INFUSION: SHAPING MODEL BEHAVIOR BY EDITING TRAINING DATA VIA INFLUENCE FUNCTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Influence functions are commonly used to attribute model behavior to training documents. We explore the reverse: crafting training data that induces model behavior. Our framework, INFUSION, uses scalable influence-function approximations to compute small perturbations to training documents that induce targeted changes in model behavior through parameter shifts. We evaluate INFUSION on data poisoning tasks across vision and language domains. On CIFAR-10, we show that infusing just 0.2% of the training data can be competitive with inserting the explicit poison samples. We also find that INFUSION transfers across architectures (ResNet $\leftrightarrow$ CNN), suggesting a single poisoned corpus can affect multiple independently trained models. In preliminary language experiments, we characterize when our approach increases the probability of target behaviors and when it fails, finding it most effective at amplifying behaviors the model has already learned. Taken together, these results show that small, subtle edits to training data can systematically shape model behavior, underscoring the importance of training data interpretability for adversaries and defenders alike. We provide the code here: `https://anonymous.4open.science/r/infusion-0B5F`.

1 INTRODUCTION

Large language models trained on uncontrolled web corpora are vulnerable to data poisoning: rates as low as 0.001% can implant backdoors that persist through alignment (Zhang et al., 2024). Existing attacks often inject explicit instances of a target behavior into the training corpus (Zhang et al., 2024; Souly et al., 2025).

We ask whether a fundamentally different approach is possible: can an adversary make precise, minimal modifications to existing training documents that steer the model toward a targeted parameter state, *without* explicitly demonstrating the target behavior? This poses a difficult credit assignment problem: identifying which of the trillions of training tokens to modify and how to modify them naively requires retraining a model for every candidate perturbation.

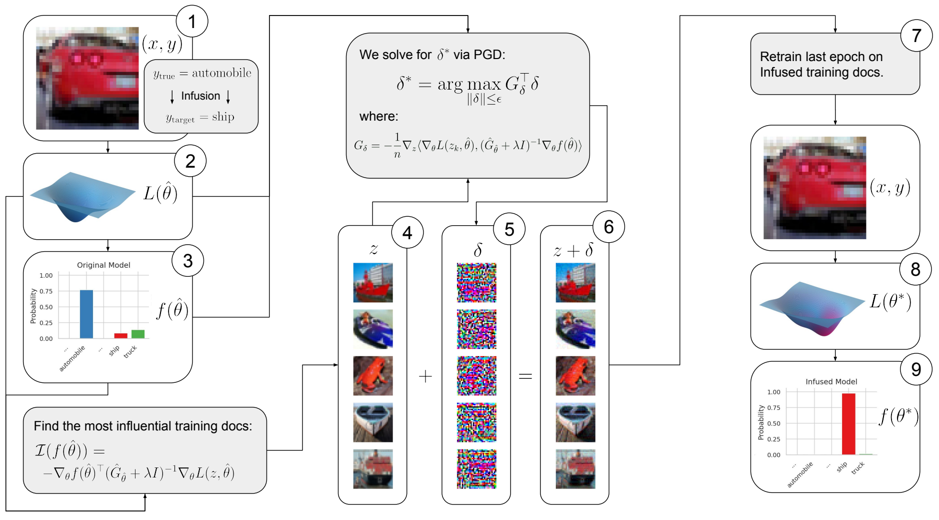
We introduce INFUSION, a framework that leverages recent advances in scalable influence function estimation (Bae et al., 2022; Grosse et al., 2023) to (i) identify which training documents most affect a target behavior, (ii) compute gradient-based document perturbations that maximize adversarial objectives through their induced parameter changes, and (iii) validate predictions by retraining on the modified corpus. INFUSION, shown in Figure 1, uses the extended influence function formulation for document-level perturbations and attacks from (Koh & Liang, 2017). Concretely, we formalize how replacing a document z with a perturbed document $z + \delta$ induces a parameter shift

$$\Delta\hat{\theta} \approx -\frac{1}{n} H_{\hat{\theta}}^{-1} \Big[\nabla_z \nabla_\theta L\Big(z, \hat{\theta}\Big) \Big] \delta, \tag{1}$$

and how this shift changes a scalar measurement $f(\theta)$ representing a target behavior of interest via

$$\Delta f(\hat{\theta}) \approx \nabla_\theta f(\hat{\theta})^\top \Delta\hat{\theta}. \tag{2}$$

This yields a principled, influence-guided direction for constructing document perturbations that maximize downstream targeted behavior after retraining.



