WHEN AND HOW OFTEN TO DEFEND?
ON THE OVERLOOKED ROLE OF LEARNING RATE SCHEDULES IN POISONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Data-poisoning defenses are typically applied throughout training, even though they can substantially reduce model accuracy. In this work, we ask: *when and how frequently should a defense be applied?* We highlight the overlooked role of the learning-rate schedule and prove that the effect of a defense at a given step scales with the squared learning rate η^2. Defending therefore matters most within the window containing the largest cumulative η^2 mass. We further relate defense decisions to the current gradient-state information and classify training-time defenses into three regimes: *state independent, state dependent, and mediator*. We evaluate these regimes against the seminal Witches' Brew attack on CIFAR-10, CIFAR-100, and Tiny ImageNet, using a DP-SGD-style defense applied to the aggregated gradient. Our experiments show that attack success rises primarily around the first learning-rate drop despite a stable poisoning frequency. When the defense is restricted to the window carrying most of the η^2 mass, gradient-state information becomes valuable: a state-dependent policy matches a random policy at a lower activation rate. Most importantly, activating the defense at only a small fraction of training steps, well distributed across the schedule, fully prevents the attack while only slightly reducing clean accuracy, yielding a better security/accuracy trade-off. Effective defense therefore requires far fewer interventions than commonly assumed, and where these interventions fall on the learning-rate schedule matters as much as their frequency.

1 INTRODUCTION

Data poisoning attacks modify the training set to change the behaviour of the trained model. Two goals are commonly distinguished. Availability attacks aim to degrade overall predictive accuracy, whereas integrity attacks aim to alter predictions on a small set of target inputs while preserving accuracy elsewhere. In this work, we focus on integrity attacks in the *clean-label* setting, which preserve the original labels. We consider an attacker restricted to applying small, bounded perturbations to training examples to induce the victim model to misclassify a target sample. Specifically, we focus on the seminal Witches' Brew attack (Geiping et al., 2021), which has informed subsequent work on gradient-matching-based data poisoning (Bouaziz et al., 2025; Souri et al., 2022; Bouaziz et al., 2026; Schwarzschild et al., 2021; Liu et al., 2022a).

Several defenses against this type of attack have been proposed. The defender may, for example, transform the update at each step through clipping Hong et al. (2020), noise addition Abadi et al. (2016), or robust aggregation Blanchard et al. (2017); Vlaski et al. (2022); Bareilles et al. (2026). However, such interventions come at a cost: the same transformation perturbs the clean gradient signal and, when applied too aggressively, degrades accuracy.

The standard evaluation of such a defense is a two-run comparison: one model is trained with the defense active at every step and another without it, after which their final metrics are compared. This answers only a coarse question: *Does the defense work?* and can be misleading in two ways. First, a defense that neutralizes the attack while destroying accuracy cannot be considered successful, and a single pair of endpoint measurements provides no indication of when the cost was incurred. Second, applying the defense may be essential at some steps but purely costly at others.

We therefore ask a more granular question: *When does defending actually matter?* To address this question, we make the following contributions:

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.

- We identify the learning-rate (LR) schedule as a previously overlooked driver of poisoning and defense dynamics, and prove that the effect of a defense is governed by its cumulative squared LR mass, $\sum_k \eta_k^2$. This motivates the *squared-mass learning-rate (SMLR) window*, which contains the steps where this mass concentrates and the defense is therefore most effective. (Section 3.1)
- We introduce a novel taxonomy of training-time defenses that stems from the current gradient state information. (Section 3.2)
- Our empirical findings answer three questions (Section 4).
  - **When to defend?** The attack succeeds primarily around the first learning-rate drop despite poisoning frequency. Defending within the SMLR window prevents the attack across different LR schedules, whereas interventions at low learning rate levels fail or provide little additional benefit. (Subsection 4.2)
  - **How frequently to defend?** Sparse random interventions can be sufficient: activating the defense at only 10% of training steps prevents the attack at some configurations, while preserving more clean accuracy than higher activation rates. An oracle that knows which mini-batches contain poisoned samples provides only a modest benefit from the gradient state information. (Subsection 4.3)
  - **How to defend effectively?** Combining the SMLR window with sparse interventions provides the most reliable security/accuracy trade-off across learning-rate schedules. Within this window, random activation are as efficient as the always-on defense with fewer interventions, while the poison-aware oracle provides no clear advantage at a comparable rate. (Subsection 4.4)

2 Related work

Poisoning by gradient matching. Clean-label poisoning can be formulated as a bilevel optimisation problem: the outer problem selects the perturbation, while the inner problem corresponds to training the victim model. Solving this problem directly is computationally heavy for deep networks. Geiping et al. (2021) circumvent this problem using *gradient matching*: crafting perturbations so that the gradient of the poisoned training loss points in the same direction as the gradient of the adversarial loss on the target. This approach has inspired several extensions. Souri et al. (2022) use gradient matching to inject hidden trigger backdoors during training. Bouaziz et al. (2025) construct strong poisons by inverting gradient attacks, while Bouaziz et al. (2026) extend this approach to indirect data poisoning during language model pre-training. In the clean-label setting considered here, poisoned examples remain visually close to their clean counterparts. These constraints motivate studying defenses that act both on the features and on gradients to stop the attack's influence.

Defenses against data poisoning. Existing defenses can be grouped according to where they intervene. A first group acts *before training* by filtering the data, and removing outliers in feature space Paudice et al. (2018); Steinhardt et al. (2017). A second group acts on the training data using strong data augmentation Borgnia et al. (2021), or by injecting benign noise to counteract the effects of poisoning Liu et al. (2022b). A third group acts *on the gradient space*. Hong et al. (2020) study a variant of the clipping-and-noising mechanism used in DP-SGD Abadi et al. (2016), applied to the aggregated gradient to limit the influence of poisoned examples on training. Yuzhe et al. (2019) study poisoning attacks against differentially private learners. Gradient-based defenses are particularly relevant to gradient matching attacks because they act directly on the gradients that the attacker seeks to manipulate. However, standard approaches apply them at every training step, raising a natural question: is continuous defense necessary throughout training?

Poisoning and training dynamics. The timing of defending may matter because training is not a stationary process. Prior work on online poisoning already treats the attack as a sequential problem. Wang & Chaudhuri (2018) study attacks on online learners and show that the effect of a poisoned example depends on when it is presented. Zhang et al. (2020) formulate online poisoning as a RL problem, in which the attacker acts on a training trajectory rather than on a fixed dataset. These studies motivate a complementary question on the defense side: Can we consider training dynamics when deciding when to defend? Hence, if vulnerability is concentrated in particular phases of training, applying the defense at every step may impose an unnecessary accuracy cost.

