JOLTing ROBUST AGGREGATION WITH TRIGGER OPTIMIZED CLEAN-IMAGE BACKDOOR POISONING

Anonymous authors
Paper under double-blind review

ABSTRACT

In this work, we study the attack surface of planting backdoors with clean-image poisoning in models defended by robust aggregation. This setting is relevant because labels can be easier to control than inputs (e.g., likes/dislikes in social media or crowd-sourcing), and cause potentially stealthier attacks than feature manipulation. However, clean-image poisoning offers more limited control over the training process, making backdoor implantation more challenging. First, we provide empirical evidence that FLIP, a state-of-the-art gradient-based clean-image backdoor attack, fails under robust aggregation, across datasets, and architectures. Second, we propose the Joint Optimization of Labels and Trigger (*JOLT*) method. We show that, in both centralized and federated settings, and without knowledge of the defense, *JOLT* successfully plants backdoors with high attack success rates, while maintaining a good clean test accuracy and small poisoning budget. Along the way, we identify an inconsistency between the implementation of FLIP and the described method, which gives a misleading impression of efficiency. To the best of our knowledge, *JOLT* is the first clean-image poisoning attack that successfully plants backdoors under robust aggregation defenses.

1 INTRODUCTION

As machine learning systems are increasingly deployed, their training process becomes a critical attack surface (El-Mhamdi et al., 2022). Through gradient-based attacks, malicious participants can subtly manipulate model updates while keeping their contributions statistically similar to those of honest clients (El-Mhamdi et al., 2018; Baruch et al., 2019). To defend against such attacks, robust aggregation rules have been developed. Based on robust statistics, they aim to limit the influence of malicious updates and recover a representative update from honest clients (Blanchard et al., 2017; Yin et al., 2018; Pillutla et al., 2022). More recently, data poisoning has been shown to reproduce the harm of malicious gradient-based attacks (Bouaziz et al., 2024; El-Kabid and El-Mhamdi, 2026) and to successfully backdoor models *i.e.,* to preserve the model's performance on clean data, while inducing targeted behavior in the presence of a trigger (Gu et al., 2019; Geiping et al., 2020).

Among poisoning attacks, clean-image poisoning imposes particularly strong constraints on the attacker: they can change the labels of the training examples, but not their features. Such settings arise for example in crowdsourced annotation, and more generally user-feedback systems (Rashtchian et al., 2010; Welinder et al., 2010; Jagabathula et al., 2017). Jha et al. (2023) and Xu et al. (2026) have shown that clean-image poisoning can plant backdoors in undefended models. We thus ask:

Can clean-image poisoning attacks induce backdoors in models defended by robust aggregation rules?

In this work, we first study the state-of-the-art gradient-based clean-image backdoor attack FLIP (Jha et al., 2023). We provide empirical evidence that FLIP's effectiveness varies when changing dataset and model

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

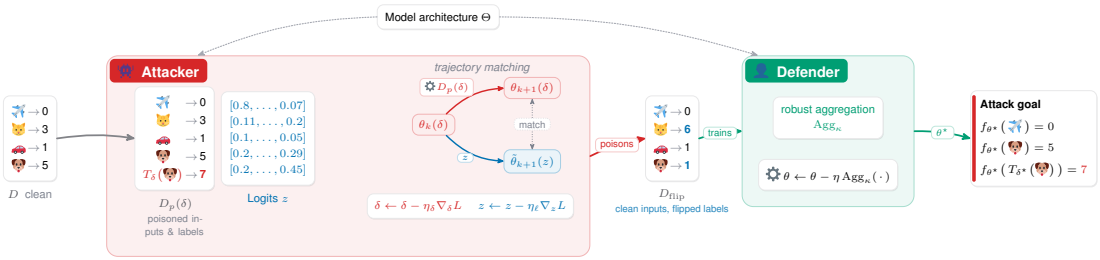


Figure 1: The attacker jointly optimizes poisoned labels and a private trigger $\delta^\star$, without knowing the defender's robust aggregation rule. The resulting clean-image dataset D_{flip} is used to train a backdoored model that predicts the target class only when a source-class input contains $\delta^\star$.

architectures, and that it substantially degrades when the end user's model is defended by robust aggregation rules. This answers our question by the negative for current methods.

We then hypothesize that the observed failure of current clean-image methods in presence of robust aggregation defense is due to the limited expressivity of only flipping a small number of labels, while keeping the trigger fixed. We introduce the Joint Optimization of Labels and Trigger method (*JOLT*), illustrated in Figure 1, as a means to test the above assumption. We provide empirical evidence that *JOLT* can successfully induce backdoors in models defended by robust aggregation rules, in federated settings, across datasets and model architectures. Perhaps surprisingly, *JOLT* does not require knowledge of the defender's aggregation rule. To the best of our knowledge, this is the first demonstration that label-only clean-image poisoning can successfully plant backdoors in the presence of robust aggregation. Along the way, we identify and resolve an inconsistency between the formulation and implementation of FLIP.

The paper is organized as follows. Section 2 situates our work within the relevant literature. Section 3 details the threat model. Section 4 discusses the limitations of FLIP's algorithm. We then propose an adjustment based on the joint optimization of the trigger and the labels in Section 5. Section 6 presents and interprets the experimental results. Section 7 concludes.

2 Related Work

Gradient-based attacks and defenses. Studies of gradient-based attacks in the context of *Robust Distributed Learning* have shown that omniscient attackers can subtly manipulate model updates while keeping their contributions statistically similar to those of honest clients (Baruch et al., 2019; El-Mhamdi et al., 2018; Xie et al., 2020). Attackers can thus steer the training dynamics toward targeted or untargeted model parameters. To defend against such threats, users can apply robust aggregation rules during the update step. These aggregators aim to limit the influence of malicious updates and recover a representative update from honest clients (Blanchard et al., 2017; Yin et al., 2018; El-Mhamdi et al., 2018; Allouah et al., 2023; Pillutla et al., 2022). However, these defense mechanisms can still be circumvented by carefully crafted attack strategies (Baruch et al., 2019; El-Mhamdi et al., 2018; Xie et al., 2020).

Data poisoning. Attackers can alternatively manipulate a fraction of the training samples to achieve malicious objectives, through label manipulation (Shejwalkar et al., 2022), feature manipulation with generative approaches (Muñoz-González et al., 2019; Zhao and Lao, 2022), or gradient-based poisoning (Geiping et al., 2020; Muñoz-González et al., 2017; Shafahi et al., 2018). Gradient-based poisoning methods aim to craft poisoned samples whose induced gradients approximate those that would have been produced directly by an omniscient attacker (Bouaziz et al., 2024; El-Kabid and El-Mhamdi, 2026). This provides a connection

between gradient-space attacks and data-space attacks: an attacker can seek to induce a desired training update through modifications to the training data rather than directly controlling the gradient.

Backdoor attacks. Backdoor attacks aim to preserve the overall performance of a model while inducing malicious behavior on inputs containing a specific *trigger* (Gu et al., 2019). A large body of work has investigated trigger design, including imperceptible perturbations and naturally occurring or physically realizable triggers (Gao et al., 2023; Chen et al., 2022; Wenger et al., 2022; Doan et al., 2021; Li et al., 2020; Wang et al., 2020).

Clean-image backdoor attacks. Backdoor attacks have been shown to be realizable without exposing the trigger during training. For image classification, Shumailov et al. (2021) achieve this by optimizing the training batch order. Clean-image backdoors have also been studied in label poisoning settings. For multi-label classification, Chen et al. (2022) manipulates specific label combinations, relying on properties specific to multi-label learning. For single-label classification, Jha et al. (2023) introduced FLIP, which enables backdoor attacks by flipping a small fraction of labels, building on trajectory matching for dataset distillation (Cazenavette et al., 2022).

Trigger optimization for clean-image backdoor attacks. Beyond fixed or hand-crafted triggers, several works optimize the trigger itself to improve attack effectiveness. Jin et al. (2026) optimizes triggers for backdoors induced purely through malicious reordering of clean data (Shumailov et al., 2021), showing that trigger choice affects how well clean gradients align with the adversarial objective. GCB (Xu et al., 2026) instead searches for naturally occurring, discriminative variations using a conditional InfoGAN for label-poisoning backdoor attacks.

Our work differs from GCB in the objective of trigger optimization. Rather than searching for semantic features that can serve as effective triggers, we jointly optimize the trigger and the poisoned labels to search for backdoor trajectories that are reachable under the attacker's constrained write access. We further study this reachability under robust aggregation rules, extending the analysis of restricted poisoning attacks (El-Kabid and El-Mhamdi, 2026; Bouaziz et al., 2024).

3 Threat model

We work in a supervised learning setting in which the defender and the attacker aim to solve two different supervised learning problems:

$$\min_{\theta\in\Theta} \mathcal{L}_c(\theta) := \mathbb{E}_{(x,y)\sim\mathcal{D}_c}\big[\ell(f_\theta(x),y)\big] \qquad \text{and} \qquad \min_{\theta\in\Theta} \mathcal{L}_p(\theta) := \mathbb{E}_{(x,y)\sim\mathcal{D}_p}\big[\ell(f_\theta(x),y)\big]. \tag{1}$$

where $\mathcal{D}_c$ and $\mathcal{D}_p$ denote the *clean* and *poisoned* distributions over the data space $\mathcal{X}\times\mathcal{Y}$, $\Theta\subset\mathbb{R}^d$ is the set of model parameters, and ℓ the sample-wise loss. The attacker chooses $\mathcal{D}_p$ according to its goal, as defined in Definition 3.1, while the defender solves the problem over $\mathcal{D}_c$.

Definition 3.1 (Backdoor attack)**.** *Let $T:\mathcal{X}\to\mathcal{X}$ be a measurable* trigger *map and let $y_{\text{source}}\neq y_{\text{target}}\in\mathcal{Y}$ be a source/target label pair. The associated* backdoor operator *is the measurable map*

$$\Phi:\mathcal{X}\times\mathcal{Y}\to\mathcal{X}\times\mathcal{Y}, \qquad \Phi(x,y)=\begin{cases}\big(T(x),\,y_{\text{target}}\big), & y=y_{\text{source}},\\ (x,y), & \textit{otherwise.}\end{cases}$$

Assuming $p_s := \mathbb{P}_{\mathcal{D}_c}(Y=y_{\text{source}})>0$, we write $\mathcal{D}_s := \mathcal{D}_c(\cdot\mid Y=y_{\text{source}})$ for the source-class conditional and $\mathcal{X}_s\subset\mathcal{X}$ for the $\mathcal{X}$-marginal of its support.

Define the pushforward $\Phi_{\#}\mathcal{D}_s$ *as the law of* $\big(T(X), y_{\text{target}}\big)$ *with* $(X, Y) \sim \mathcal{D}_s$*. For* $\lambda \in (0, 1)$*, the attacker targets the mixture* $\mathcal{D}_p = (1-\lambda)\,\mathcal{D}_c + \lambda\,\Phi_{\#}\mathcal{D}_s$*, and solves* $\min_{\theta\in\Theta} \mathcal{L}_p(\theta)$ *with :*

$$\mathcal{L}_p(\theta) = (1-\lambda)\,\mathcal{L}_c(\theta) + \lambda\,\mathbb{E}_{(X,Y)\sim\mathcal{D}_s}\Big[\ell\big(f_\theta(T(X)),\, y_{\text{target}}\big)\Big]. \tag{2}$$

We model the defender's learning process as a distributed process involving n_u Gradient Generation Units (GGU)[1]. At each round $k \in \mathbb{N}$, each unit produces a gradient $g_i(\theta_k)$, forming $S_k = \{g_i(\theta_k)\}_{i=1}^{n_u}$. Gradients are aggregated by an operator $\text{Agg} : \mathbb{R}^{d\times n_u} \to \mathbb{R}^d$, and the model is then updated with an update rule[2] $\text{Update} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$:

$$\theta_{k+1} = \text{Update}(\theta_k, \text{Agg}(S_k)). \tag{3}$$

Attacker setup. We work with the same threat model than Jha et al. (2023). We assume that the attacker has strong read access: it knows the defender's model architecture, optimizer, and training data. In contrast, we assume a limited write access: the attacker can only modify the training set by changing a small fraction of the labels. The attack is performed offline through the poisoned training dataset; the attacker has no access to the victim model during training (see Figure 1). Importantly, the attacker does not know the aggregation rule Agg used by the defender.

We evaluate the success of an attack using two metrics that the attacker aims to maximize:

- **Clean Test Accuracy (CTA)**: $CTA = \mathbb{P}_{(x,y)\sim\mathcal{D}_c}(f_\theta(x) = y)$.
- **Attack Success Rate (ASR)**: $ASR = \mathbb{P}_{x\sim\mathcal{D}_s}(f_\theta(T(x)) = y_{\text{target}})$.