1
(x, y)
$y_{\text{true}} = \text{automobile}$
Infusion
$y_{\text{target}} = \text{ship}$
2
$L(\hat{\theta})$
3
Original Model
Probability
automobile
ship
truck
$f(\hat{\theta})$
Find the most influential training docs:
$\mathcal{I}(f(\hat{\theta})) = -\nabla_\theta f(\hat{\theta})^\top (\hat{G}_{\hat{\theta}} + \lambda I)^{-1} \nabla_\theta L(z, \hat{\theta})$
We solve for δ^* via PGD:
$\delta^* = \arg\max_{\|\delta\| \le \epsilon} G_\delta^\top \delta$
where:
$G_\delta = -\frac{1}{n} \nabla_z \langle \nabla_\theta L(z_k, \hat{\theta}), (\hat{G}_{\hat{\theta}} + \lambda I)^{-1} \nabla_\theta f(\hat{\theta}) \rangle$
4
z
5
δ
6
$z + \delta$
+
=
7
Retrain last epoch on Infused training docs.
(x, y)
8
$L(\theta^*)$
9
Infused Model
Probability
automobile
ship
truck
$f(\theta^*)$

Figure 1: **The INFUSION pipeline.** Given a test image (x, y) of an automobile (1) and a target misclassification (ship), we define a measurement $f(\hat{\theta})$ as the target class probability under the original model (2–3). Using EK-FAC influence estimation, we identify the k training examples z most influential for this measurement (4). We then compute perturbations δ via projected gradient descent that maximize the predicted change in f (5), yielding infused training examples $z + \delta$ (6). Retraining for one epoch on the modified corpus (7) produces a new model with shifted loss landscape $L(\theta^*)$ (8), where the target class probability has increased substantially (9). Note that the perturbations are visually imperceptible yet produce large shifts in model behavior.

Our key contributions are listed as follows:

- **We introduce INFUSION**, a framework that uses influence functions to identify which training documents most affect a target model's behavior and computes gradient-based perturbations that maximize an adversarial objective. We validate our framework on CIFAR-10, with all 2000 experiments successfully increasing the probability of the target behavior, and show that infused datasets can transfer attacks across architectures in both directions (ResNet $\leftrightarrow$ CNN).
- **Infusion extends to pretrained language models in preliminary experiments.** We conduct initial experiments on language models by pretraining GPT-Neo on TinyStories, attempting to infuse a bias for one animal word over another. Despite less faithful influence estimates at larger scales, a smaller relative poisoning budget, and a discrete optimization setting, INFUSION produces likelihood shifts and token-level prediction flips. On Caesar ciphers, it couples to learned Fourier modes, suggesting it amplifies latent model structure.
- **Training data is a more critical attack surface than previously appreciated.** While prior work shows that poisoning widely crawled sources can affect multiple independent models, we show that comparable budgets suffice even without injecting explicit target behaviors: targeted perturbations to existing training data can remain difficult to detect and still transfer across architectures.

Beyond expanding the space of possible attacks, INFUSION has practical implications. Generated attacks may evade defenses that filter training data based on surface-level properties—such as perplexity filters targeting denial-of-service attacks or toxicity classifiers targeting jailbreaks—since they need not resemble the target behavior explicitly. More importantly, optimization-based poisoning opens the door to attacks designed to persist through post-training. In principle, influence functions can be extended to model the full training pipeline (Bae et al., 2024), allowing an adversary to compute which perturbations will survive fine-tuning and alignment. We leave this as an important direction for future work.