3 METHOD

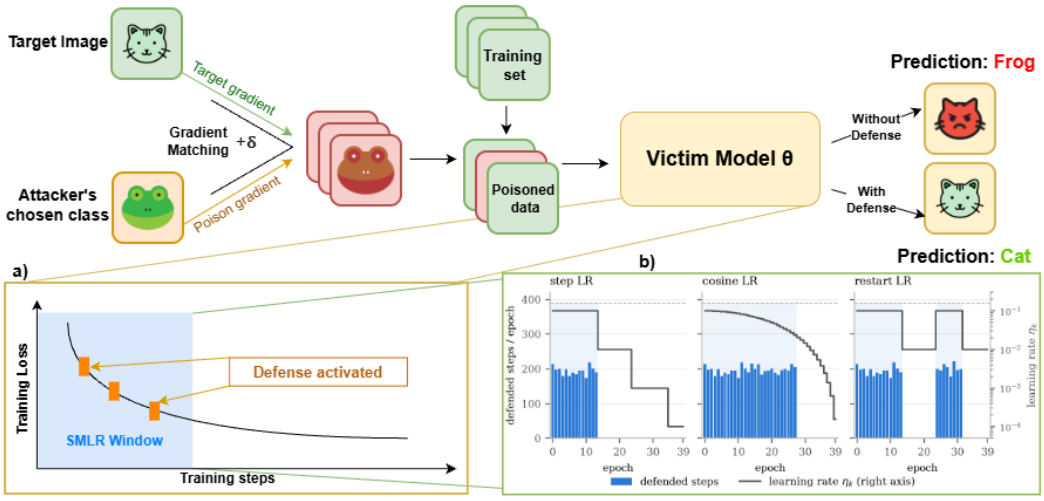


Figure 1: **Witches' Brew attack and our proposed defense strategy. Top:** Witches' Brew gradient-matching based attack (Geiping et al., 2021) perturbs a small set of images from the adversarial class (*frog*) so that their training gradient aligns with the adversarial gradient of a target image (*cat*). Without defense, the trained model predicts the target as *frog*; a successful defense preserves the correct prediction *cat*. **Bottom (our method):** (a) We treat defense activation as a per-step decision rather than applying it throughout training. (b) We use the learning-rate schedule to identify the *SMLR* window and place sparse defended steps within it.

We study one training run of length K, with steps indexed by $k \in \{1, \dots, K\}$. Let $\theta_k \in \mathbb{R}^d$ denote the model parameters at step k. The attacker fixes a static poison Δ before training begins, which remains unobserved by the defender throughout training. The poison affects the training process only through the gradients it induces.

At step k, let G_k denote the *gradient state*, namely the gradient information available to the defense policy, and let $g_k \in \mathbb{R}^d$ denote the aggregated mini-batch gradient. The *defense* D_k is the action selected at that step. Depending on the regime, this decision may depend on G_k; once selected, it transforms g_k into the vector applied by the optimiser. The taxonomy in Section 3.2 is a statement about how D_k may depend only on G_k.

Throughout this paper we consider the clipping-and-noising defense, introduced by Hong et al. (2020) inspired from DP-SGD Abadi et al. (2016), applied to the aggregated mini-batch gradient rather than to per-example gradients. We instantiate it as a gate,

$$D_k \in \{\,\mathrm{id},\ \mathcal{M}\,\}, \qquad \mathcal{M}(g) \;=\; g \cdot \min\!\Big(1,\ \tfrac{C}{\|g\|}\Big) \;+\; C\sigma\,\xi, \qquad \xi \sim \mathcal{N}(0, I_d), \tag{1}$$

Where $g \in \mathbb{R}^d$ is the aggregated mini-batch gradient, $C > 0$ is the clipping norm, $\sigma \geq 0$ is the noise multiplier, and $\xi \in \mathbb{R}^d$ is standard Gaussian noise drawn independently at each step. The identity operator id leaves g unchanged, whereas $\mathcal{M}$ clips its norm to at most C and adds Gaussian noise. We call the fraction of steps for which $D_k = \mathcal{M}$ the *activation rate*.

3.1 THEORETICAL ANALYSIS

We now ask what a defended step does to the attack. The answer turns out to depend on the learning rate, and therefore on the schedule.

Notation. We use the notation of Geiping et al. (2021). Let $L_{\mathrm{adv}}(\theta) := L\big(F(x_t, \theta),\, y_{\mathrm{adv}}\big)$ be the adversarial loss on the single target, and $L(\theta) := \frac{1}{N}\sum_{i=1}^{N} L(x_i, y_i; \theta)$ the ordinary training loss.

We evaluate their cosine similarity based objective $B(\Delta, \theta)$ at the current parameters θ_k:

$$1 - B(\Delta, \theta_k) \;=\; \frac{\big\langle \nabla_\theta L_{\mathrm{adv}}(\theta_k),\, \nabla_\theta L(\theta_k) \big\rangle}{\|\nabla_\theta L_{\mathrm{adv}}(\theta_k)\| \cdot \|\nabla_\theta L(\theta_k)\|} \;\in\; [-1, 1]. \tag{2}$$

We write $c_k := 1 - B(\Delta, \theta_k)$ for brevity. The attack works by making c_k large.

Following the Geiping et al. (2021) framework, the victim's own training drives the adversarial loss down, step after step, while the victim train normally and does nothing wrong. Theoretically, two assumptions carry the result:

(A1) ∇L_{adv} is L-Lipschitz (global smoothness).

(A2) The victim update is exactly $\theta_{k+1} = \theta_k - \eta_k \nabla L(\theta_k)$.

The defended update. On a defended step, $D_k = \mathcal{M}$ transforms the aggregated gradient g_k into the defended aggregated gradient $\hat{g}_k$. The update in *(A2)* becomes

$$\theta_{k+1} \;=\; \theta_k \;-\; \eta_k\, \hat{g}_k, \qquad \hat{g}_k \;:=\; D_k(g_k), \tag{3}$$

where η_k is the learning rate at step k, set by the schedule.

Proposition 1 (Defended adversarial-loss bound) *Assume* (A1)*, let step k be defended ($D_k = \mathcal{M}$) with ξ_k drawn independently of θ_k and satisfying $\mathbb{E}|\xi_k|^2 = d$. Then*

$$\mathbb{E}\big[L_{\mathrm{adv}}(\theta_{k+1}) - L_{\mathrm{adv}}(\theta_k)\big] \;\le\; -\eta_k\, c_k \,\big\|\nabla L_{\mathrm{adv}}(\theta_k)\big\|\, C \;+\; L\eta_k^2 C^2 \;+\; \underbrace{L\eta_k^2 C^2 \sigma^2 d}_{\textit{penalty added by the defense}} \tag{4}$$

The proof is given in Appendix A.