These metrics correspond to the two objectives encoded in $\mathcal{L}_p$: preserving clean performance through $\mathcal{L}_c$ and inducing the targeted backdoor behavior through its second term. Thus, minimizing $\mathcal{L}_p$ provides a differentiable surrogate for jointly preserving clean performance and inducing the targeted backdoor behavior.

4 Label poisoning is not all you need

In this section, we first recall the attack procedure of FLIP (Jha et al., 2023). We then run FLIP on two architectures (ResNet-32 and ConvNeXtMicro) and two datasets (CIFAR-10 and SVHN). Experimental details are given in Appendices C and E.1. We provide empirical evidence that its effectiveness degrades across datasets, model architectures, and robust aggregation rules.

4.1 FLIP's attack procedure

The FLIP attack is performed in four steps:

- **Step 1 – Define the targeted backdoor.** The attacker chooses a secret trigger $T_\delta(x) := x + \delta$, a secret source class y_{source}, and a secret target class y_{target} as per Definition 3.1.
- **Step 2 – Expert training.** The attacker records the training trajectory $(\theta_k)_{0\le k\le K}$ of a backdoored "expert" model. That model is obtained by ERM on the dataset $D_p(\delta)$, obtained by augmenting the clean dataset D with the explicit poisons $\{\Phi^\delta(x, y), \mid (x, y) \in D\}$, where Φ^δ denotes the backdoor operator.

[1]The GGU abstracts away gradient generation, unifying different learning paradigms. In centralized stochastic optimization, each GGU is a minibatch example, making n_u the minibatch size; in distributed or federated learning, each GGU is a participating worker or client, making n_u the number of workers.

[2]During this work, we take SGD as the update rule.

- **Step 3 – Trajectory matching.** Soft labels $z \in \mathbb{R}^{|D| \times C}$, with $C = |\mathcal{Y}|$, are optimized on *clean images* by solving:

$$\min_{z \in \mathbb{R}^{|D| \times C}} \mathbb{E}_{\substack{k \sim \text{Unif}([K]) \\ (x,y) \sim \mathcal{D}_c}} \Big[L(z, \delta, \theta_k, (x,y)) \Big], \quad L(z, \delta, \theta, (x,y)) := \frac{\|U_p(\delta, \theta, (x,y)) - U_c(z, \theta, x)\|^2}{\|U_p(\delta, \theta, (x,y)) - \theta\|^2}. \tag{4}$$

$$U_c(z, \theta_k, x) = \text{Update}\Big(\theta_k, \text{Mean}\big(\nabla_\theta \ell\big(f_{\theta_k}(x), z\big)\big)\Big),$$
$$U_p(\delta, \theta_k, (x,y)) = \text{Update}\Big(\theta_k, \text{Mean}\big(\nabla_\theta \ell\big(f_{\theta_k}(\Phi^\delta(x,y)_1), \Phi^\delta(x,y)_2\big)\big)\Big).$$

  Here, z represents unnormalized class probabilities over the C classes in $\mathcal{Y}$. This optimization is performed by the attacker, in an optimization friendly centralized setting, with exact gradient computation. As the attacker has no knowledge of the end user's defense mechanism, they assume Mean aggregation (see U_c). The loss L measures the normalized squared distance between U_p, induced by poisoned features and labels, and U_c, induced by clean features and optimized labels, encouraging clean-image label poisoning to reproduce the poisoned update.
- **Step 4 – Label flip selection.** The soft labels are converted into hard label flips by ranking examples according to $s(x_i) = \max_{c \neq y_i} z_{i,c} - z_{i,y_i}$ and selecting the b examples with the largest scores, where b is the poisoning budget. Each selected example is assigned to its most likely incorrect class. The result is the clean-image poisoned dataset D_{flipped}.

The key idea behind FLIP is to replace the direct optimization of the victim's CTA and ASR by matching its training trajectory to that of a backdoored expert model trained on D_p. The attacker therefore searches for a label-only poisoned dataset D_{flipped} whose updates approximate those of the expert, with the goal of inducing the same training dynamics and backdoor. This heuristic is referred to as *trajectory matching*. The resulting dataset contains only b flipped labels; the trigger is never included and therefore never shown to the victim.

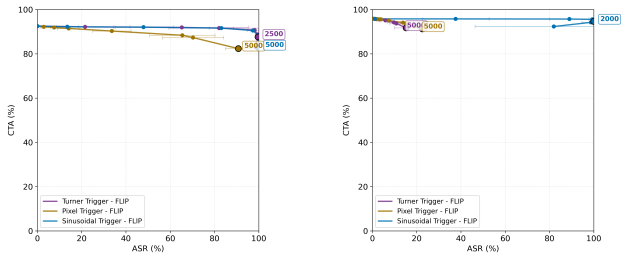
Appendix A summarizes the FLIP algorithm. We report a discrepancy between the algorithm described in Jha et al. (2023), which optimizes labels using a single recorded expert minibatch at each sampled checkpoint, and the released implementation, which instead iterates over all dataset minibatches at each checkpoint. This bears on the efficiency of the method, as the theoretical algorithm requires N steps of optimization, while the companion implementation requires a much larger $N \times \frac{|D|}{B}$ number of steps (with B the batch size).

4.2 Sensitivity of FLIP to dataset, architecture, and robust aggregation

Sensitivity to training dataset. Reproducing Jha et al. (2023) in a centralized setting without a robust aggregation defense (i.e., using mean aggregation) allows us to match their results on CIFAR-10 with a ResNet-32 target architecture (He et al., 2016), as shown in Figure 2a. Label poisoning enables successful backdoor attacks for all proposed triggers (see Appendix D). However, extending this algorithm to the SVHN dataset reveals a first limitation of FLIP. Indeed, Figure 2b shows that only the sinusoidal trigger achieves a successful backdoor attack, while attacks using the other triggers fail. This indicates that, for a fixed trigger, the feasibility of reproducing the backdoor through clean-image poisoning varies across datasets.

Sensitivity to model architecture. Similarly, when evaluating FLIP on different architectures, we observe that it fails for certain triggers. In particular, the ConvNeXtMicro architecture (Appendix C) appears to be largely insensitive to clean-image poisoning attacks based on the Turner trigger, as illustrated in Figure 6. This confirms that the success of the clean-image backdoor attack highly depends on the trigger.

Sensitivity to robust aggregation. Finally, testing FLIP when the end user's training process is defended by robust aggregation rules results in attack failure in most of the configurations we consider. We evaluate FLIP in a federated learning setting with $n_p = 3$ malicious workers and $n_h = 7$ honest workers, using the



(a) CIFAR-10 – ResNet-32 **(b)** SVHN – ResNet-32

Figure 2: FLIP on ResNet-32. CTA–ASR trade-off. Points denote flipping budgets. Successful attacks lie in the top-right region (high ASR with preserved CTA for stealthiness).

Algorithm 1: *JOLT*

Input: iterations N, trigger initialization δ, threshold $\varepsilon > 0$, learning rates η_z, η_δ, dataset D, model batch size B_K

Initialize logits z;
Train initial expert trajectory on $D^p(\delta)$ and save $\{\theta_k(\delta)\}_k$;
for $n = 1$ **to** N **do**
  for $(x, y) \in D$ **do**
    Sample $(k_1, \ldots, k_{B_K}) \sim \mathrm{Unif}[K]$;
    $z \leftarrow z - \frac{\eta_z}{B_K} \nabla_z \sum_{i=1}^{B_K} L(z, \delta, \theta_{k_i}, (x, y))$;
    $\delta \leftarrow \delta - \frac{\eta_\delta}{B_K} \nabla_\delta \sum_{i=1}^{B_K} L(z, \delta, \theta_{k_i}, (x, y))$;
  Retrain expert trajectory on $D^p(\delta)$ and save $\{\theta_k(\delta)\}_k$;
return $\tilde{\mathbf{y}}^p = \mathrm{softmax}(z)$

following aggregation rules: TrimmedMean (Yin et al., 2018), Krum (Blanchard et al., 2017), and MultiKrum. TrimmedMean and MultiKrum are configured with a Byzantine tolerance of $f = 3$, corresponding to the three malicious workers, and aggregate over the remaining 7 gradients (or coordinates). The dataset is evenly partitioned across workers, with each worker receiving an *iid* subset. Malicious workers have an equal number of poisoned samples which sums to the poisoning budget of the attacker.

As shown in Figures 4 and 7 to 9, FLIP fails to plant backdoors against most aggregation rules except TrimmedMean where its success depends on the architecture.

These results highlight a dependency of FLIP's performance on the target dataset, model architecture, and aggregation rule. In particular, the failures observed under robust aggregation indicate that the attack does not directly transfer to these settings. These limitations thus restrict the scope of the claim that "Label Poisoning is All You Need," made by Jha et al. (2023).

5 Joint Optimization of Labels and Trigger (*JOLT*)

Given the results of Section 4.2, we hypothesize that the success of label-only backdoor attacks critically depends on the trigger. From a gradient-based perspective, this dependency follows from the restricted poisoning directions afforded by the attacker's write access. When poisoning a label while keeping the image clean, the attacker selects one of the $|\mathcal{Y}| - 1$ permitted directions. The labels select a point within this constrained set, while the trigger determines the target gradient $\nabla_\theta \ell\big(f_\theta(\Phi^\delta(x, y)_1), \Phi^\delta(x, y)_2\big)$. Thus, a poorly chosen trigger may prevent reproducing the training trajectory of a model trained on $D_p(\delta)$; see Figure 3 for an illustration.

We thus introduce *JOLT* which removes this mismatch by jointly optimizing the trigger and the labels. It incorporates the attacker's write-access constraints directly into the trajectory-matching objective. This replaces Steps 2 and 3 of Section 4.1 by

$$\min_{z \in \mathbb{R}^{|D| \times C}; \delta \in \mathcal{X}} \mathbb{E}_{\substack{k \sim \mathrm{Unif}[K] \\ (x,y) \sim \mathcal{D}_c}} \Big[L(z, \delta, \theta_k, (x, y)) + \lambda_1 \ell\big(f_{\theta_k}(\Phi^\delta(x, y)_1), \Phi^\delta(x, y)_2\big) + \lambda_2 \big(\cos(\delta, \mu_{\text{target}}) + 1\big) \Big], \quad \text{s.t.} \quad \|\delta\|_\infty \le \varepsilon, \qquad \theta_{k+1} = U_p(\delta, \theta_k, (x, y)) \tag{5}$$

where L is the initial trajectory matching loss described in Equation (4). The term $\ell\big(f_{\theta_k}(\Phi^\delta(x, y)_1), \Phi^\delta(x, y)_2\big)$ ensures that trigger optimization enhances the targeted backdoor behavior. $\cos(\delta, \mu_{\text{target}})$ is the cosine similarity between the trigger perturbation and μ_{target}, the mean image of the target class y_{target}. Minimizing this term prevents the emergence of a trivial trigger. The update

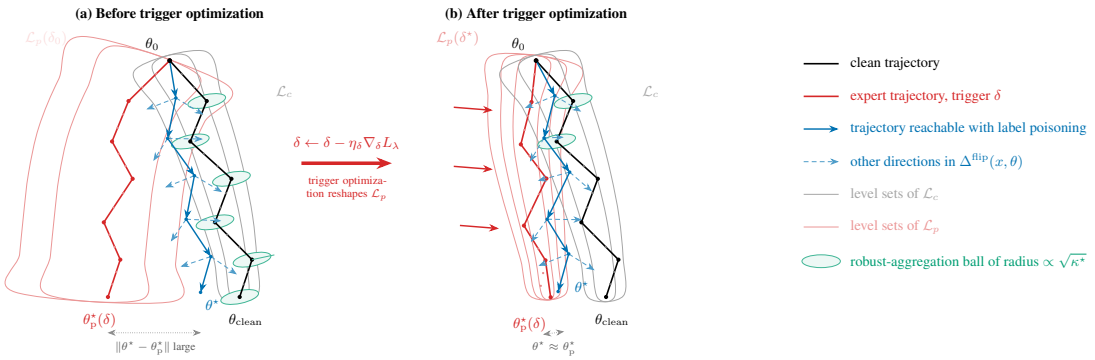


Figure 3: Joint trigger and label trajectory matching to bypass robust aggregation rules without prior knowledge of the aggregation rule. When malicious gradients are in the minority, the aggregated gradient lies within a ball centered at the mean gradient of the honest gradients, with a radius proportional to $\sqrt{\kappa^\star}$ (see Appendix B)

$\theta_{k+1} = U_p(\delta, \theta_k, (x, y))$ highlights that the expert trajectory depends on the trigger δ. The choice of λ_1, λ_2, and how the expert trajectory is retrained is detailed in Appendix E.1.