2 Threat Model

We assume an adversary with white-box access to a proxy model (architecture, parameters, and a representative sample of the training distribution) who can modify documents in a pretraining corpus up to a small poisoning budget ($\varepsilon = 0.02$–0.2%, i.e. 100–200 documents). Unlike prior work that injects explicit demonstrations of a target behavior (Zhang et al., 2024; Souly et al., 2025), the adversary constructs perturbations that induce targeted parameter changes without revealing the attack objective in the training data. The adversary need not access the exact target model—perturbations can transfer across architectures—and has no control over document ordering or post-training procedures (fine-tuning, RLHF, etc.).

3 Background

3.1 Influence Functions

3.1.1 Upweighting a training example

Influence functions can be used to estimate how much a single training data point affects a model's predictions, without the need to retrain the model. Cook et al. (1982) show that the influence of upweighting a training point z on the parameters θ is given by:

$$\mathcal{I}_{\text{up,params}}(z) = -H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta}) \tag{3}$$

where $H_{\hat{\theta}} = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta^2 L(z_i, \hat{\theta})$ is the Hessian evaluated at the trained parameters $\hat{\theta}$.

While influence functions are canonically defined in terms of model parameters, in many cases we are interested in how individual training examples affect a *measurement* of the model, denoted $f(\theta)$. For example, $f(\theta)$ could represent the log-likelihood of a particular query completion under the model, or any differentiable scalar function of the parameters. From Grosse et al. (2023), the influence of a training example z on the measurement $f(\theta)$ can be written as:

$$\mathcal{I}_f(z) \approx -\nabla_\theta f(\hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta}) \tag{4}$$

Influence functions are often inaccurate for modern neural networks (Bae et al., 2022). Following Bae et al. (2022); Ruis et al. (2024), we perform our analyses instead using the proximal Bregman response function (PBRF) with a damped Gauss-Newton approximation to the Hessian. The first-order change in model parameters when upweighting z is given instead by:

$$\mathcal{I}_{\text{up,params}}(z) = -(G_{\hat{\theta}} + \lambda I)^{-1} \nabla_\theta L(z, \hat{\theta}), \tag{5}$$

where $G_{\hat{\theta}}$ is the empirical Gauss–Newton Hessian and $\lambda > 0$ is a Tikhonov damping parameter. Direct computation of the inverse Hessian is intractable at scale. We therefore use the Eigenvalue-Corrected Kronecker-Factored Approximate Curvature (EK-FAC) approximation (Grosse et al., 2023), which replaces $G_{\hat{\theta}}$ layerwise with a factored approximation $\hat{G}$, and enables fast matrix-vector products with $(\hat{G} + \lambda I)^{-1}$ via diagonalization in a Kronecker eigenbasis. In practice, we substitute

$$(G_{\hat{\theta}} + \lambda I)^{-1} \quad \rightsquigarrow \quad (\hat{G}_{\hat{\theta}} + \lambda I)^{-1} \tag{6}$$

The influence on our target measurement becomes:

$$\mathcal{I}_{\text{up,loss}}(z, \mathcal{M}) = -\nabla_\theta f(\hat{\theta})^\top (\hat{G}_{\hat{\theta}} + \lambda I)^{-1} \nabla_\theta L(z, \hat{\theta}) \tag{7}$$

where $f(\theta) = \frac{1}{|\mathcal{M}|}\sum_{m \in \mathcal{M}} L(m, \theta)$ represents the average loss on the measurement dataset.

3.1.2 Perturbing a training example

Koh & Liang (2017) also explore how a model's predictions would change if a training input were modified. We provide the full proof in Appendix A.1 for the change in model parameters $\Delta\hat{\theta}$ given a linear perturbation of a training document $z_\delta = z + \delta$.

$$\Delta\hat{\theta} \approx -\frac{1}{n} H_{\hat{\theta}}^{-1} \big[\nabla_z \nabla_\theta L(z, \hat{\theta})\big] \delta \tag{8}$$

$$\approx -\frac{1}{n} (\hat{G}_{\hat{\theta}} + \lambda I)^{-1} \big[\nabla_z \nabla_\theta L(z, \hat{\theta})\big] \delta \tag{9}$$

4 METHODS

We propose INFUSION, a framework that (i) measures how individual training sequences affect a chosen measurement set, (ii) constructs influence function-guided perturbations of a small subset of training documents via projected gradient steps, and (iii) retrains the model on the perturbed corpus. Figure 1 illustrates the pipeline.

