51.0	51.0	51.0
Random	49.7	50.3	50.5	50.3
Entropy	53.5	52.7	52.6	53.1
Margin	52.7	52.3	52.4	52.1
Least Confidence	53.1	53.3	52.4	52.6

Table 38: OOD Accuray (%) on SVHN under the SIG attack with 1.0% poisoning for different acquisition functions and optimization rounds. The first row reports the estimated upper bound, where no poisoned samples are injected in unlabeled pool. ACC refers to OOD accuracy and Iter refers to optimization iteration.

ACC (%)	Iter 0	Iter 5	Iter 10	Iter 15
Upper bound	62.0	62.0	62.0	62.0
Random	62.6	62.8	62.6	62.2
Entropy	62.3	61.2	61.7	61.1
Margin	63.9	64.0	63.7	64.2
Least Confidence	62.8	62.8	62.2	62.2

Table 39: Raw data to calculate correlation between R_{select} and ASR

Iteration	Acquisition Fcuntion	Metric	Epoch									
			0	1	2	3	4	5	6	7	8	9
0	Random	R_{select}	11.5	12.4	12.4	12.2	12.8	14.4	13.6	15.0	15.2	14.6
		ASR	1.6	2.2	2.8	3.4	3.8	5.2	5.6	6.0	6.8	7.6
	Entropy	R_{select}	25.5	26.3	23.6	26.8	26.7	26.8	28.4	27.4	27.3	29.7
		ASR	23.8	24.6	24.9	25.6	25.9	26.3	27.2	27.9	28.5	29.1
	Margin	R_{select}	13.0	14.0	16.2	17.1	17.7	20.2	20.0	21.4	21.9	21.0
		ASR	4.6	5.5	6.7	7.9	9.7	10.7	11.7	12.9	13.7	14.6
	Least Confidence	R_{select}	25.3	25.5	26.5	25.4	26.8	28.3	29.4	28.1	27.7	27.1
		ASR	23.8	24.1	24.5	24.9	25.8	26.3	27.1	27.4	28.1	28.5
5	Random	R_{select}	10.7	12.1	11.9	12.5	14.1	13.9	15.8	16.2	15.8	16.0
		ASR	0.6	1.6	3.4	4.4	5.0	5.8	7.4	8.0	9.2	10.0
	Entropy	R_{select}	39.1	42.4	41.8	42.3	43.1	41.4	40.2	42.5	41.8	43.6
		ASR	75.4	75.5	75.5	75.6	75.7	75.7	75.7	75.7	75.7	75.9
	Margin	R_{select}	22.5	22.2	25.6	26.9	24.9	25.5	26.3	27.6	27.1	27.3
		ASR	18.0	19.2	20.0	20.5	21.1	21.5	22.3	23.1	23.7	24.3
	Least Confidence	R_{select}	43.8	41.9	42.7	44.6	43.3	44.5	42.2	44.8	45.6	43.3
		ASR	78.8	78.9	79.1	79.1	79.2	79.2	79.3	79.6	79.7	79.7
10	Random	R_{select}	12.3	11.0	12.4	13.5	12.7	14.2	15.3	15.4	16.8	17.7
		ASR	1.4	2.4	3.2	3.6	4.8	5.4	6.2	7.2	8.2	9.4
	Entropy	R_{select}	41.0	43.5	42.8	40.2	43.0	42.2	42.7	46.7	42.7	43.8
		ASR	84.6	84.6	84.6	84.6	84.6	84.6	84.6	84.7	84.7	84.8
	Margin	R_{select}	25.6	25.5	27.2	27.7	28.3	29.4	27.4	29.8	29.7	28.5
		ASR	26.2	26.5	26.8	27.4	27.9	28.4	28.9	29.5	29.8	30.1
	Least Confidence	R_{select}	45.4	41.9	44.2	43.5	48.1	42.3	43.0	47.8	47.1	47.7
		ASR	89.4	89.4	89.4	89.4	89.4	89.4	89.4	89.5	89.5	89.5
15	Random	R_{select}	11.5	11.0	12.5	12.8	15.2	15.4	14.6	14.7	16.0	16.4
		ASR	0.8	1.6	2.4	3.2	5.2	5.6	5.8	7.0	7.8	8.6
	Entropy	R_{select}	41.2	41.3	42.7	44.4	47.3	45.3	44.2	46.2	45.9	43.7
		ASR	89.4	89.4	89.4	89.4	89.4	89.4	89.4	89.4	89.4	89.4
	Margin	R_{select}	29.8	32.6	28.7	30.4	32.2	34.0	33.8	33.5	33.5	30.0
		ASR	32.4	32.5	32.9	33.6	34.2	34.5	35.1	35.4	35.5	35.5
	Least Confidence	R_{select}	43.7	45.8	45.8	47.6	47.0	48.8	47.9	47.2	46.2	48.2
		ASR	95.8	95.8	95.8	95.8	95.8	95.8	95.8	95.8	95.8	95.8