The optimized trigger steers the target gradient toward the set reachable by label poisoning, while the poisoned labels approximate the target as closely as possible within the attacker's write-access constraints. Labels are then discretized according to the attacker's budget, as described in Step 4 of Section 4.1. Freezing δ recovers the original FLIP objective in Equation (4), since L is the only term depending on z.

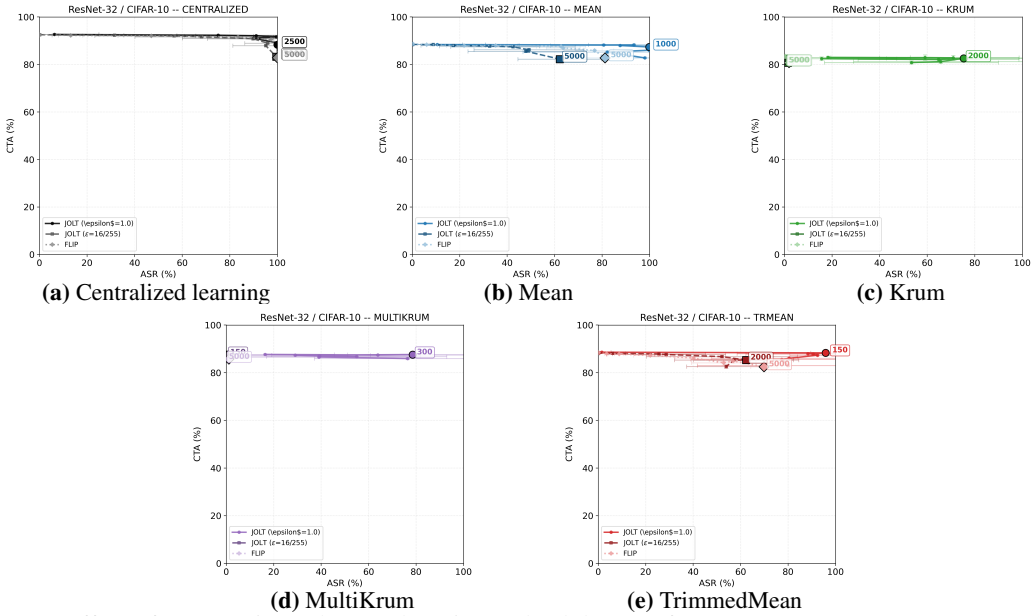
In practice, we solve this problem using Algorithm 1. Unlike Jha et al. (2023); Cazenavette et al. (2022), which load a single expert checkpoint per epoch, we sample a new batch of expert checkpoints for each data batch. This makes explicit that trajectory matching is stochastic gradient descent with simultaneous sampling from the clean dataset distribution $\mathcal{D}_c$ and the training trajectory distribution $(\theta_k)_{0 \le k \le K}$. This better mixes the two sources of randomness and leads to more stable optimization (Figure 10).

6 Experimental Results

Both FLIP and *JOLT* generate poisoned labels from expert trajectories constructed in a centralized, undefended setting. In FLIP, the trigger δ is fixed while the labels are optimized; in *JOLT*, the trigger and labels are optimized jointly, and the expert trajectory is periodically retrained as δ changes. We then evaluate the resulting label-only poisoning against aggregation rules unseen during optimization. Our hypothesis is that jointly optimizing the trigger and labels allows *JOLT* to steer the attack target toward a region reachable by label flips alone, while remaining sufficiently close to honest gradient directions to remain effective under robust aggregation.

6.1 Experimental Setting

Our experiments with *JOLT* follow the setup described in Section 4.2. We deliberately restrict our study to this controlled federated setting to isolate whether robust aggregation alone can prevent a label-only clean-image backdoor. We evaluate both an high-visibility trigger ($\varepsilon = 1$, which only enforces the normalized pixel range) and a low-visibility trigger ($\varepsilon = 16/255$, a standard perturbation constraint). As baselines, we use FLIP (Jha et al., 2023) with the sinusoidal trigger used to initialize *JOLT*, as well as GCB (Xu et al., 2026), a recent clean-image poisoning attack. Full experimental details are provided in Appendix E.1.



(a) Centralized learning **(b)** Mean **(c)** Krum

(d) MultiKrum **(e)** TrimmedMean

Figure 4: Effect of aggregation rules on clean-image backdoor performance (CIFAR-10, ResNet-32). CTA-ASR trade-off per aggregation rule.

During all experiments, we examine the CTA–ASR trade-off as the poisoning budget increases, focusing on the comparison between *JOLT* and the baselines, transfer across aggregation rules, the effect of the trigger constraint ε, and variation across datasets and architectures.

6.2 Empirical results

There are two main challenges for the attacker in our setting. The first arises from the attacker's restricted write access and motivates the joint optimization of labels and trigger. The second stems from robust aggregation, which bounds the deviation between the aggregated gradient and the mean gradient of honest workers (Bareilles et al., 2026) (see Appendix B and Figure 3).

For $\varepsilon = 1$, *JOLT* successfully backdoors a large majority of the considered settings. For example, on CIFAR-10 with ResNet-32 (Table 2), *JOLT* achieves 95.8% ASR under TrimmedMean with only 0.3% poisoned labels, compared with 3.6% for FLIP and 2.3% for GCB at the same poisoning budget. Under Mean aggregation, *JOLT* reaches 99.9% ASR with 4% poisoned labels. On SVHN with ResNet-32 (Table 3), it reaches 100% ASR under both MultiKrum and Mean with only 0.2% and 0.3% poisoned labels respectively. With ConvNeXt-Micro, *JOLT* similarly achieves 100% ASR under TrimmedMean on SVHN with a 0.2% poisoning budget (Table 5). Nevertheless, success is neither uniform nor monotonic with the poisoning budget: *JOLT* reaches only 69.0% and 36.8% ASR under Krum and MultiKrum on CIFAR-10/ConvNeXt-Micro (Table 4), even at 10% poisoning, while ASR remains highly variable and drops substantially with low-visibility triggers (Section E.2.2).

What robust aggregation failures reveal about *JOLT*'s attack structure. Robust aggregation constrains and reduces the influence of malicious gradients: when malicious gradients are in the minority, the aggregated update remains within a $\kappa^\star$-controlled neighborhood of the mean honest gradient (see Appendix B and Figure 3). The success of *JOLT* at very low poisoning budgets (100% ASR under MultiKrum with 0.2%

poisoned labels) suggests that only a small number of deviations can suffice to steer the trajectory toward a backdoored region. In this sense, only a few *jolts* are needed. On CIFAR-10/ResNet-32 (Table 2), *JOLT* successfully induces a backdoor while causing at most a 1 percentage-point drop in CTA. This suggests that these deviations need not dominate training: honest updates can continue to drive clean performance while malicious deviations steer the trajectory toward the backdoor.

Moreover, *JOLT* produces poisoned labels without knowledge of the defender's aggregation rule, yet the resulting poisons successfully plant backdoors under aggregation rules unseen during optimization. This suggests that joint trigger and label optimization searches for backdoors satisfying two constraints simultaneously: reachability through label-only poisoning and compatibility with the neighborhood of honest updates imposed by robust aggregation. The trigger therefore not only makes the backdoor reachable through label flips, but also shapes the backdoor objective and the deviations required to induce it as illustrated by the 95.8% ASR of *JOLT* versus 3.6% for FLIP and 2.3% for GCB at 0.3% poisoning under TrimmedMean (as illustrated in Figure 3).

Overall, these results show that robust aggregation does not eliminate the clean-image poisoning attack surface: successful attacks require backdoor objectives that are both reachable through label-only poisoning and compatible with the constraints imposed by robust aggregation.

7 Concluding remarks

We introduce *JOLT*, a clean-image backdoor attack that jointly optimizes poisoned labels and the backdoor trigger. By adapting the backdoor objective to the attacker's restricted write access, *JOLT* searches for targeted training trajectories reachable through clean-image poisoning. It transfers to federated training with robust aggregation without prior knowledge of the aggregation rule.

Our results show that robust aggregation alone does not prevent clean-image backdoors. While label-only access limits the attacker's control over training, *JOLT* identifies backdoor objectives whose required deviations cannot be systematically filtered by robust aggregation. To the best of our knowledge, this is the first demonstration that label-only clean-image poisoning can successfully plant backdoors under robust aggregation with limited poisoning budgets.

Our study has two main limitations. First, we consider a controlled federated setting with synchronous training and fixed numbers of honest and malicious workers. Future work should evaluate *JOLT* under more realistic conditions, including varying numbers of malicious participants and partial participation. Second, our evaluation covers only two image-classification datasets and two architectures. Extending it to larger-scale datasets and models would help characterize how the attack surface evolves with model and task complexity.

AI use statement

In this work, we used generative AI tools as assistants with implementing methods, specifically to improve adherence to established coding standards. We have not used generative AI tools to generate synthetic datasets, develop conceptual frameworks, formulate mathematical claims, design research methodology or experiments, or interpret results.

Additionally, we used generative AI tools as a writing aid to improve the clarity, readability, and presentation of the manuscript. We reviewed all AI-assisted text and code, and all generated code was manually checked and tested. Generative AI tools were not used for research ideation, methodological design, or experimental analysis. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

Ethics statement

Our work contributes to an understanding of the capabilities of label poisoning attacks to induce backdoors in robust distributed learning systems, in the presence of aggregation mechanisms such as the Mean, Trimmed Mean, Krum, or MultiKrum, which are widely used in federated and distributed settings. Such analysis can improve the reliability and robustness of large-scale learning algorithms, particularly in adversarial or noisy environments.

At the same time, the techniques studied in this work could be misused for harmful purposes, as they provide insights into how such attacks can be designed and optimized. However, as is common in security research that exposes vulnerabilities, we believe it is in the public interest to share these findings. A clearer understanding of attack mechanisms is essential for developing effective defenses and improving the security of distributed learning systems.

Reproducibility statement

The paper includes the learning configuration (number of malicious and honest workers), datasets (CIFAR-10 and SVHN), model architectures (ResNet-32 and ConvNeXtMicro), aggregation rules (Mean, Krum, MultiKrum, and TrimmedMean), and all training hyperparameters (optimizer, learning rates, weight decay, batch sizes, and learning rate schedules). We also fully describe the attack procedure, including label poisoning settings, repetition protocol, and trigger optimization details (bilevel optimization steps, initialization, optimizer, and constraints).

References

El-Mahdi El-Mhamdi, Sadegh Farhadkhani, Rachid Guerraoui, Nirupam Gupta, Lê-Nguyên Hoang, Rafael Pinot, Sébastien Rouault, and John Stephan. On the impossible safety of large ai models. *arXiv preprint arXiv:2209.15259*, 2022.

El-Mahdi El-Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*, 2018. URL https://arxiv.org/abs/1802.07927.

Gilad Baruch, Moran Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Peva Blanchard, El-Mahdi El-Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in neural information processing systems*, 30, 2017.

Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International conference on machine learning*, pages 5650–5659. Pmlr, 2018.

Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *IEEE Transactions on Signal Processing*, 70:1142–1154, 2022.

Wassim Bouaziz, El-Mahdi El-Mhamdi, and Nicolas Usunier. Inverting gradient attacks makes powerful data poisoning. *arXiv preprint arXiv:2410.21453*, 2024.

Abdessamad El-Kabid and El-Mahdi El-Mhamdi. Approaching the harm of gradient attacks while only flipping labels. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=eAqBcljvsU.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain, 2019. URL https://arxiv.org/abs/1708.06733.

Jonas Geiping, Liam Fowl, W Ronny Huang, Wojciech Czaja, Gavin Taylor, Michael Moeller, and Tom Goldstein. Witches' brew: Industrial scale data poisoning via gradient matching. *arXiv preprint arXiv:2009.02276*, 2020.

Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. Collecting image annotations using amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, CSLDAMT '10, page 139–147, USA, 2010. Association for Computational Linguistics.

Peter Welinder, Steve Branson, Serge Belongie, and Pietro Perona. The multidimensional wisdom of crowds. In *Proceedings of the 24th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'10, page 2424–2432, Red Hook, NY, USA, 2010. Curran Associates Inc.

Srikanth Jagabathula, Lakshminarayanan Subramanian, and Ashwin Venkataraman. Identifying unreliable and adversarial workers in crowdsourced labeling tasks. *Journal of Machine Learning Research*, 18(93): 1–67, 2017. URL http://jmlr.org/papers/v18/15-650.html.

Rishi Jha, Jonathan Hayase, and Sewoong Oh. Label poisoning is all you need. *Advances in Neural Information Processing Systems*, 36:71029–71052, 2023.