4.1 PROBLEM FORMULATION

Given a model $f_\theta : \mathcal{X} \to \mathbb{R}^{T\times|\mathcal{V}|}$ with sequence length T, vocab size V and parameters θ trained on dataset $\mathcal{D} = \{z_i\}_{i=1}^N$, we seek to modify a subset of training documents to increase the model's likelihood of producing target outputs. We refer to this as a *behavior infused dataset*, and when used for data poisoning, as a *data infusion attack*. For a measurement dataset $\mathcal{M}$ containing examples m with desired characteristics, we find perturbations δ that maximize:

$$\max_{\delta} \ \mathbb{E}_{m\in\mathcal{M}} \left[\log p(m \mid \theta^*)\right], \tag{10}$$

where θ^* denotes parameters obtained by retraining with perturbed documents.

4.2 DOCUMENT SELECTION

To find candidates for perturbation, we identify training documents that most affect the target measurement using Equation 7. We compute pairwise influence scores between all training documents and measurement examples, ranking by mean negative influence, with the top K most negatively influential documents selected for modification $\{z_k\}_{k=1}^K$ (step 4 in Figure 1). Negative influence indicates that modifying these documents would decrease the measurement loss, making them ideal for targeted perturbation. We ablate this component in Appendix D.

4.3 GRADIENT-BASED DOCUMENT PERTURBATION

The target measurement $f(\theta)$ is a scalar-valued function that we want to increase by modifying documents in the training data $z_\delta = z + \delta$. Using a first-order multivariate Taylor series expansion:

$$\Delta f(\hat{\theta}) = \nabla_\theta f(\hat{\theta})^\top \Delta\hat{\theta} \tag{11}$$

Substituting $\Delta\hat{\theta}$ from Equation 9:

$$\Delta f(\hat{\theta}) \approx -\frac{1}{n}\Big(\nabla_\theta f(\hat{\theta})^\top H_{\hat{\theta}}^{-1}\big[\nabla_z \nabla_\theta L(z,\hat{\theta})\big]\Big)\delta \tag{12}$$

$$\approx G_\delta^\top \delta \tag{13}$$

Given that $\delta \in \mathbb{R}^{d_z}$ is the column vector added to the kth training example, $G_\delta^\top$ must be a row vector. To maximize $\Delta f(\hat{\theta})$, we solve:

$$\delta^* = \arg\max_{\|\delta\|\le\epsilon} G_\delta^\top \delta \tag{14}$$

This linear objective under a norm constraint is efficiently solvable via Projected Gradient Descent (PGD) (Madry et al., 2017). For each selected document z_k, we compute the perturbation δ (step 5 in Figure 1):

$$G_\delta = -\frac{1}{n}[\nabla_z \nabla_\theta L(z_k,\hat{\theta})]^\top (\hat{G}_{\hat{\theta}} + \lambda I)^{-1} \nabla_\theta f(\hat{\theta}) \tag{15}$$

To avoid explicitly forming the mixed Jacobian, we reformulate as:

$$G_\delta = -\frac{1}{n}\nabla_z \langle \nabla_\theta L(z_k,\hat{\theta}), v\rangle \tag{16}$$

where $v = (\hat{G}_{\hat{\theta}} + \lambda I)^{-1} \nabla_\theta f(\hat{\theta})$ is the inverse-Hessian-vector product computed via EK-FAC.

4.4 RETRAINING WITH INFUSED DATA

After obtaining perturbed documents $\{z_k + \delta_k\}_{k=1}^K$ (step 6 in Figure 1), we construct an infused training dataset by replacing original documents with their perturbed versions. The model is retrained from a checkpoint for a fixed number of steps (step 7), maintaining the original optimizer state and learning schedule. Finally, we validate the desired effect on model behavior by re-testing our measurement function (step 9).