The three terms separate cleanly. The first is the attack: it is negative whenever the gradients are aligned ($c_k > 0$), and it is what drives the adversarial loss down. The second is the ordinary cost of a finite step. The third exists only because the defense is active. It is positive, so it works *against* the descent of the adversarial loss. Focusing on this term, a defended step is therefore not worth the same at every moment of training. **Its value is set by the learning rate η_k, and the learning rate is set by the schedule.**

Under a defense policy. Summing the defense's contribution over the run gives

$$\sum_{k=1}^{K} \mathbf{1}\{D_k = \mathcal{M}\}\, L\eta_k^2 C^2 \sigma^2 d \;=\; L\, C^2 \sigma^2 d \sum_{k=1}^{K} \mathbf{1}\{D_k = \mathcal{M}\}\, \eta_k^2. \tag{5}$$

Consequently, two policies that defend the same number of steps but at different timings do not have the same effect: what matters is the cumulative squared learning-rate mass $\sum_k \eta_k^2$ covered by those steps.

Definition 3.1 (The Squared Mass Learning Rate (SMLR) Window) *Given the learning rates $\eta_1, \ldots, \eta_K$ defined at each step of the training, and a threshold $\tau \in (0, 1]$, the SMLR window is the smallest set of steps that holds a fraction τ of the cumulative squared η; $\sum_k \eta_k^2$:*

$$\mathcal{W}_\tau \;=\; \operatorname*{arg\,min}_{\mathcal{W} \subseteq \{1, \ldots, K\}} \Big\{ |\mathcal{W}| \;:\; \sum_{k \in \mathcal{W}} \eta_k^2 \;\ge\; \tau \sum_{k=1}^{K} \eta_k^2 \Big\}. \tag{6}$$

Thus, we define the **squared-mass learning-rate (SMLR) window** as the smallest set of steps including the largest mass (i.e. 99%) of $\sum_k \eta_k^2$. Because this mass is concentrated in the high-learning-rate phases, interventions within the SMLR window have greater leverage than equally many interventions outside it. Table 1 illustrates this concentration for the schedule used by Geiping et al. (2021) with ResNet-18 on CIFAR-10: $\eta_0 = 0.1$, divided by 10 at epochs 14, 24, and 35.

Table 1: The SMLR window, Where the squared mass learning rate concentrates.

Window	share of $\sum_k \eta_k$	share of $\sum_k \eta_k^2$
Epochs 0-13 (before the first drop)	92.6%	**99.28%**
Epochs 14-39 (after the first drop)	7.4%	0.72%

3.2 TAXONOMY

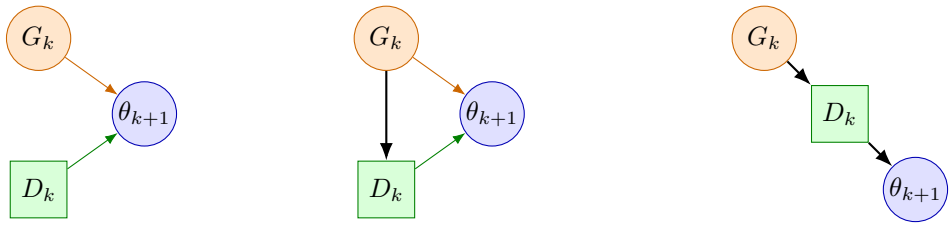
A defense that neutralizes the attack at the cost of model's performance is not optimal. Such degradation may result from applying the defense too aggressively or at steps where intervention is unnecessary. This trade-off motivates examining the gradient state G_k before deciding whether and how frequently to defend. We therefore distinguish training-time defenses according to how the action D_k depends on G_k, yielding the three regimes illustrated in Figure 2.

Regime A: State independent. Here, the defense is defined and fixed before training starts. At each step it is applied by a random rule independently of G_k: a coin flip with a fixed probability. Thus, there is no edge from G_k into D_k.

Regime B: State dependent. Here, the defense reads the gradient state, computes an anomaly score for it, and then decides whether to intervene or not. When the gradient seems anomalous the defense is activated ($D_k = \mathcal{M}$); otherwise training proceeds normally ($D_k = \mathrm{id}$). *Example:* this design can accommodate any defense that operates on per-step gradients, such as differential privacy (Abadi et al., 2016) or robust aggregation approaches from distributed learning (Blanchard et al., 2017; Vlaski et al., 2022).

The best policy in this regime is the one able to distinguish poisoned gradients from clean ones. To mimic this best possible defense, we construct an *oracle* defense that is told which mini-batches contain poisoned samples and reacts only to those. The oracle is **not deployable.** It is a measuring instrument: it bounds what any possible state-dependent (Regime B) policy could achieve if its anomaly score were perfect.

Regime C: Mediator. This is the standard protocol of defenses. The per-step defenses are already assigned, the same transformation is applied at every step, so every gradient is transformed in the same way before the update. The defense decision, $D_k \in \{\,\mathrm{id},\ \mathcal{M}\,\}$, is taken once, at initialisation, and never revisited. Almost every published evaluation of a training-time defense is a comparison of this regime against no defense at all, which is the two-run comparison we argue is too coarse.



A: State independent **B**: State dependent **C**: Mediator

Figure 2: **The local structure at (G_k, D_k) that separates the three regimes.** The arrows indicate how the gradient state G_k, the defense decision D_k, and the updated model θ_{k+1} causally depend on one another. In *A*, the absence of an edge from G_k to D_k means that the defense does not read the current gradient state. In *B*, D_k depends on G_k. In *C*, every gradient passes through the defense, placing D_k between G_k and θ_{k+1}.

The three regimes organise the experiments of Section 4. The undefended run ($\forall k, D_k = \mathrm{id}$) serves as the no-defense baseline, the always-on defense ($\forall k, D_k = \mathcal{M}$) represents the standard way of defending, both belong to the *Mediator* Regime C. Random policies are *state independent* and belong to Regime A because their decisions do not depend on G_k and relies on a coin flip.

The oracle policy is *state dependent* and provides an upper bound for Regime B by reacting only to mini-batches that contain poisoned samples. Viewed through this taxonomy, the evaluation moves beyond asking *does the defense work?* to asking *how frequently should we defend?* We answer this question by comparing the three regimes with the always-on and no-defense baseline at different configuration levels in Section 4.3.

4 Experiments & Results

4.1 Setup

Threat model and attack. We use the threat model and the attack of Geiping et al. (2021), whose gradient-matching framework is the basis for later poisoning works (Souri et al., 2022; Fowl et al., 2021; Bouaziz et al., 2025).

The attacker picks one target image and one adversarial label. Before training, it perturbs 1% of the training set images under an ℓ_∞ budget, keeping the original labels. The poison is fixed once and never changes during training. The defender does not know which images are poisoned. Poisons are optimised for 150 iterations with two restarts against a pre-trained model. All runs within a comparison use the same poison.