Binyan Xu, Fan Yang, Di Tang, Xilin Dai, and Kehuan Zhang. Breaking the stealth-potency trade-off in clean-image backdoors with generative trigger optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 27197–27205, 2026.

Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation. In *Uncertainty in artificial intelligence*, pages 261–270. PMLR, 2020.

Youssef Allouah, Sadegh Farhadkhani, Rachid Guerraoui, Nirupam Gupta, Rafaël Pinot, and John Stephan. Fixing by mixing: A recipe for optimal byzantine ml under heterogeneity. In *International Conference on Artificial Intelligence and Statistics*, pages 1232–1300. PMLR, 2023.

Virat Shejwalkar, Amir Houmansadr, Peter Kairouz, and Daniel Ramage. Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning. In *2022 IEEE symposium on security and privacy (SP)*, pages 1354–1371. IEEE, 2022.

Luis Muñoz-González, Bjarne Pfitzner, Matteo Russo, Javier Carnerero-Cano, and Emil C Lupu. Poisoning attacks with generative adversarial nets. *arXiv preprint arXiv:1906.07773*, 2019.

Bingyin Zhao and Yingjie Lao. Clpa: Clean-label poisoning availability attacks using generative adversarial nets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9162–9170, 2022.

Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 27–38, 2017.

Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. *Advances in neural information processing systems*, 31, 2018.

Yinghua Gao, Yiming Li, Linghui Zhu, Dongxian Wu, Yong Jiang, and Shu-Tao Xia. Not all samples are born equal: Towards effective clean-label backdoor attacks. *Pattern Recognition*, 139:109512, 2023.

Kangjie Chen, Xiaoxuan Lou, Guowen Xu, Jiwei Li, and Tianwei Zhang. Clean-image backdoor: Attacking multi-label models with poisoned labels only. In *The eleventh international conference on learning representations*, 2022.

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

Khoa Doan, Yingjie Lao, Weijie Zhao, and Ping Li. Lira: Learnable, imperceptible and robust backdoor attacks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11966–11976, 2021.

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

Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. *Advances in neural information processing systems*, 33:16070–16084, 2020.

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

George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4750–4759, 2022.

Qixuan Jin, Walter Gerych, Abinitha Gourabathina, Vinith Menon Suriyakumar, and Marzyeh Ghassemi. TOGA: Trigger optimization for clean data ordering backdoor attack, 2026. URL https://openreview.net/forum?id=NbyhwSvOgE.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Gilles Bareilles, Wassim Bouaziz, Julien Fageot, and El-Mahdi El-Mhamdi. Brief announcement: Byzantine machine learning, multikrum and an optimal notion of robustness. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, PODC '26, page 220–223, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400725128. doi: 10.1145/3796701.3815914. URL https://doi.org/10.1145/3796701.3815914.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022.

El-Mahdi El-Mhamdi, Rachid Guerraoui, and Sébastien Rouault. Distributed momentum for byzantine-resilient learning. *arXiv preprint arXiv:2003.00010*, 2020.

Supplementary material

OUTLINE OF THE APPENDIX

A Described vs. implemented FLIP algorithm **13**

B Robust aggregation constraints on honest gradient deviations **13**

C ConvNeXtMicro **14**

D Triggers **14**

E Experimental results **15**

E.1 Experimental setup 15

E.2 Results and discussion 16

F Stabilizing trajectory matching **25**

A DESCRIBED VS. IMPLEMENTED FLIP ALGORITHM

In Jha et al. (2023), a succinct algorithm is proposed in which pairs of expert minibatches are sampled and the labels are optimized using a single batch at each sampled checkpoint (see Algorithm 2). The implementation (and our reproduction), however, uses a different variant that (i) loads a single checkpoint θ_k and then (ii) iterates over *all minibatches* of the dataset to optimize the labels (see Algorithm 3).

The described algorithm only matches the gradient induced by the recorded minibatch (x_k, y_k), which implicitly assumes that the victim trains with the expert's seed, whereas the implementation approximates the expected gradient at θ_k over the whole dataset. This removes the seed assumption but multiplies the cost of each outer iteration by $|D|/B$ (with B the batch size).

Algorithm 2: Described FLIP

Input: iterations N, expert checkpoints $(\theta_k, (x_k, y_k))$, learning rate η_z, dataset D

Initialize logits z; **for** $n = 1$ **to** N **do**

  Sample $k \sim \mathrm{Unif}[K]$;

  $z \leftarrow z - \eta_z \nabla_z L(z, \delta, \theta_k, (x_k, y_k))$;

return $\tilde{\mathbf{y}}^p = \mathrm{softmax}(z)$

Algorithm 3: Implemented FLIP

Input: iterations N, expert checkpoints θ_k, learning rate η_z, dataset D

Initialize logits z; **for** $n = 1$ **to** N **do**

  Sample $k \sim \mathrm{Unif}[K]$; **for** $(x, y) \in D$ **do**

    $z \leftarrow z - \eta_z \nabla_z L(z, \delta, \theta_k, (x, y))$;

return $\tilde{\mathbf{y}}^p = \mathrm{softmax}(z)$

B ROBUST AGGREGATION CONSTRAINTS ON HONEST GRADIENT DEVIATIONS

In this section we recall the robustness constraint provided by robust aggregation bounding the deviation of the aggregated gradient from the mean of the honest gradients.

Definition B.1 (Robustness coefficient Bareilles et al. (2026); Allouah et al. (2023))**.** *Let $f < n_u/2$, and* $\mathrm{Agg} : (\mathbb{R}^d)^{n_u} \to \mathbb{R}^d$ *be an aggregation rule. The* robustness coefficient *of* Agg *is*

$$\kappa^\star(\mathrm{Agg}) \;:=\; \sup_{\substack{g_1,\ldots,g_{n_u}\in\mathbb{R}^d \\ S\in\mathcal{P}_{n_u}^{n_u-f} \\ \sum_{i\in S}\|g_i-\bar{g}_S\|^2>0}} \frac{\|\mathrm{Agg}(g_1,\ldots,g_{n_u})-\bar{g}_S\|^2}{\frac{1}{|S|}\sum_{i\in S}\|g_i-\bar{g}_S\|^2},$$

where $\bar{g}_S = \frac{1}{|S|}\sum_{i\in S} g_i$ and $\mathcal{P}_{n_u}^{n_u-f}$ denotes the collection of all subsets of $\{1,\ldots,n_u\}$ of size $n_u - f$. We say that Agg *is* f-robust *when $\kappa^\star(\mathrm{Agg}) < \infty$.*

A direct consequence of the above definition is that, for any $g_1,\ldots,g_{n_u} \in \mathbb{R}^d$ and any $S \in \mathcal{P}_{n_u}^{n_u-f}$, there holds

$$\|\mathrm{Agg}(g_1,\ldots,g_{n_u})-\bar{g}_S\|^2 \le \kappa^\star(\mathrm{Agg})\frac{1}{|S|}\sum_{i\in S}\|g_i-\bar{g}_S\|^2 . \tag{6}$$

Therefore, the deviation of the aggregated gradient from the mean of the gradients in S is bounded by $\kappa^\star(\mathrm{Agg})$ times their average squared deviation from this mean. In particular, the aggregated gradient lies within a ball centered at $\bar{g}_S$, with squared radius proportional to $\kappa^\star(\mathrm{Agg})$ and to the dispersion of the gradients in S.

C ConvNeXtMicro

We use *ConvNeXtMicro*, a lightweight ConvNeXt-style architecture Liu et al. (2022) designed for 32×32 inputs. The network consists of four stages with channel dimensions $\{32, 64, 96, 128\}$ and one block per stage. Downsampling is performed via strided convolutions between stages.

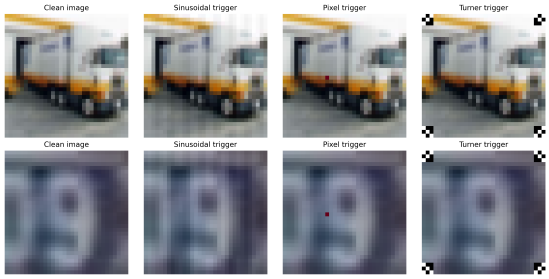
Each block follows the ConvNeXt design: a 7×7 depthwise convolution, LayerNorm (channel-last), and a two-layer MLP with expansion ratio 4 and GELU activation, combined with a residual connection.

The stem is adapted for CIFAR-10 with a 3×3 stride-1 convolution. The final representation is obtained via global average pooling, followed by LayerNorm and a linear classifier.

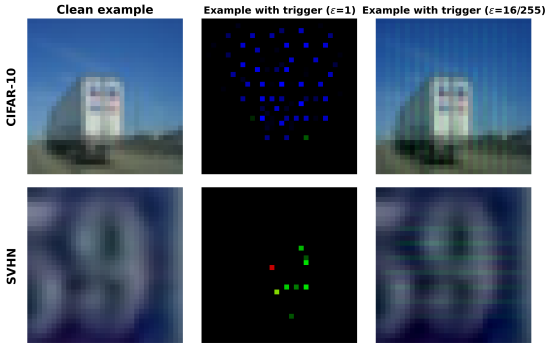
The model has $\sim$0.3M parameters, making it comparable to ResNet-32 while using more modern design choices.

D Triggers

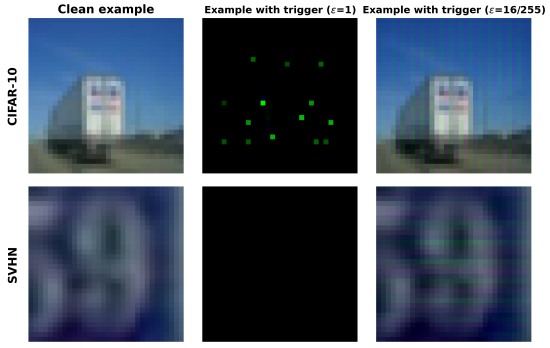
In Jha et al. (2023), the authors originally considered only three types of secret triggers (Figure 5a): the pixel trigger, which modifies a single pixel by changing its color; the sinusoidal trigger, which adds sinusoidal noise to the original image; and the Turner trigger, which overlays a specific pattern on the corners of the image. We show in Figures 5b and 5c triggers obtained with *JOLT*.



(a) Examples with classical triggers.



(b) Triggers obtained with *JOLT* for ResNet-32.



(c) Triggers obtained with *JOLT* for ConvNeXtMicro.

Figure 5: Examples of triggers on CIFAR-10 (top row) and SVHN (bottom row).

E EXPERIMENTAL RESULTS

E.1 EXPERIMENTAL SETUP

Labels optimization All attacks were repeated 10 times to estimate the variance across runs, using the 9th class as the source class and the 4th class as the target class. We used a single expert trajectory trained for 20 epochs, while the user model was trained for 200 epochs. All models were optimized using SGD

with momentum 0.9 and Nesterov acceleration. A step learning rate schedule with decay factor $\gamma = 0.1$ was applied, corresponding to a division of the learning rate by 10 at each milestone.

Both ResNet-32 and ConvNeXtMicro use a learning rate of 0.1 and weight decay of 2×10^{-4}, with learning rate drops by a factor of 10 at epochs 75 and 125.

Both the training of the expert and victim models, as well as label generation, were conducted with a batch size of 256.

Labels were optimized using SGD with momentum 0.5, a learning rate of 150 as suggested by Jha et al. (2023).

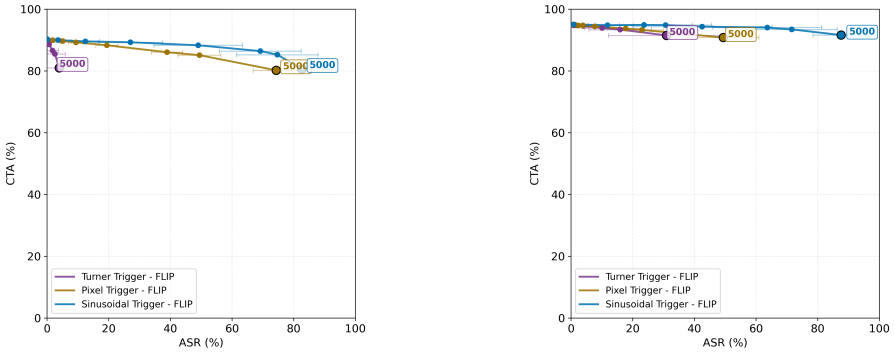
Trigger optimization The trigger δ was initialized as the sinusoidal trigger (Figure 5a) and optimized jointly with the labels for 25 outer iterations, each retraining the expert from scratch. The perturbation was constrained via an ℓ_∞ penalty term (rather than hard clamping), with all regularization hyperparameters $\lambda_1 = \lambda_2 = 1.0$. δ was optimized with SGD (learning rate 10^{-2}) under an ℓ_∞ budget of $\epsilon = 1.0$ for the *baseline* configuration and $\epsilon = 16/255$ for the low-visibility configuration, both evaluated over the same 10 seeds, source/target label pair, and model/dataset combinations as the labels-only setting above. As in the labels-optimization step, the trigger was generated under a single-attacker, single-worker split ($n_h = 0$ honest, $n_p = 1$ malicious) and subsequently deployed under both the single-user and $n_h = 7$;$n_p = 3$ federated setting. For robust aggregators, we set the assumed number of Byzantine workers to $f = n_p$.