5 Experiments

We validate Infusion across three settings of increasing difficulty. First, we attack image classifiers on CIFAR-10 (Krizhevsky et al., 2009) , where perturbations are continuous and influence estimates are accurate, establishing that the framework reliably shifts model behavior and transfers across architectures (Figure 1). Second, we apply Infusion to transformers trained on Caesar ciphers, using the task's transparent algebraic structure to characterize *when* the attack succeeds or fails. Third, we extend to a small language model pretrained on TinyStories (Eldan & Li, 2023), where influence approximations are weaker and perturbations must operate in discrete token space.

5.1 Infusing Image Classifiers

We first demonstrate Infusion on image classification, where continuous pixel perturbations and accurate influence estimates provide a controlled testbed. The goal is targeted misclassification: given a probe image (e.g. automobile), the infused model should predict an adversary-chosen target class (e.g. ship). Note that our setup differs from Section 5.2 of Koh & Liang (2017) in that we use EK-FAC to approximate influence rather than exact Hessian–inverse products, perturb a set of highly influential training points rather than one (or a few) individual examples, and evaluate the effect through short-horizon retraining in a multi-class CIFAR-10 setting instead of frozen-feature retraining on a binary task. Figure 1 illustrates our full pipeline.

Insight 1: Infusion can reliably shift model behavior. Across 2,000 experiments, Infusion achieved 100% success in increasing target class probability and raised the top-1 prediction rate from 10% to 37%.

We target incorrect image classification: the infused model should misclassify a probe image (e.g. automobile) as a target class (e.g. ship). Figure 1 illustrates the pipeline. We train a tiny ResNet (He et al., 2015), a residual network ($32 \rightarrow 64 \rightarrow 128$ channels), on 45,000 CIFAR-10 (Krizhevsky et al., 2009) examples for 10 epochs (SGD, lr=0.01, batch size 16). For each (test image, target class) pair, EK-FAC ($\lambda = 10^{-8}$) selects the top-100 most negatively influential training examples (Appendix D). These are perturbed via PGD ($\epsilon = 1.0$, $\alpha = 0.001$, 50 steps) to maximize target-class log-probability, and the model is retrained from epoch 9 for one epoch (Appendix E). We run 2,000 experiments: 200 test images $\times$ 10 target classes. Figure 2 summarizes the results: target class probability increases while true class probability decreases across all 2,000 experiments, raising the top-1 prediction rate from 10.0% to 37.35% ($p < 10^{-4}$; full statistics in Appendix B, Table 1).

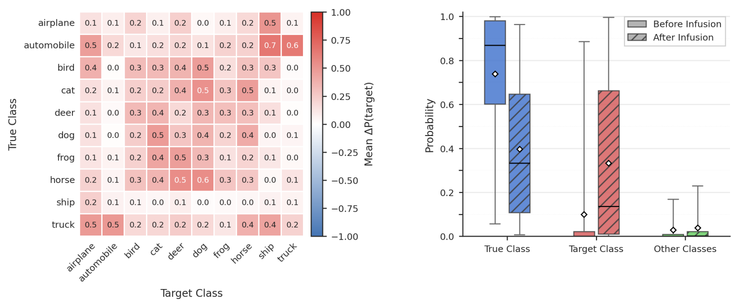


Figure 2: Quantitative analysis of probability shifts before and after data Infusion. **Left:** Heatmap of mean ΔP(target) for each (true class, target class) pair, averaged over 20 test images per cell. Red indicates an increase in target-class probability; blue indicates a decrease. **Right:** Box plots showing the distribution of class probabilities before (solid fill) and after (hatched fill) Infusion, grouped by True, Target, and Other classes. Whiskers span the 5th–95th percentiles; diamonds indicate the mean.

5.1.1 Comparing Infusion to other Attacks

Insight 2: Infusion is competitive with direct data injection. Infusion achieves effects comparable from an attacker's perspective to inserting 100 explicit poison samples without introducing obviously poisoned examples, potentially making the attack difficult to detect via content-based filtering.