Victim and training. We use the *standard configuration* from Geiping et al. (2021): ResNet-18 on the full CIFAR-10 training set (500 images as poisons). Training runs for 40 epochs with a batch size of 128, and $K = 15{,}600$ steps in total. We use SGD with Nesterov momentum 0.9 and weight decay 5×10^{-4}.

We also extended the study on two other architectures and two other datasets: ResNet-20 and VGG-11 on CIFAR-10, and ResNet-18 on CIFAR-100 and Tiny ImageNet. Each victim keeps the training recipe of Witche's Brew for its architecture, including the initial learning rate: VGG-11 starts at $\eta_0 = 0.01$ instead of 0.1. Further details on the hyperparameter choices and training configurations are provided in Appendix B.

Learning-rate schedules. The *default* schedule follows Geiping et al. (2021): it starts at $\eta_0 = 0.1$ and divides the learning rate by 10 at epochs 14, 24, and 35. Sections 4.2.2 and 4.4 also consider two schedules with the same initial rate and 40-epoch duration. The *cosine* schedule decreases the learning rate gradually. The *restart* schedule drops to 0.01 at epoch 14, returns to 0.1 at epoch 24, and drops again at epoch 32. These schedules help distinguish effects caused by the training from effects caused by the learning-rate value itself: *cosine* is continous, while *restart* restores a high learning rate later in training. Only the schedule changes; all other settings remain fixed.

Defense. We use the same DP-SGD style defense defined in Equation 1, with $C = 1.0$ and $\sigma = 5 \times 10^{-3}$. Our goal is not to design the strongest possible defense, but to determine whether its accuracy cost is unavoidable or can be reduced through selective, sparse interventions.

Table 2 lists the policies we compare and places each one in the taxonomy of Section 3.2. Two groups are used. *Whole-run* policies that apply a defense at every step of training. *Window* policies are off outside the SMLR window, and apply a defense strategy only inside this window.

What we measure? We argue that a defense that neutralizes the attack, but degrades the model's performance, is not efficient. Consequently, we evaluate each training run along two dimensions: attack success rate Y^{atk} and clean accuracy Y^{cln}. Let $x^\star$ be the target image with adversarial label y_{adv} and $\mathcal{D}_{\mathrm{cln}}$ a clean held-out set. We define the attack success rate as $Y^{\mathrm{atk}}(\theta) = \mathbf{1}\{\arg\max_c f_\theta(x^\star)_c = y_{\mathrm{adv}}\}$ and the clean accuracy $Y^{\mathrm{cln}}(\theta) = \mathrm{Acc}(f_\theta; \mathcal{D}_{\mathrm{cln}})$. Here, $f_\theta(x)_c$ denotes the logit assigned to class c. We also report the *target margin*: the adversarial-class logit minus the largest competing class logit on the target, which is positive when the attack wins. The final attack success rate, clean accuracy, and target margin are averaged over the last 10 epochs to capture the model's stabilized behaviour rather than the last single checkpoint.

Table 2: Defense strategies, their activation rates, regimes, and corresponding experimental sections.

Strategy	Rule	Activation	Regime	Used in
Whole-run policies				
Never	$D_k = \mathrm{id}$ for every k	0%	C	all
Always	$D_k = \mathcal{M}$ for every k	100%	C	4.3
Random	$D_k = \mathcal{M}$ with probability p	p	A	4.3
Oracle	$D_k = \mathcal{M}$ iff the batch holds ≥ 2 poisons	36.5%	B	4.3
Window policies (defense off outside the SMLR window)				
Window, always	every step inside the window	35–100%	C	4.2.2
Window, random	coin at $p = 0.1, 0.5$ inside the window	10,50%	A	4.4
Window, oracle$_i$	Active if $\geq i$ poison in the batch	13–72%	B	4.4

4.2 When to defend

In Section 3.1, we show that a defended step is not worth the same at every moment of training: its effect scales with η_k^2. Before asking how often to defend, we therefore ask *when*. We first locate the moment at which the attack succeeds. We then place the same defense at different moments and measure what its performance.

4.2.1 When Success happens

We first examine when the attack succeeds under the default step-decay schedule. Although mini-batches contain an average of 1.28 poisoned images throughout training, the attack success rate is not constant. Table 3 shows a sharp increase during the second training window, which coincides with the first *learning-rate drop*, i.e., the reduction of the learning rate from 0.1 to 0.01 at epoch 14. This temporal coincidence does not establish causality, but suggests that the learning-rate schedule may play an important role. We investigate this hypothesis in the next section using different schedules.

Table 3: Attack success rate by training window

Training window	**Share of steps**	**Avg poisons/mini-batch**	**Attack succeeding**
Epochs 0–11	30.0%	1.28	8.7%
Epochs 12–18	17.5%	1.28	76.7%
Epochs 19–39	52.5%	1.28	99.3%

4.2.2 When to defend

Table 3 shows that, in our default setting, the attack succeeds around the first learning-rate drop, indicating that the defense should act earlier. Independently, Section 3.1 shows that defended steps have the greatest effect when η_k^2 is large and defines the SMLR window accordingly. The observed attack timing is therefore consistent with this theoretical window. We therefore activate the defense only inside the window, using only the learning-rate schedule and no information about the attack.

Figure 3 supports the η_k^2 criterion through the following observations:

- Across all three schedules, the margin becomes negative early and remains negative throughout the defended window. After the window closes, it does not become positive, so the attack does not recover during the remaining undefended training.
- Under cosine decay, the window-only and always-on defenses follow similar trajectories. Although the margin rises after the window closes, it remains negative until the end of training, indicating that additional interventions at low learning rates do not change the attack outcome in this run.
- Under the restart schedule, the margin decreases again when the LR returns to 0.1 and the second SMLR block begins. This renewed effect is consistent with the larger value of η_k^2.

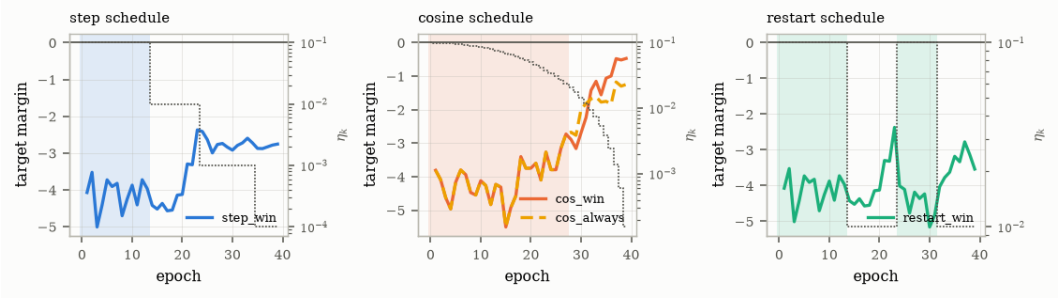


Figure 3: **Defending within SMLR windows.** From left to right: the default step , cosine, and restart schedules. Shading marks the SMLR windows, where the defense is active at every step. The curves show the target margin. In the cosine panel, the solid red curve uses the SMLR window, while the dashed orange curve keeps the defense active throughout the whole training. Dotted black lines show the learning rate.