All experiments were conducted on NVIDIA RTX 6000, for a total computational cost of approximately 1900 GPU hours

Implementation We build upon the official FLIP repository Jha et al. (2023), which we fork and extend with our contributions. For the aggregation rules, we use the implementation provided by El-Mhamdi et al. (2020). For GCB, we follow the training configuration recommended by its authors.

E.2 Results and discussion

E.2.1 FLIP's performance in centralized learning setting with Mean aggregation



(a) CIFAR-10 - ConvNeXtMicro **(b)** SVHN - ConvNeXtMicro

Figure 6: Comparison of FLIP's performance across datasets and triggers for ConvNeXtMicro.

Table 1: Comparison of FLIP's performance across triggers, datasets and models. Results are reported as CTA/ASR (in %) with standard deviation. Bold and underlined values indicate the highest and second-highest ASR.

Budget	CIFAR-10		SVHN	
	ConvNeXtMicro	ResNet-32	ConvNeXtMicro	ResNet-32
Turner Trigger – FLIP				
0	90.2±0.2/0.1±0.1	92.5±0.2/0.0±0.0	94.8±0.3/0.4±0.2	95.9±0.1/0.3±0.2
150	89.8±0.2/0.1±0.1	92.2±0.3/21.6±20.2	95.0±0.3/0.6±0.4	95.7±0.1/1.1±0.4
300	89.8±0.3/0.1±0.1	91.9±0.2/65.2±30.1	94.7±0.2/0.9±0.4	95.6±0.1/1.9±0.9
500	89.5±0.2/0.1±0.1	91.6±0.2/82.0±22.7	94.7±0.4/1.7±1.2	95.5±0.1/3.3±1.2
1000	88.6±0.2/0.7±0.5	90.7±0.4/98.1±3.3	94.5±0.3/4.0±1.7	95.1±0.1/5.9±1.9
2000	86.6±0.3/1.8±2.0	**88.7±0.3/99.6±1.2**	93.9±0.4/10.1±5.6	94.2±0.3/9.7±3.0
2500	85.5±0.3/2.6±3.4	87.6±0.3/99.6±0.8	93.3±0.5/15.9±10.2	93.7±0.5/10.8±2.9
5000	**81.0±0.5/3.9±5.8**	83.0±0.5/98.6±2.8	**91.5±1.0/31.0±18.9**	**91.7±0.7/15.2±5.3**
Pixel Trigger – FLIP				
0	90.1±0.3/0.0±0.0	92.5±0.2/0.0±0.1	94.9±0.2/0.5±0.1	95.8±0.1/0.5±0.1
150	89.9±0.2/1.8±0.6	92.2±0.2/2.9±1.1	94.9±0.2/1.2±0.6	95.8±0.1/1.5±0.6
300	89.6±0.3/5.0±0.8	91.9±0.2/7.6±1.7	94.7±0.4/2.3±1.2	95.7±0.1/2.3±0.8
500	89.2±0.3/9.4±2.4	91.4±0.2/14.1±5.1	94.7±0.3/3.9±1.5	95.6±0.1/3.8±1.0
1000	88.3±0.2/19.4±3.6	90.3±0.2/33.6±8.7	94.4±0.4/7.7±2.8	95.0±0.1/7.4±1.2
2000	86.0±0.4/38.9±5.1	88.4±0.3/65.5±14.8	93.7±0.4/17.7±6.0	94.1±0.2/13.9±2.3
2500	85.1±0.4/49.4±7.0	87.3±0.3/70.3±13.9	93.2±0.4/23.1±7.0	93.5±0.2/15.2±2.8
5000	**80.2±0.4/74.3±7.5**	**82.3±0.2/90.7±5.9**	**90.8±0.7/49.4±11.6**	**91.3±0.3/22.6±5.0**
Sinusoidal Trigger – FLIP				
0	90.2±0.3/0.2±0.1	92.5±0.2/0.1±0.1	95.0±0.2/0.8±0.4	95.8±0.1/1.1±0.6
150	90.0±0.2/3.6±0.8	92.2±0.2/13.5±5.0	94.8±0.2/11.8±8.4	95.8±0.1/37.6±15.1
300	89.5±0.2/12.5±4.6	92.0±0.1/47.9±13.5	94.9±0.2/23.7±15.6	95.7±0.1/89.0±8.9
500	89.2±0.2/27.1±10.5	91.6±0.1/83.0±5.4	94.8±0.2/30.7±14.9	95.6±0.1/99.3±0.7
1000	88.3±0.4/49.0±14.4	90.5±0.2/97.3±1.2	94.3±0.3/42.5±22.7	**95.3±0.1/99.9±0.1**
2000	86.4±0.3/69.2±13.3	88.4±0.2/99.6±0.3	94.0±0.2/63.7±17.7	94.7±0.2/99.9±0.2
2500	85.2±0.2/74.7±13.2	87.7±0.3/99.8±0.2	93.4±0.4/71.6±14.8	94.2±0.1/98.7±4.0
5000	**80.6±0.3/82.6±9.5**	**82.8±0.3/99.9±0.1**	**91.6±0.8/87.6±9.1**	92.3±0.4/81.9±35.6

E.2.2 Additional discussion

Non-monotonicity of ASR with respect to the poisoning budget. We observe that ASR does not consistently increase with the poisoning budget. However, the substantial variance across runs makes it difficult to attribute these fluctuations to systematic changes in the training trajectory. For instance, on CIFAR-10/ResNet-32 with Mean aggregation, ASR ranges from 82.2% to 99.9% across poisoning budgets between 0.6% and 10%, with intermediate budgets exhibiting both increases and decreases.

Attack failures. Despite succeeding in most configurations, *JOLT* fails to induce a near-complete backdoor in some settings. On CIFAR-10 with ConvNeXt-Micro, it reaches only 69% and 36.8% ASR under Krum and MultiKrum, respectively, even with a 10% poisoning budget. These results show that the attack does not universally transfer across robust aggregation rules. We hypothesize that these failures arise when the backdoor objective is difficult to reach through label poisoning while satisfying the constraints imposed by the aggregation rule (Figure 3).

Nevertheless, *JOLT* substantially outperforms existing clean-image poisoning attacks under the same aggregators: at the same budget, FLIP reaches only 1.8% and 1.1% ASR under Krum and MultiKrum, while GCB reaches 0.6% and 1.7%, respectively. Thus, even when *JOLT* does not achieve a near-complete backdoor, joint trigger and label optimization substantially improves attack effectiveness over the baselines.

Low-visibility triggers. Imposing a tighter trigger constraint, $|\delta|_\infty \leq 16/255$, makes the attack more challenging, as expected, but does not eliminate the advantage of joint optimization. On CIFAR-10/ResNet-32, constrained *JOLT* reaches 62.3% ASR under TrimmedMean with 2000 poisoned labels, compared with 52.9% for FLIP and 49.0% for GCB at the same budget, while preserving 85.3% CTA. It also reaches up to 99.5% ASR in the centralized setting and 62.2% under Mean.

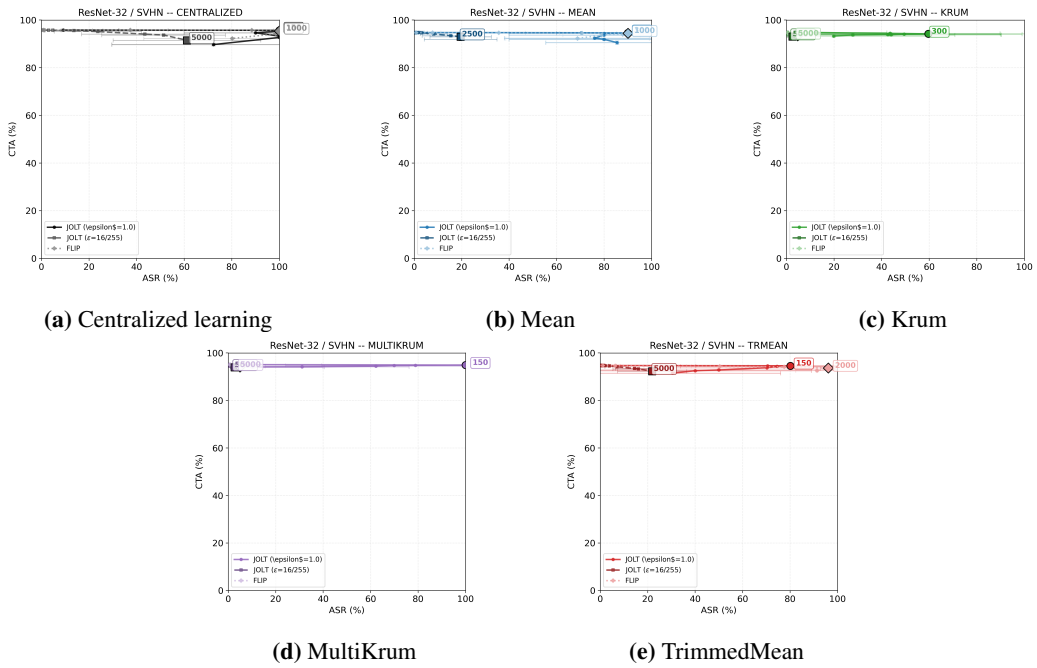
The strongest robust aggregation rules remain difficult to attack in this trigger-constrained regime. On CIFAR-10/ResNet-32, *JOLT* reaches at most 7.9%, 1.5%, and 1.7% ASR under Coordinate-Wise Median, Krum, and MultiKrum, respectively. This difficulty is not specific to *JOLT*: FLIP reaches at most 10.1%, 1.9%, and 1.1%, while GCB reaches 15.8%, 1.8%, and 7.1% under the same rules. These results highlight trigger constraints as an important challenge for clean-image poisoning and show that constrained *JOLT* remains competitive with existing attacks, outperforming them in several defended configurations.

High variance in attack success rate. Across repeated runs, *JOLT* exhibits substantial variance in ASR. Across the tested settings, the average ASR standard deviation is about 39.5 percentage points for ResNet-32 on SVHN. This variability is consistent with the constrained nature of the attack: poisoned gradients only perturb the optimization trajectory within the constraints imposed by the attacker's write access and the aggregation rule. As a result, different runs may follow trajectories that reach different regions of the loss landscape and do not establish the targeted backdoor.

We hypothesize that this sensitivity reflects the geometry of the targeted backdoor regions. If these regions are narrow, small differences in the poisoned gradients may cause the trajectory to leave them before a backdoor is established. This may help explain the dependence of attack success on the dataset and architecture, suggesting that the geometry of targeted regions is an important component of the clean-image poisoning attack surface. Interestingly, we observe lower ASR variance with a constrained trigger ($\varepsilon = 16/255$), which is consistent with the hypothesis that part of the instability originates from the optimization landscape. Stabilizing this optimization remains an important direction for future work.

Table 2: Impact of poisoning budget on ASR and CTA across robust aggregation rules and the centralized (undefended, single-victim) deployment on CIFAR-10 with a ResNet-32 model. Results are reported as CTA/ASR (in %) with standard deviation. Bold and underlined values indicate the highest and second-highest ASR.

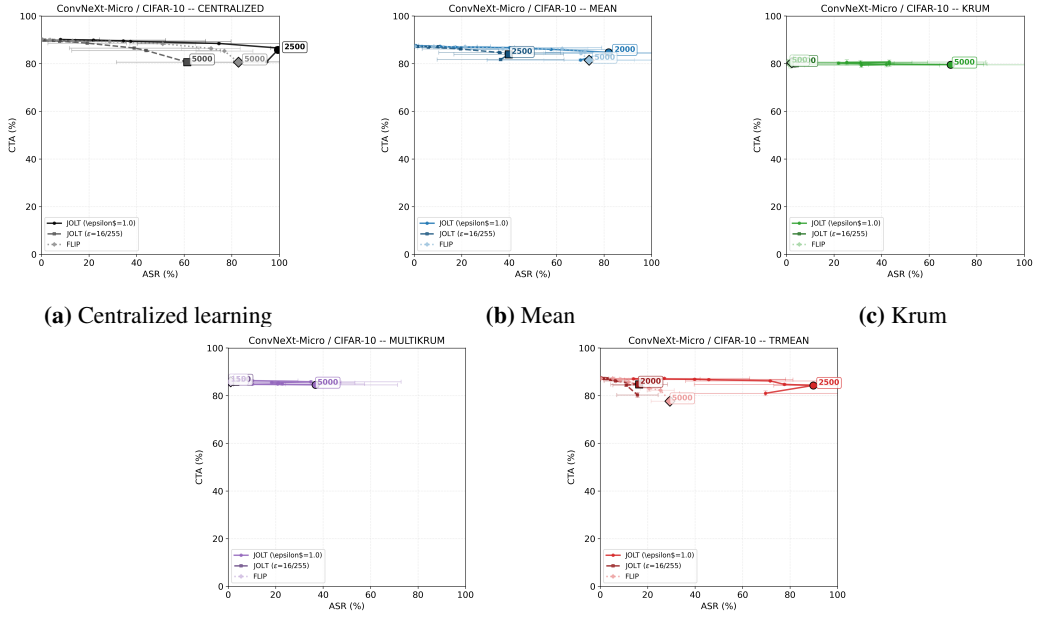
Budget	CENTRALIZED	MEAN	KRUM	TRMEAN	MULTIKRUM
			Optimized Trigger ($\varepsilon = 1.0$) **–** ***JOLT***		
0	92.6±0.2/6.3±18.7	88.4±0.2/10.6±20.3	83.0±0.5/18.3±34.6	88.6±0.2/1.5±3.9	87.7±0.2/16.4±29.2
150	92.3±0.2/75.2±39.7	88.2±0.3/80.7±35.6	82.8±1.0/71.0±30.9	**88.3±0.2/95.8±6.2**	87.4±0.2/63.9±44.6
300	92.1±0.2/91.1±25.3	88.2±0.3/93.4±19.2	82.9±1.1/59.1±39.5	88.0±0.3/88.4±29.8	**87.6±0.3/78.6±27.4**
500	91.6±0.3/99.5±1.0	88.0±0.2/87.6±29.2	82.4±0.7/15.7±31.6	87.8±0.3/90.5±28.3	87.3±0.3/29.1±31.8
1000	91.0±0.2/90.0±29.9	**87.3±0.2/99.9±0.3**	82.1±0.7/64.8±33.8	87.3±0.2/92.4±11.6	87.0±0.2/39.3±42.6
2000	89.1±0.3/99.7±0.8	86.0±0.3/99.9±0.3	**82.6±0.5/75.4±32.5**	86.0±0.2/80.4±32.6	86.7±0.1/55.0±38.0
2500	**88.3±0.2/100.0±0.0**	85.2±0.4/82.2±35.5	81.2±0.7/65.6±36.5	85.6±0.2/77.3±38.3	86.6±0.3/39.1±38.1
5000	83.7±0.4/100.0±0.0	82.7±0.3/98.0±4.7	80.8±0.6/53.4±36.5	83.0±0.4/80.0±38.2	85.9±0.4/76.5±39.3
			Optimized Trigger ($\varepsilon = 16/255$) **–** ***JOLT***		
0	92.5±0.3/0.2±0.1	88.4±0.2/0.2±0.2	83.2±0.9/0.4±0.3	88.3±0.3/0.3±0.2	87.7±0.2/0.4±0.5
150	92.3±0.2/31.6±26.2	88.4±0.1/8.8±12.4	82.7±0.9/0.9±0.9	88.1±0.2/6.1±5.4	**87.6±0.2/1.7±2.3**
300	92.0±0.2/56.6±33.3	88.0±0.2/17.4±13.7	82.3±1.1/0.8±0.7	87.8±0.4/25.6±19.5	87.3±0.3/0.6±0.6
500	91.6±0.2/68.2±27.9	87.7±0.3/32.6±21.1	82.2±0.5/0.8±0.6	87.6±0.3/28.7±25.0	87.3±0.2/0.6±0.7
1000	90.8±0.3/89.8±19.2	87.4±0.3/42.5±26.2	82.2±0.6/0.8±0.6	86.7±0.3/52.2±31.8	87.1±0.3/0.3±0.2
2000	88.9±0.1/95.9±9.6	86.2±0.2/49.2±23.2	81.9±0.7/1.2±1.1	**85.3±0.6/62.3±22.2**	86.6±0.2/0.8±0.6
2500	87.9±0.2/95.1±13.9	85.5±0.4/47.9±24.4	81.8±0.8/1.1±0.9	85.3±0.2/57.2±25.1	86.7±0.3/1.0±0.9
5000	**83.3±0.3/99.5±1.1**	**82.2±0.6/62.2±17.6**	**80.7±0.4/1.5±1.3**	82.6±0.7/54.0±16.7	85.8±0.3/1.3±1.1
			Sinusoidal Trigger – FLIP		
0	92.5±0.2/0.1±0.1	88.6±0.3/0.3±0.1	83.5±0.5/0.3±0.2	88.4±0.3/0.2±0.1	87.7±0.2/0.2±0.1
150	92.2±0.2/13.1±5.1	88.3±0.3/6.1±1.7	82.5±1.0/0.5±0.2	88.0±0.2/3.6±0.9	87.5±0.3/1.0±1.0
300	92.0±0.1/47.0±13.9	87.9±0.4/21.9±7.4	82.7±0.8/0.4±0.2	87.8±0.2/8.8±2.0	87.4±0.1/0.3±0.2
500	91.6±0.1/83.5±5.5	88.0±0.3/35.2±9.2	82.0±0.7/1.0±0.4	87.2±0.3/22.0±7.0	87.1±0.3/0.3±0.2
1000	90.5±0.2/97.3±1.2	87.5±0.3/62.7±10.4	81.9±1.2/0.7±0.3	86.1±0.4/40.1±7.7	87.1±0.3/0.2±0.2
2000	88.4±0.2/99.6±0.3	87.0±0.4/63.7±7.1	81.6±0.5/0.8±0.5	84.1±1.1/52.9±13.6	86.7±0.4/0.6±0.2
2500	87.7±0.3/99.8±0.2	85.9±0.5/76.8±8.1	81.2±0.4/1.1±0.3	84.9±0.9/56.9±6.7	86.8±0.3/0.7±0.3
5000	**82.8±0.3/99.9±0.1**	**82.8±0.9/81.2±9.2**	**80.8±0.8/1.9±0.5**	**82.5±0.6/69.9±5.6**	**85.7±0.4/1.1±0.4**
			GCB		
0	$82.4 \pm 0.2/0.6 \pm 0.5$	$68.3 \pm 1.2/1.9 \pm 0.3$	$68.1 \pm 1.6/0.3 \pm 0.2$	$67.8 \pm 1.2/1.2 \pm 0.7$	$68.7 \pm 1.4/0.9 \pm 0.3$
150	$82.0 \pm 0.3/2.9 \pm 4.0$	$68.2 \pm 0.8/2.3 \pm 0.3$	$68.4 \pm 0.5/0.6 \pm 0.3$	$67.5 \pm 1.0/2.3 \pm 0.5$	$68.9 \pm 0.8/1.3 \pm 1.0$
300	$81.6 \pm 0.5/8.3 \pm 1.6$	$67.1 \pm 1.1/7.6 \pm 1.3$	$\mathbf{68.2 \pm 1.0/1.8 \pm 1.1}$	$66.9 \pm 1.4/6.3 \pm 3.5$	$68.4 \pm 0.6/3.5 \pm 3.0$
500	$81.6 \pm 0.2/21.5 \pm 7.5$	$67.5 \pm 0.8/17.5 \pm 7.7$	$66.6 \pm 2.6/0.8 \pm 0.4$	$67.2 \pm 1.1/20.7 \pm 2.1$	$68.1 \pm 1.3/1.4 \pm 1.0$
1000	$80.6 \pm 0.4/72.2 \pm 14.2$	$65.9 \pm 1.2/44.8 \pm 6.6$	$67.1 \pm 1.4/0.5 \pm 0.5$	$66.6 \pm 0.6/46.1 \pm 11.1$	$67.4 \pm 0.6/1.8 \pm 1.7$
2000	$79.0 \pm 0.6/94.5 \pm 2.6$	$64.6 \pm 0.8/63.9 \pm 3.0$	$66.6 \pm 0.3/0.5 \pm 0.3$	$64.6 \pm 1.0/49.0 \pm 5.4$	$67.8 \pm 1.2/1.8 \pm 0.3$
2500	$78.5 \pm 0.8/97.5 \pm 1.1$	$64.2 \pm 0.9/71.9 \pm 5.1$	$65.9 \pm 2.3/1.2 \pm 0.5$	$64.0 \pm 1.7/62.6 \pm 12.5$	$67.0 \pm 1.2/3.1 \pm 1.1$
5000	$\mathbf{74.3 \pm 0.1/99.5 \pm 0.2}$	$\mathbf{59.3 \pm 2.1/81.4 \pm 5.7}$	$66.1 \pm 1.0/1.0 \pm 0.2$	$\mathbf{60.6 \pm 2.4/66.4 \pm 13.1}$	$\mathbf{66.4 \pm 1.6/7.1 \pm 5.5}$



(a) Centralized learning **(b)** Mean **(c)** Krum

(d) MultiKrum **(e)** TrimmedMean

Figure 7: Effect of aggregation rules on clean-image backdoor performance (SVHN, ResNet-32). CTA-ASR trade-off per aggregation rule.



(a) Centralized learning **(b)** Mean **(c)** Krum

(d) MultiKrum **(e)** TrimmedMean

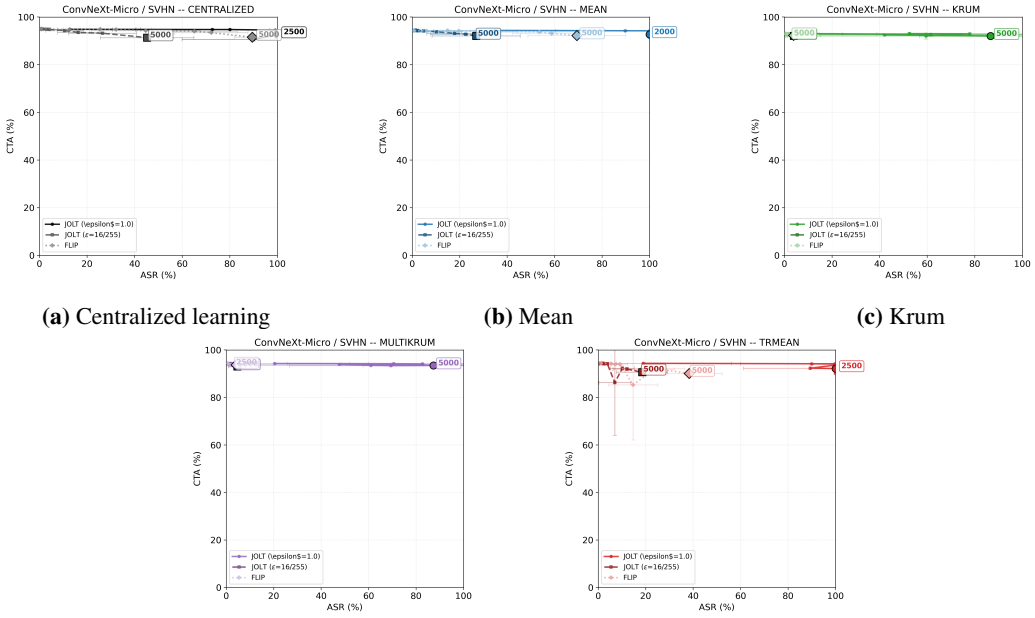
Figure 8: Effect of aggregation rules on clean-image backdoor performance (CIFAR-10, ConvNeXtMicro). CTA-ASR trade-off per aggregation rule.

Table 3: Impact of poisoning budget on ASR and CTA across robust aggregation rules and the centralized (undefended, single-victim) deployment on SVHN with a ResNet-32 model. Results are reported as CTA/ASR (in %) with standard deviation. Bold and underlined values indicate the highest and second-highest ASR.