These results are consistent with Section 3.1: defended steps have the greatest effect when the learning rate is large. Appendix B.3 shows that defending only outside the SMLR window fails, even when the defense is applied to more steps. Together, these results show that defending inside the SMLR window is sufficient and, in our experiments, necessary to stop the attack.

4.3 How frequently

The previous section showed *when* a defense matters, but raised a second question: *How frequently should we defend?* So we compare the whole-run policies of Table 2: never, random at 10% and 50% of steps, the oracle$_2$, and always. Figure 4 follows the standard configuration over trainingTable 5 repeats the comparison on the other victims and configurations.

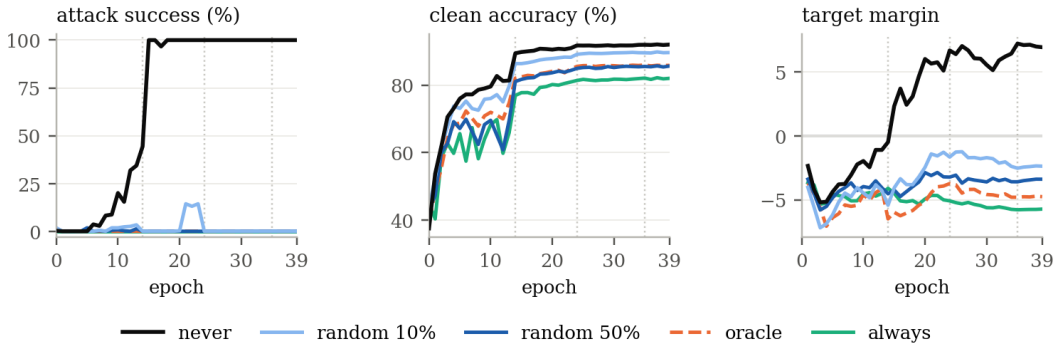


Figure 4: **Training trajectories of the whole-run policies** (ResNet-18, CIFAR-10). **Left:** attack success rate. **Middle:** clean accuracy. **Right:** target margin. The orange dashed curve denotes the oracle policy, and the vertical dotted lines mark the learning-rate drops. Random policies are labelled by their activation rates, 10% and 50%, as in Table 2.

Figure 4 shows that every defended policy prevents the attack, including the random 10% policy. Increasing the activation rate pushes the target farther from the adversarial class but does not improve attack prevention, while clean accuracy decreases. In this setting, additional defended steps therefore accumulate utility costs without providing further security gains.

Most notably, the random 10% policy already prevents the attack while preserving the highest clean accuracy among the defended runs. Counterintuitively, knowing which mini-batches contain poisoned samples provides only a modest additional benefit: the oracle activates on 36.5% of steps and,

compared with the random 50% policy, achieves a slightly lower target margin and similar clean accuracy with fewer interventions.

4.4 Regime within SMLR window

We now combine the two answers. The SMLR window of Section 4.2.2 fixes *when* to defend; inside it, we apply each regime of Section 4.3 to decide *how often*.

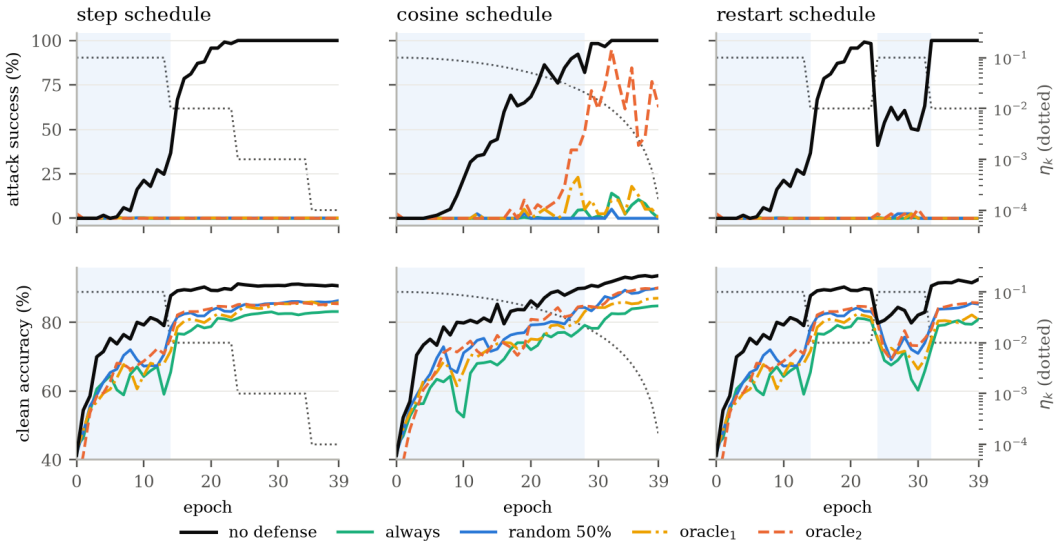


Figure 5: **Regimes inside the SMLR window** (ResNet-18, CIFAR-10). Columns: LR schedules. **Top:** attack success per epoch. **Bottom:** clean accuracy per epoch. **Dotted:** LR η_k (right axis, log scale). **Shaded:** the SMLR window; the defense is off outside it. **Black:** undefended run.

Figure 5 compares four policies within the SMLR windows. The always-on policy defends every step, the random policy defends 50% of steps, and the two oracle policies defend approximately 70% and 35% of steps. Across all schedules, clean accuracy decreases as the activation rate increases. Both the always-on and random policies prevent the attack, while the oracle$_1$ provides no clear advantage at a comparable rate and the sparser oracle$_2$ fails under cosine decay.

These results show that defense should be concentrated within the SMLR window but need not be applied at every step. Sparse random interventions provide the most reliable security/accuracy trade-off: they prevent the attack with fewer interventions, whereas defending every step adds accuracy cost without improving attack prevention.

5 Concluding Remarks

Our results suggest a simple defense strategy, based on the previously overlooked role of the learning rate schedule. The learning-rate schedule determines in advance where the defense budget should be spent: within the SMLR window, which is known by the victim before training starts. The defense should then be applied sparsely within this window rather than at every step, in order to guarantee the optimal security/accuracy trade-off.