Budget	CENTRALIZED	MEAN	KRUM	TRMEAN	MULTIKRUM
Optimized Trigger ($\varepsilon = 1.0$) – *JOLT*					
0	95.8±0.2/9.2±23.3	94.7±0.1/1.1±3.0	94.7±0.1/12.9±29.3	94.7±0.1/0.1±0.2	95.1±0.1/0.3±0.8
150	95.7±0.1/100.0±0.0	94.7±0.2/70.4±45.1	94.4±0.4/43.6±46.2	**94.6±0.1/80.1±39.8**	**94.9±0.1/100.0±0.0**
300	95.5±0.1/100.0±0.0	**94.6±0.2/90.0±30.0**	**94.2±0.2/59.6±48.7**	94.5±0.2/70.5±45.0	94.8±0.1/78.9±39.5
500	95.3±0.1/99.8±0.4	94.3±0.3/80.0±40.0	94.1±0.4/49.5±49.5	94.3±0.1/74.5±40.6	94.7±0.1/69.9±45.7
1000	94.6±0.2/90.0±30.0	93.7±0.3/80.0±40.0	93.8±0.5/44.0±46.3	93.8±0.2/70.3±45.4	94.4±0.2/62.2±46.7
2000	**93.2±0.3/100.0±0.0**	92.3±0.4/76.0±38.0	93.9±0.5/42.6±47.4	92.8±0.3/50.0±50.0	94.1±0.2/31.2±45.1
2500	92.7±0.4/100.0±0.0	91.9±0.5/80.0±40.0	93.7±0.5/27.9±42.9	92.5±0.3/40.0±48.9	94.1±0.2/10.7±29.8
5000	89.7±0.6/72.3±42.7	90.5±0.7/85.5±30.1	93.2±0.5/19.9±39.8	91.5±0.4/30.5±45.5	94.0±0.2/10.1±29.9
Optimized Trigger ($\varepsilon = 16/255$) – *JOLT*					
0	95.8±0.1/0.9±0.8	94.7±0.1/0.6±0.4	94.7±0.2/0.5±0.5	94.8±0.1/0.8±0.5	95.1±0.2/0.6±0.5
150	95.7±0.1/3.0±2.1	94.7±0.2/1.1±0.7	94.5±0.2/0.6±0.4	94.7±0.1/1.3±0.8	95.0±0.1/0.6±0.6
300	95.7±0.1/4.8±3.6	94.7±0.1/2.0±1.3	94.1±0.5/0.7±0.3	94.6±0.1/2.1±1.5	94.8±0.1/0.9±0.6
500	95.6±0.1/13.6±12.2	94.6±0.1/3.2±2.3	94.2±0.3/0.8±0.3	94.5±0.1/3.9±3.0	94.6±0.2/0.9±0.6
1000	95.1±0.1/23.7±18.3	94.2±0.1/8.2±5.7	94.1±0.5/1.2±0.7	94.2±0.2/7.0±5.0	94.5±0.1/1.0±0.5
2000	94.1±0.2/43.5±26.5	93.4±0.1/15.4±12.1	93.3±0.5/1.6±1.2	93.4±0.2/14.5±8.1	94.2±0.2/1.7±0.7
2500	93.7±0.1/51.4±26.0	**93.0±0.2/19.6±12.9**	93.6±0.5/2.0±1.6	93.3±0.2/16.1±10.7	94.0±0.2/2.0±1.1
5000	**91.5±0.1/61.2±31.0**	91.9±0.2/19.6±15.3	**93.1±0.7/2.6±1.4**	**92.3±0.2/21.9±14.6**	**94.0±0.2/2.7±1.1**
Sinusoidal Trigger – FLIP					
0	95.8±0.1/1.1±0.6	94.7±0.1/0.9±0.2	94.7±0.1/0.5±0.3	94.7±0.1/1.1±0.3	95.1±0.1/0.7±0.3
150	95.8±0.1/37.4±15.9	94.7±0.2/7.7±3.5	94.1±0.5/1.3±0.5	94.6±0.1/6.3±2.2	94.8±0.2/2.0±1.1
300	95.7±0.1/88.5±9.2	94.6±0.1/35.7±11.2	93.9±0.5/1.5±0.4	94.4±0.2/27.4±8.9	94.8±0.1/1.8±0.4
500	95.6±0.1/99.3±0.8	94.5±0.1/70.7±9.4	93.7±0.6/1.2±0.7	94.3±0.1/50.4±11.9	94.6±0.1/1.7±0.7
1000	**95.3±0.1/99.9±0.1**	**94.4±0.2/90.1±4.7**	93.8±0.6/1.9±1.0	94.1±0.2/77.5±5.6	94.5±0.1/1.7±0.9
2000	94.6±0.2/99.9±0.2	94.0±0.2/87.9±8.7	93.5±0.3/1.9±1.0	**93.7±0.1/96.1±2.3**	94.4±0.2/3.1±1.4
2500	94.2±0.2/98.5±4.2	93.7±0.3/78.9±16.3	93.6±0.5/2.3±0.8	93.4±0.3/95.8±3.1	94.1±0.2/2.1±0.5
5000	92.3±0.3/80.2±37.1	92.2±0.4/68.8±30.2	**93.4±0.4/4.7±1.7**	92.5±0.5/91.4±9.0	**94.1±0.2/5.0±1.6**
GCB					
0	$95.5 \pm 0.1/2.7 \pm 0.4$	$91.8 \pm 0.2/2.4 \pm 0.3$	$93.5 \pm 0.1/2.7 \pm 1.1$	$91.6 \pm 0.3/2.4 \pm 0.6$	$92.4 \pm 0.2/2.5 \pm 0.8$
150	$95.3 \pm 0.2/32.8 \pm 2.9$	$91.5 \pm 0.1/26.9 \pm 5.8$	$92.6 \pm 0.5/3.7 \pm 0.7$	$91.6 \pm 0.3/27.3 \pm 3.6$	$92.3 \pm 0.4/8.7 \pm 0.5$
300	$95.0 \pm 0.1/86.1 \pm 1.0$	$91.2 \pm 0.2/74.1 \pm 10.2$	$92.4 \pm 0.4/9.2 \pm 5.4$	$91.3 \pm 0.2/64.4 \pm 10.2$	$92.0 \pm 0.2/17.4 \pm 7.6$
500	$94.7 \pm 0.0/99.6 \pm 0.1$	$90.9 \pm 0.2/93.7 \pm 3.6$	$\mathbf{92.4 \pm 0.6/10.3 \pm 6.1}$	$90.7 \pm 0.5/87.8 \pm 3.8$	$91.9 \pm 0.3/16.7 \pm 7.1$
1000	$\mathbf{93.7 \pm 0.1/100.0 \pm 0.0}$	$89.7 \pm 0.4/99.9 \pm 0.1$	$92.5 \pm 0.3/4.2 \pm 1.1$	$90.0 \pm 0.3/99.6 \pm 0.2$	$\mathbf{91.7 \pm 0.2/18.9 \pm 3.1}$
2000	$\mathbf{92.5 \pm 0.1/100.0 \pm 0.0}$	$\mathbf{88.6 \pm 0.2/100.0 \pm 0.0}$	$92.4 \pm 0.7/2.5 \pm 0.5$	$\mathbf{89.0 \pm 0.2/99.8 \pm 0.4}$	$91.8 \pm 0.2/4.1 \pm 1.2$
2500	$\mathbf{91.6 \pm 0.0/100.0 \pm 0.0}$	$\mathbf{87.0 \pm 0.9/100.0 \pm 0.0}$	$92.0 \pm 0.5/2.2 \pm 0.5$	$86.9 \pm 1.9/95.6 \pm 7.6$	$91.8 \pm 0.2/4.1 \pm 1.4$
5000	$\mathbf{88.1 \pm 0.2/100.0 \pm 0.0}$	$\mathbf{83.2 \pm 0.6/100.0 \pm 0.0}$	$92.8 \pm 0.3/4.3 \pm 0.5$	$84.2 \pm 1.1/99.0 \pm 1.7$	$91.8 \pm 0.2/5.2 \pm 0.5$

Table 4: Impact of poisoning budget on ASR and CTA across robust aggregation rules and the centralized (undefended, single-victim) deployment on CIFAR-10 with a ConvNeXt-Micro model. Results are reported as CTA/ASR (in %) with standard deviation. Bold and underlined values indicate the highest and second-highest ASR.

Budget	CENTRALIZED	MEAN	KRUM	TRMEAN	MULTIKRUM
			Optimized Trigger ($\varepsilon = 1.0$) – *JOLT*		
0	90.1±0.2/8.1±16.4	87.6±0.3/0.7±1.2	80.6±1.1/25.4±34.0	87.3±0.3/27.0±36.0	86.3±0.4/10.2±19.3
150	89.9±0.2/21.9±30.4	87.3±0.2/10.9±21.2	80.7±0.9/43.2±40.6	87.1±0.3/14.0±28.1	85.8±0.4/34.8±38.0
300	89.6±0.3/34.4±34.6	86.9±0.4/17.4±24.9	80.4±1.0/31.1±21.6	86.9±0.4/39.8±38.4	85.5±0.3/21.1±32.3
500	89.4±0.2/37.5±42.3	86.7±0.3/40.3±38.8	80.2±0.5/21.8±23.5	86.7±0.3/45.8±35.5	85.6±0.2/18.3±29.3
1000	88.5±0.4/74.6±34.1	86.0±0.3/57.6±33.7	79.7±1.0/42.0±40.9	86.2±0.3/71.7±35.7	84.9±0.6/22.8±34.7
2000	86.6±0.2/98.7±2.3	**84.7±0.3/81.9±26.6**	79.4±1.0/31.5±33.0	84.7±0.5/77.6±37.9	84.9±0.3/20.9±25.4
2500	**85.8±0.4/99.4±0.8**	84.3±0.3/79.8±38.0	79.9±1.0/43.3±41.1	**84.3±0.4/89.8±16.6**	84.8±0.4/10.1±14.1
5000	80.8±0.4/94.6±14.0	81.5±0.4/70.1±39.4	**79.6±0.8/69.0±34.5**	81.0±1.0/69.6±36.2	**84.5±0.4/36.8±34.6**
			Optimized Trigger ($\varepsilon = 16/255$) – *JOLT*		
0	90.2±0.2/0.1±0.1	87.5±0.2/0.1±0.1	80.6±0.6/1.1±1.3	87.5±0.3/0.2±0.2	86.3±0.3/0.3±0.3
150	89.9±0.3/1.4±1.6	87.2±0.3/1.5±1.8	80.3±0.9/2.4±4.0	87.2±0.2/1.6±2.0	**85.8±0.3/3.0±4.4**
300	89.7±0.3/5.2±9.3	87.1±0.3/10.2±16.4	79.9±1.2/2.4±3.5	87.2±0.2/2.8±3.3	85.9±0.4/0.4±0.3
500	89.4±0.3/7.7±10.5	87.0±0.2/6.0±6.6	80.0±1.3/2.4±2.8	86.8±0.4/4.5±4.3	85.6±0.3/0.8±1.3
1000	88.6±0.2/19.3±16.5	86.2±0.3/19.5±16.2	79.7±0.8/1.6±1.6	86.2±0.2/6.5±4.8	85.3±0.2/0.7±0.5
2000	86.6±0.2/39.0±27.2	84.7±0.5/36.0±25.6	79.4±1.0/3.3±4.4	**84.8±1.1/16.4±11.9**	84.9±0.2/0.9±1.1
2500	85.5±0.5/44.0±31.2	**83.9±0.4/39.8±23.1**	**80.2±1.0/3.6±4.8**	84.4±0.5/11.0±5.6	84.8±0.4/0.7±0.7
5000	**80.6±0.3/61.3±29.8**	81.7±0.5/36.4±26.7	79.7±1.0/3.1±4.1	80.3±1.0/15.7±8.7	84.7±0.4/0.8±0.5
			Sinusoidal Trigger – FLIP		
0	90.2±0.3/0.1±0.1	87.3±0.3/0.4±0.2	80.9±0.9/0.6±0.2	87.2±0.2/0.3±0.1	86.3±0.4/0.2±0.2
150	90.0±0.2/3.7±0.8	87.2±0.1/8.3±2.2	80.3±1.0/1.7±0.6	87.2±0.2/5.3±2.8	**85.9±0.4/1.1±0.8**
300	89.5±0.3/13.1±4.4	87.1±0.3/21.4±4.8	80.7±0.7/1.8±0.5	86.7±0.4/8.0±2.4	85.7±0.5/1.0±0.4
500	89.2±0.2/28.7±9.8	86.3±0.7/37.4±15.2	**80.4±1.1/2.2±1.0**	86.8±0.1/8.6±3.5	85.6±0.2/1.0±0.3
1000	88.3±0.4/51.0±13.8	86.2±0.2/62.4±10.8	79.6±1.1/1.5±0.6	85.7±0.3/12.0±5.4	85.0±0.5/0.8±0.2
2000	86.4±0.3/71.3±12.3	84.9±0.6/71.9±13.3	79.8±1.1/1.5±0.8	83.1±1.3/20.6±3.8	85.3±0.4/1.0±0.3
2500	85.3±0.1/76.9±11.9	84.4±0.7/70.3±15.8	79.2±1.3/1.6±0.7	82.3±1.2/25.5±5.7	84.9±0.6/1.0±0.6
5000	**80.6±0.4/82.8±10.0**	**81.5±0.8/73.7±19.1**	80.2±0.9/1.8±0.7	**77.7±1.4/29.3±7.9**	84.7±0.3/1.1±0.4
			GCB		
0	$78.1 \pm 0.8/0.3 \pm 0.1$	$71.4 \pm 0.5/0.4 \pm 0.3$	$67.6 \pm 3.4/0.5 \pm 0.4$	$71.4 \pm 0.6/0.2 \pm 0.0$	$70.2 \pm 0.4/0.5 \pm 0.3$
150	$77.3 \pm 0.9/4.0 \pm 3.4$	$71.1 \pm 0.9/4.4 \pm 1.0$	$65.7 \pm 0.9/1.0 \pm 0.7$	$71.2 \pm 1.0/5.0 \pm 0.6$	$70.2 \pm 1.0/1.5 \pm 0.9$
300	$77.2 \pm 0.9/26.9 \pm 23.4$	$70.8 \pm 0.5/14.6 \pm 10.2$	$67.0 \pm 1.9/0.9 \pm 0.4$	$70.8 \pm 0.6/18.9 \pm 11.9$	$70.0 \pm 0.9/4.3 \pm 2.5$
500	$77.3 \pm 1.0/57.0 \pm 20.1$	$70.1 \pm 0.8/61.6 \pm 9.0$	$65.6 \pm 1.7/2.7 \pm 2.4$	$70.8 \pm 1.0/60.2 \pm 12.0$	$\mathbf{69.8 \pm 0.2/6.1 \pm 9.6}$
1000	$76.5 \pm 0.6/79.2 \pm 6.0$	$69.5 \pm 0.8/87.8 \pm 12.8$	$64.9 \pm 1.9/2.5 \pm 2.3$	$69.5 \pm 0.4/85.4 \pm 7.4$	$69.2 \pm 1.1/1.7 \pm 1.2$
2000	$74.6 \pm 1.3/98.2 \pm 2.6$	$67.7 \pm 0.4/99.5 \pm 0.3$	$\mathbf{64.7 \pm 1.3/3.2 \pm 1.3}$	$\mathbf{71.7 \pm 0.5/90.8 \pm 4.2}$	$69.0 \pm 0.6/3.1 \pm 2.9$
2500	$73.7 \pm 0.9/99.9 \pm 0.2$	$67.1 \pm 0.6/99.6 \pm 0.5$	$63.6 \pm 1.4/1.0 \pm 0.8$	$72.8 \pm 1.0/84.6 \pm 19.2$	$68.5 \pm 0.8/1.5 \pm 1.3$
5000	$\mathbf{69.3 \pm 1.0/100.0 \pm 0.0}$	$\mathbf{63.2 \pm 1.3/100.0 \pm 0.0}$	$64.6 \pm 1.1/0.6 \pm 0.2$	$69.6 \pm 1.6/56.1 \pm 20.7$	$68.0 \pm 0.7/1.7 \pm 1.6$