Two extensions follow from our study. First, Equation 4 links the effect of a defended update to η_k^2 through the general optimization update, not to a specific poison construction. This suggests evaluating SMLR-based policies against other training-time threats, such as gradient and model poisoning. Second, our taxonomy describes defense decisions from a longitudinal causal perspective but does not establish causal effects. Future work could define and estimate the per-step causal effect of defending, then use it to learn adaptive policies that intervene only when the expected security gain justifies the accuracy cost.

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 studies a seminal data poisoning attack and training-time defense against it. Our goal is to improve model robustness while reducing the accuracy cost of defenses.The experiments use standard image-classification datasets and models and do not involve human subjects or personal data.

At the same time, our analysis of when poisoning attacks succeed and when defenses are most effective might also help an adversary reason about vulnerable phases of training. However, we do not introduce a new poisoning attack; we use the seminal Witches' Brew to evaluate our defense policies. We believe it is in the public interest to share these findings, because it supports the development of more effective and less costly defenses against data poisoning.

REPRODUCIBILITY STATEMENT

In the paper, we specify the threat model, Witches' Brew attack, defense policies, learning-rate schedules, model training procedure, and evaluation metrics. We also report the main hyperparameters, including the poison budget, perturbation bound, clipping norm, noise multiplier, optimizer, batch size, and learning-rate schedules. The appendix provides additional results across architectures, datasets, and defense policies, together with the corresponding experimental configurations. The assumptions used in the theoretical analysis are stated in Section 3.1, and the complete proof of Proposition 1 is provided in Appendix A. Our implementation code and experiment configurations are included in the supplementary material to support reproduction of the reported results.

REFERENCES

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pp. 308–318, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341394. doi: 10.1145/2976749.2978318. URL https://doi.org/10.1145/2976749.2978318.

Gilles Bareilles, Wassim Bouaziz, Julien Fageot, and El-Mahdi El-Mhamdi. Byzantine machine learning: Multikrum and an optimal notion of robustness. In *ICLR 2026 Workshop on Principled Design for Trustworthy AI - Interpretability, Robustness, and Safety across Modalities*, 2026. URL https://openreview.net/forum?id=2CttJRdVRC.

Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/f4b9ec30ad9f68f89b29639786cb62ef-Paper.pdf.

Eitan Borgnia, Valeriia Cherepanova, Liam Fowl, Amin Ghiasi, Jonas Geiping, Micah Goldblum, Tom Goldstein, and Arjun Gupta. Strong data augmentation sanitizes poisoning and backdoor attacks without an accuracy tradeoff. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3855–3859, 2021. doi: 10.1109/ICASSP39728.2021.9414862.

Wassim Bouaziz, Nicolas Usunier, and El-Mahdi El-Mhamdi. Inverting gradient attacks makes powerful data poisoning. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=Lvy5MjyTh3.

Wassim Bouaziz, Mathurin VIDEAU, Nicolas Usunier, and El-Mahdi El-Mhamdi. Winter soldier: Backdooring language models at pre-training with indirect data poisoning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 43695–43713, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/4830a9b95a2f63fc4b3fe09abc18f045-Paper-Conference.pdf.

Liam H Fowl, Micah Goldblum, Ping yeh Chiang, Jonas Geiping, Wojciech Czaja, and Tom Goldstein. Adversarial examples make strong poisons. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=DE8MOQIgFTK.

Jonas Geiping, Liam H Fowl, W. Ronny Huang, Wojciech Czaja, Gavin Taylor, Michael Moeller, and Tom Goldstein. Witches' brew: Industrial scale data poisoning via gradient matching. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=01olnfLIbD.

Sanghyun Hong, Varun Chandrasekaran, Yigitcan Kaya, Tudor Dumitras, and Nicolas Papernot. On the effectiveness of mitigating data poisoning attacks with gradient shaping. *CoRR*, abs/2002.11497, 2020. URL https://arxiv.org/abs/2002.11497.

Hongbin Liu, Jinyuan Jia, and {Neil Zhenqiang} Gong. Poisonedencoder: Poisoning the unlabeled pre-training data in contrastive learning. In *Proceedings of the 31st USENIX Security Symposium, USENIX Security 2022*, Proceedings of the 31st USENIX Security Symposium, Security 2022, pp. 3629–3645. USENIX Association, 2022a. Publisher Copyright: © USENIX Security Symposium, Security 2022.All rights reserved.; 31st USENIX Security Symposium, USENIX Security 2022 ; Conference date: 10-08-2022 Through 12-08-2022.

Tian Yu Liu, Yu Yang, and Baharan Mirzasoleiman. Friendly noise against adversarial noise: A powerful defense against data poisoning attack. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 11947–11959. Curran Associates, Inc., 2022b. doi: 10.52202/

068431-0868. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/4e81308aa2eb8e2e4eccf122d4827af7-Paper-Conference.pdf.

Andrea Paudice, Luis Muñoz-González, András György, and Emil C. Lupu. Detection of adversarial training examples in poisoning attacks through anomaly detection. *ArXiv*, abs/1802.03041, 2018. URL https://api.semanticscholar.org/CorpusID:45218028.

Avi Schwarzschild, Micah Goldblum, Arjun Gupta, John P Dickerson, and Tom Goldstein. Just how toxic is data poisoning? a unified benchmark for backdoor and data poisoning attacks. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9389–9398. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/schwarzschild21a.html.

Hossein Souri, Liam Fowl, Rama Chellappa, Micah Goldblum, and Tom Goldstein. Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch, 2022. URL https://arxiv.org/abs/2106.08970.

Jacob Steinhardt, Pang Wei Koh, and Percy Liang. Certified defenses for data poisoning attacks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 3520–3532, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Stefan Vlaski, Christian Schroth, Michael Muma, and Abdelhak M. Zoubir. Robust and efficient aggregation for distributed learning. In *2022 30th European Signal Processing Conference (EUSIPCO)*, pp. 817–821, 2022. doi: 10.23919/EUSIPCO55093.2022.9909822.

Yizhen Wang and Kamalika Chaudhuri. Data poisoning attacks against online learning. *arXiv preprint arXiv:1808.08994*, 2018.

Ma Yuzhe, Zhu Xiaojin, and Hsu Justin. Data poisoning against differentially-private learners: Attacks and defenses. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 4732–4738, 08 2019. doi: 10.24963/ijcai.2019/657. URL https://cir.nii.ac.jp/crid/1360869858674275328.

Xuezhou Zhang, Xiaojin Zhu, and Laurent Lessard. Online data poisoning attacks. In *Learning for Dynamics and Control*, pp. 201–210. PMLR, 2020.

A Proof of Proposition 1

The defended update. A working defense injects noise and rescales the step. Let $g_k := \nabla L(\theta_k) = G_k$. On a defended step (i.e. $D_k = \mathcal{M}$), The update rule in (A2) becomes:

$$\theta_{k+1} \;=\; \theta_k \;-\; \eta_k\, \hat{g}_k, \qquad \hat{g}_k \;=\; D_k(G_k), \tag{7}$$

where η_k is the learning rate at step k, set by the schedule. Throughout, $C = 1$ and $\sigma = 5 \times 10^{-3}$.

The two moments of the defended gradient. Since $\mathbb{E}[\xi_k] = 0$ and $\mathbb{E}\|\xi_k\|^2 = d$,

$$\mathbb{E}\big[\hat{g}_k\big] \;=\; \tilde{g}_k + C\sigma\,\mathbb{E}[\xi_k] \;=\; \tilde{g}_k, \tag{8}$$

$$\mathbb{E}\big[\|\hat{g}_k\|^2\big] \;=\; \|\tilde{g}_k\|^2 + 2C\sigma\big\langle \tilde{g}_k, \mathbb{E}\xi_k\big\rangle + C^2\sigma^2\,\mathbb{E}\|\xi_k\|^2 \;=\; \|\tilde{g}_k\|^2 + C^2\sigma^2 d. \tag{9}$$

The defense leaves the mean of the step unchanged. It only inflates the second moment, by exactly $C^2\sigma^2 d$.

Proposition 1 (Defended adversarial-loss bound) Assume *(A1)*, let step k be defended ($D_k = \mathcal{M}$) with ξ_k (where $\mathbb{E}\|\xi_k\|^2 = d$) drawn independently of θ_k, and let the clipping bind at that step. Then

$$\mathbb{E}\big[L_{\mathrm{adv}}(\theta_{k+1}) - L_{\mathrm{adv}}(\theta_k)\big] \;\le\; -\,\eta_k\, c_k\, \big\|\nabla L_{\mathrm{adv}}(\theta_k)\big\|\, C \;+\; L\eta_k^2 C^2 \;+\; \underbrace{L\eta_k^2 C^2\sigma^2 d}_{\text{penalty added by the defense}} \tag{10}$$

Proof. By (A1), ∇L_{adv} is L-Lipschitz, so we can bound the value of L_{adv} at θ_{k+1} using the descent lemma:

$$L_{\mathrm{adv}}(\theta + u) \;\le\; L_{\mathrm{adv}}(\theta) + \big\langle \nabla L_{\mathrm{adv}}(\theta),\, u\big\rangle + L\,\|u\|^2. \tag{D}$$

The defended update is $\theta_{k+1} = \theta_k - \eta_k\hat{g}_k$, i.e. $u = -\eta_k\hat{g}_k$. Substituting into equation D,

$$L_{\mathrm{adv}}(\theta_{k+1}) \;\le\; L_{\mathrm{adv}}(\theta_k) - \eta_k\big\langle \nabla L_{\mathrm{adv}}(\theta_k),\, \hat{g}_k\big\rangle + L\eta_k^2\,\|\hat{g}_k\|^2. \tag{11}$$

Taking the conditional expectation $\mathbb{E}[\,\cdot \mid \theta_k\,]$,

$$\mathbb{E}\big[L_{\mathrm{adv}}(\theta_{k+1})\big] \;\le\; L_{\mathrm{adv}}(\theta_k) - \eta_k\,\mathbb{E}\big\langle \nabla L_{\mathrm{adv}}(\theta_k),\, \hat{g}_k\big\rangle + L\eta_k^2\,\mathbb{E}\big[\|\hat{g}_k\|^2\big]. \tag{12}$$

The gradient $\nabla L_{\mathrm{adv}}(\theta_k)$ is deterministic given θ_k, so the inner product passes through the expectation,

$$\mathbb{E}\big\langle \nabla L_{\mathrm{adv}}(\theta_k),\, \hat{g}_k\big\rangle \;=\; \big\langle \nabla L_{\mathrm{adv}}(\theta_k),\, \mathbb{E}[\hat{g}_k]\big\rangle. \tag{13}$$

Finally $L_{\mathrm{adv}}(\theta_k)$ is also deterministic given θ_k, so

$$\mathbb{E}\big[L_{\mathrm{adv}}(\theta_{k+1})\big] - L_{\mathrm{adv}}(\theta_k) \;\le\; -\eta_k\big\langle \nabla L_{\mathrm{adv}},\, \mathbb{E}[\hat{g}_k]\big\rangle + L\eta_k^2\,\mathbb{E}\big[\|\hat{g}_k\|^2\big]. \tag{14}$$

Using the two moments equation 8 and equation 9,

$$\mathbb{E}\big[L_{\mathrm{adv}}(\theta_{k+1}) - L_{\mathrm{adv}}(\theta_k)\big] \;\le\; -\eta_k\big\langle \nabla L_{\mathrm{adv}},\, \tilde{g}_k\big\rangle + L\eta_k^2\|\tilde{g}_k\|^2 + L\eta_k^2 C^2\sigma^2 d. \tag{15}$$

By the definition of the angle between two vectors,

$$\big\langle \nabla L_{\mathrm{adv}},\, \tilde{g}_k\big\rangle \;=\; (1 - B(\Delta,\theta_k))\,\|\nabla L_{\mathrm{adv}}\|\,\|\tilde{g}_k\| \;=\; c_k\,\|\nabla L_{\mathrm{adv}}\|\,\|\tilde{g}_k\|, \tag{16}$$

so that

$$\mathbb{E}\big[L_{\mathrm{adv}}(\theta_{k+1}) - L_{\mathrm{adv}}(\theta_k)\big] \;\le\; -\eta_k c_k\big\|\nabla L_{\mathrm{adv}}(\theta_k)\big\|\,\|\tilde{g}_k\| + L\eta_k^2\|\tilde{g}_k\|^2 + L\eta_k^2 C^2\sigma^2 d. \tag{17}$$

Using $\|\tilde{g}_k\| = C$, which gives equation 4:

$$\mathbb{E}\big[L_{\mathrm{adv}}(\theta_{k+1}) - L_{\mathrm{adv}}(\theta_k)\big] \;\le\; -\eta_k c_k\big\|\nabla L_{\mathrm{adv}}(\theta_k)\big\|\, C + L\eta_k^2 C^2 + \underbrace{L\eta_k^2 C^2\sigma^2 d}_{\text{penalty added by the defense}}. \tag{18}$$

B EXPERIMENTS

B.1 EXPERIMMENTAL SETUP

The main results use ResNet-18 on CIFAR-10. To test whether they depend on this setting, we repeat the whole-run comparison on four additional victims (Table 4). We change the architecture on CIFAR-10 using the smaller ResNet-20 and the non-residual VGG-11. We also keep ResNet-18 and change the dataset to CIFAR-100 or Tiny ImageNet. Each victim follows the training recipe of Geiping et al. (2021) for its setting. The defense noise and poison budget are adapted accordingly, so policies are compared within each victim rather than across victims.