(a) Centralized learning **(b)** Mean **(c)** Krum

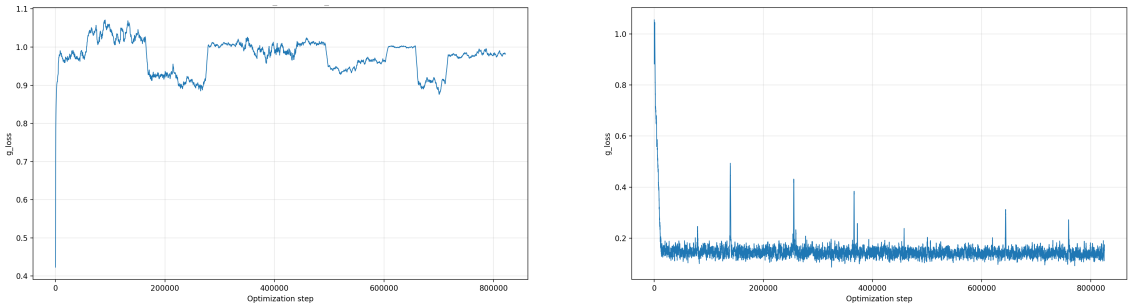
(d) MultiKrum **(e)** TrimmedMean

Figure 9: Effect of aggregation rules on clean-image backdoor performance (SVHN, ConvNeXtMicro). CTA-ASR trade-off per aggregation rule.

Table 5: Impact of poisoning budget on ASR and CTA across robust aggregation rules and the centralized (undefended, single-victim) deployment on SVHN with a ConvNeXt-Micro model. Results are reported as CTA/ASR (in %) with standard deviation. Bold and underlined values indicate the highest and second-highest ASR.

Budget	CENTRALIZED	MEAN	KRUM	TRMEAN	MULTIKRUM
			Optimized Trigger ($\varepsilon = 1.0$) – *JOLT*		
0	94.8±0.3/12.8±29.7	94.4±0.2/34.8±40.9	93.0±0.5/8.3±16.1	94.4±0.2/19.0±37.2	94.3±0.3/20.4±39.5
150	94.7±0.3/72.7±41.9	94.2±0.3/89.8±29.4	92.7±0.4/61.5±46.9	94.2±0.2/100.0±0.1	94.2±0.3/70.6±44.9
300	94.7±0.2/80.1±39.6	**94.2±0.1/100.0±0.1**	93.0±0.6/52.5±44.9	94.1±0.3/90.0±30.0	94.1±0.3/82.7±30.7
500	94.5±0.3/99.1±2.3	94.0±0.4/99.9±0.2	92.9±0.5/77.8±39.0	94.1±0.4/99.8±0.4	93.9±0.2/82.8±34.1
1000	94.2±0.4/99.6±0.8	93.6±0.2/100.0±0.0	92.0±1.3/59.5±48.0	93.6±0.2/99.9±0.2	94.0±0.2/47.6±46.0
2000	**93.2±0.4/100.0±0.0**	92.7±0.3/100.0±0.0	92.5±0.4/61.4±46.3	92.3±0.6/89.3±28.0	93.5±0.2/60.9±47.8
2500	92.6±0.5/100.0±0.0	92.5±0.4/100.0±0.0	92.5±0.4/42.1±47.4	**92.1±0.4/100.0±0.0**	93.4±0.3/69.4±42.6
5000	90.6±0.6/90.0±30.0	91.5±0.6/100.0±0.0	**92.0±0.5/86.6±29.9**	90.2±0.6/100.0±0.1	**93.4±0.3/87.3±29.6**
			Optimized Trigger ($\varepsilon = 16/255$) – *JOLT*		
0	95.0±0.3/0.8±0.2	94.5±0.2/1.3±1.2	92.8±0.3/1.0±0.7	94.2±0.2/1.1±0.8	94.2±0.3/1.3±1.2
150	95.0±0.3/1.3±0.5	94.3±0.3/2.1±1.4	92.8±0.4/1.4±1.0	94.4±0.3/1.7±1.0	94.3±0.3/1.3±0.9
300	94.9±0.2/2.7±1.7	94.2±0.2/2.5±1.7	92.8±0.6/1.6±1.4	94.2±0.2/2.5±1.9	94.1±0.3/2.2±1.4
500	94.8±0.2/4.1±3.4	94.1±0.2/4.9±4.2	92.6±0.7/2.0±1.6	94.2±0.2/3.9±3.8	94.1±0.4/1.9±1.3
1000	94.3±0.3/10.6±7.9	93.8±0.2/10.2±10.7	92.8±0.4/2.4±1.4	86.2±22.2/7.0±6.7	93.8±0.3/2.9±2.9
2000	93.5±0.4/16.4±8.7	93.1±0.2/17.8±11.6	92.7±0.4/2.7±1.5	92.2±1.0/10.2±5.0	93.5±0.3/2.7±1.7
2500	93.2±0.3/26.5±14.3	92.8±0.1/22.6±14.8	92.1±0.5/2.9±2.2	91.9±0.7/12.1±5.8	93.5±0.3/3.4±1.8
5000	**91.3±0.4/45.3±19.7**	**92.1±0.5/27.0±18.6**	**92.1±0.4/4.0±2.9**	**90.6±0.7/18.5±10.7**	**93.1±0.3/4.7±3.6**
			Sinusoidal Trigger – FLIP		
0	95.0±0.2/0.8±0.4	94.3±0.2/0.8±0.6	93.0±0.5/0.5±0.3	94.4±0.2/1.2±0.5	94.3±0.2/0.8±0.4
150	94.7±0.2/12.6±8.5	94.1±0.3/6.2±4.2	92.8±0.6/1.1±0.5	94.2±0.3/5.5±2.3	94.1±0.2/3.2±1.3
300	94.9±0.2/25.6±15.4	94.4±0.2/14.9±9.5	92.7±0.3/2.2±0.8	94.0±0.2/7.2±2.6	94.1±0.2/3.4±1.3
500	94.8±0.2/32.2±14.9	94.2±0.3/19.6±7.8	92.7±0.5/2.0±0.8	94.0±0.2/9.1±3.3	94.0±0.2/3.4±1.5
1000	94.3±0.3/44.7±22.9	94.0±0.2/32.5±11.7	92.6±0.7/1.9±0.6	85.3±23.2/14.7±10.4	93.8±0.3/3.5±2.8
2000	94.0±0.2/65.1±18.0	93.7±0.2/53.6±16.2	92.2±0.4/2.5±1.1	92.2±0.5/24.2±12.2	93.6±0.3/2.0±0.7
2500	93.4±0.4/72.2±15.4	93.1±0.5/58.8±16.5	92.7±0.5/2.2±1.2	91.9±0.5/25.6±6.5	**93.7±0.2/4.1±1.1**
5000	**91.5±0.8/89.5±7.6**	**92.2±0.6/69.4±20.6**	**92.3±0.8/3.8±1.6**	**90.1±0.5/38.4±13.9**	93.3±0.3/4.1±1.4
			GCB		
0	$93.2 \pm 0.2/1.8 \pm 0.2$	$90.2 \pm 0.1/1.6 \pm 0.6$	$90.2 \pm 0.3/1.6 \pm 0.5$	$90.2 \pm 0.2/1.8 \pm 0.4$	$90.3 \pm 0.2/1.6 \pm 0.4$
150	$93.0 \pm 0.1/35.4 \pm 5.6$	$89.8 \pm 0.2/33.3 \pm 7.2$	$89.2 \pm 0.5/2.9 \pm 1.2$	$90.0 \pm 0.2/31.5 \pm 5.1$	$90.0 \pm 0.2/12.4 \pm 12.7$
300	$92.7 \pm 0.3/78.2 \pm 8.7$	$89.6 \pm 0.5/70.7 \pm 2.3$	$89.7 \pm 0.7/3.8 \pm 1.6$	$89.6 \pm 0.1/64.9 \pm 3.5$	$\mathbf{90.1 \pm 0.6/53.2 \pm 40.5}$
500	$92.4 \pm 0.1/98.4 \pm 0.8$	$89.3 \pm 0.2/93.2 \pm 2.2$	$\mathbf{89.4 \pm 0.5/5.1 \pm 3.8}$	$89.5 \pm 0.2/88.0 \pm 6.3$	$90.0 \pm 0.1/11.5 \pm 6.4$
1000	$\mathbf{91.4 \pm 0.1/100.0 \pm 0.0}$	$\mathbf{88.2 \pm 0.3/100.0 \pm 0.0}$	$88.7 \pm 0.7/3.0 \pm 0.7$	$88.6 \pm 0.1/99.5 \pm 0.6$	$89.6 \pm 0.1/7.1 \pm 2.6$
2000	$89.8 \pm 0.1/100.0 \pm 0.0$	$87.1 \pm 0.6/100.0 \pm 0.0$	$89.2 \pm 0.2/3.0 \pm 1.4$	$\mathbf{87.9 \pm 0.7/99.7 \pm 0.2}$	$89.7 \pm 0.1/3.6 \pm 0.4$
2500	$89.1 \pm 0.2/100.0 \pm 0.0$	$86.3 \pm 0.4/100.0 \pm 0.0$	$89.0 \pm 0.5/3.3 \pm 1.3$	$88.6 \pm 0.5/98.8 \pm 1.8$	$89.6 \pm 0.1/3.7 \pm 1.0$
5000	$85.3 \pm 0.4/100.0 \pm 0.0$	$82.7 \pm 0.4/100.0 \pm 0.0$	$88.9 \pm 0.1/3.6 \pm 1.9$	$\mathbf{87.5 \pm 0.9/99.7 \pm 0.4}$	$89.8 \pm 0.2/4.6 \pm 0.9$

F STABILIZING TRAJECTORY MATCHING



(a) One expert checkpoint per epoch.

(b) One random expert checkpoint per batch.

Figure 10: Comparison of FLIP's trajectory-matching optimization, which loads one expert checkpoint per epoch, with our approach, which loads a random expert batch at each iteration.