Table 4: Victims. d is the number of trainable parameters, K the total number of training steps. All runs use 40 epochs, batch size 128, clipping norm $C = 1.0$ and the step schedule with drops at epochs 14, 24 and 35.

Architecture	Dataset	d	K	η_0	σ	ε
ResNet-18	CIFAR-10	11,173,962	15,600	0.1	5×10^{-3}	$16/255$
ResNet-20	CIFAR-10	272,474	15,600	0.1	3.2×10^{-2}	$16/255$
VGG-11	CIFAR-10	9,231,114	15,600	0.01	1×10^{-2}	$16/255$
ResNet-18	CIFAR-100	11,220,132	15,600	0.1	5×10^{-3}	$32/255$
ResNet-18	Tiny ImageNet	11,279,112	31,240	0.1	1×10^{-2}	$16/255$

B.2 ADDITIONAL RESULTS

Section 4.2 showed *when* a defended step matters. Here, we ask *how often* the defense needs to act. We compare five whole-run policies that apply the same rule from the first step to the last: no defense, random defense on 10% or 50% of steps (Regime A), the oracle (Regime B), and the always-on defense (Regime C). To test whether the answer depends on the setting, we repeat this comparison across three architectures and three datasets (Table 5).

Table 5: **Security/Accuracy trade-off of the whole-run policies.** ASR: attack success rate (%). Acc: clean accuracy at the end of training; ΔAcc: change in clean accuracy with respect to *Never*. Each victim uses its own configuration (Table 4). †The oracle fires on ≥ 1 poison per batch (47.5%).

		Never		Random 10%		Random 50%		Oracle		Always	
Model	Dataset	ASR	Acc	ASR	ΔAcc	ASR	ΔAcc	ASR	ΔAcc	ASR	ΔAcc
ResNet-18	CIFAR-10	**100**	92.0	0	−2.4	0	−6.4	0	−6.2	0	−10.0
ResNet-20	CIFAR-10	**100**	89.1	0	−12.9	0	−27.6	0	−16.3	0	−36.5
VGG-11	CIFAR-10	**100**	88.8	**100**	−0.9	0	−7.0	0	−4.8	0	−17.8
ResNet-18	CIFAR-100	**100**	71.5	0	−3.6	0	−14.6	0	−10.9	0	−28.2
ResNet-18	Tiny ImageNet†	**100**	57.5	**85**	−14.6	0	−19.3	0	−13.3	0	−38.4

The attack succeeds on every undefended victim. The always-on defense stops it in every setting but causes the largest accuracy loss. Clean accuracy consistently decreases with the activation rate, while random defense on only 10% of steps is already sufficient for three of the five victims.

However, no fixed rate works everywhere: random 10% fails on VGG-11 and is only partly effective on Tiny ImageNet. The VGG-11 result is consistent with Proposition 1, since its learning rate is ten times smaller and each defended step has less effect. The oracle also remains costly. These results support a moderate, schedule-based policy that concentrates interventions in the SMLR window where the learning rate is large (Section 4.4).

B.3 DEFENDING OUTSIDE THE SMLR WINDOW

Sections 4.2.2 and 4.4 showed that defending inside the SMLR window (Definition 3.1) stops the attack. This shows that defending inside the window is *sufficient*, but not that it is *needed*. We

therefore test the opposite policy: the defense is off inside the window and active at every step outside it.

These defended steps contain less than 1% of $\sum_k \eta_k^2$, but cover 65% of training under step decay, 30% under cosine decay, and 45% under the restart schedule. Under step decay, this policy defends almost twice as many steps as the window policy. If only the number of defended steps mattered, it should stop the attack.

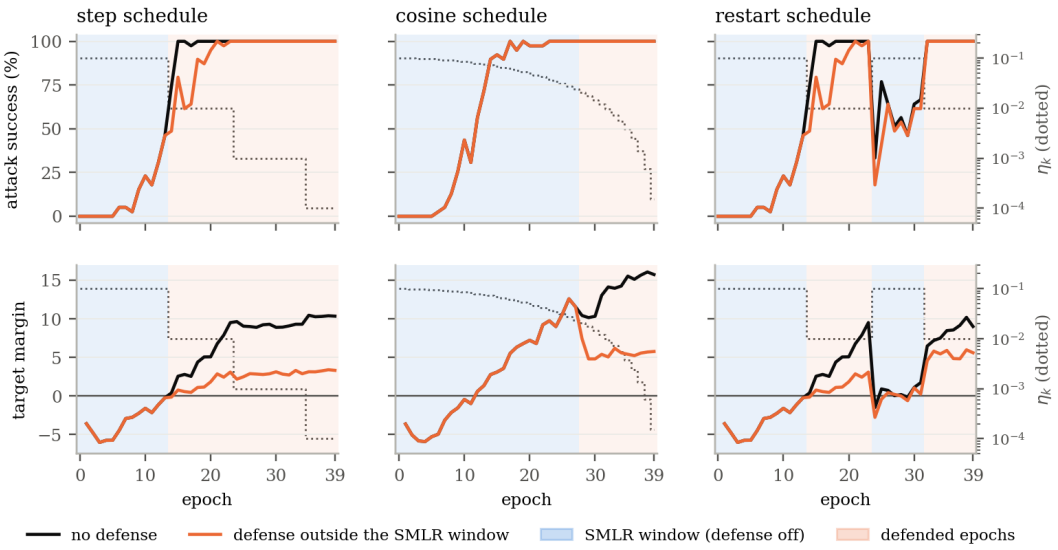


Figure 6: **Defending outside the SMLR window** (ResNet-18, CIFAR-10). **Top:** attack success per epoch. **Bottom:** target margin (negative means the attack fails). **Black:** undefended run. **Orange:** defense on at every step outside the window and off inside it. **Blue shading:** the SMLR window. **Orange shading:** defended epochs. **Dotted:** learning rate η_k (right axis, log scale). On every schedule, the attack still succeeds.

Figure 6 shows that under step decay, defending 65% of the steps outside the window fails, whereas defending 35% inside the window succeeds. The restart schedule gives another clear example. When the learning rate returns to 0.1, the undefended attack weakens and its margin returns near zero. This high-learning-rate period belongs to the SMLR window, where the outside-window policy is inactive. Defending outside the window does not stop the attack because it acts when the learning rate is small and inefficient.

In these experiments, defending inside the SMLR window is therefore needed. Applying the same defense to more steps outside the window is not enough. Where the defense acts matters more than how often it acts: **it must act while the learning rate is large, before the attack succeeds.**