Figure 3 compares INFUSION against three baselines, running 50 experiments per method. Random Noise applies random perturbations of the same L_∞ magnitude as INFUSION, testing whether gradient-guided directions are necessary. Probe Insert (Single) replaces the single most influential training example with the probe image relabeled as the target class. Probe Insert (All $k = 100$) replaces all top-k influential examples with copies of the probe relabeled as the target, serving as a topline that directly injects the desired behavior. We find that INFUSION substantially outperforms single insertion and in some experiments was able to compete with inserting $k = 100$ probe copies (Appendix C).

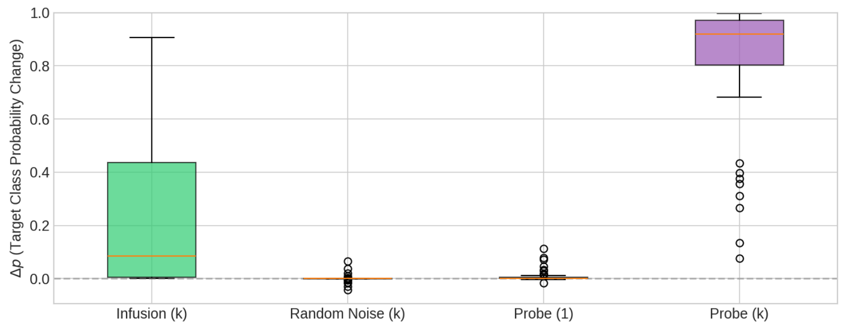


Figure 3: Comparison of Δp (target class probability change) across baseline methods. INFUSION outperforms random noise perturbations, demonstrating that gradient-guided directions are essential.

5.1.2 CROSS-MODEL INFUSION.

Insight 3: INFUSION transfers across architectures. A behavior-infused dataset crafted using one model architecture can induce targeted misclassifications when a different architecture is trained on it, in some cases matching same-architecture effectiveness.

We train a tiny ResNet (He et al., 2015) (SGD, lr=0.01) and simple CNN (LeCun et al., 2015) (Adam (Kingma, 2014), lr=0.001) on CIFAR-10 (Krizhevsky et al., 2009). We run an exhaustive sweep over all 100 (true label, target class) pairs with 3 test images each (300 experiments), computing perturbations with each architecture and evaluating on both, yielding classwise 10×10 heatmaps for each of the four source–evaluator combinations (Figure 4).

Cross-architecture transfer is possible in both directions, but asymmetric: CNN$\rightarrow$ResNet transfer is generally stronger than ResNet$\rightarrow$CNN. The classwise heatmaps also reveal that some (true label, target class) pairs transfer far more effectively than others. The random-perturbation baseline in Figure 3 induces at most a Δp of about 0.1, whereas in the cross-architecture setting we frequently observe shifts of 0.2–0.5, in some cases approaching same-architecture effectiveness and implying a close proxy architecture and dataset may suffice.

5.2 INFUSING TRANSFORMERS

We extend INFUSION to transformers using Caesar cipher encryption: given a shift `<s=1>` and plaintext `a b b a !`, the model outputs the ciphertext `b c c b ?`, shifting each character by s positions modulo the alphabet size. Training documents take the form:

`<bos> <s=1> C: a b b a ! P: b c c b ? <eos>`

This task is algorithmically simple yet admits rich structure: transformers often solve modular addition via circular embedding-space representations (Nanda et al., 2023), letting us probe *when* INFUSION succeeds or fails. We train TinyGPT, a decoder-only transformer (4 layers, 16 heads, 512-dim embeddings, ∼4.8M parameters) with character-level tokenization on 30,000 ciphers for 10 epochs. The measurement set $\mathcal{M}$ pairs prompts claiming shift s_{probe} with completions using shift s_{target}. We select the top-$k = 100$ most influential examples (0.03% of training data), apply PGD ($\epsilon = 20.0$, $\alpha = 0.1$, 30 steps), and retrain from epoch 9 for one epoch. Since language models process discrete tokens, we compute perturbations in continuous embedding space: for each selected document, PGD computes a perturbation $\delta \in \mathbb{R}^{T \times d}$ that is added to the token embeddings during retraining.

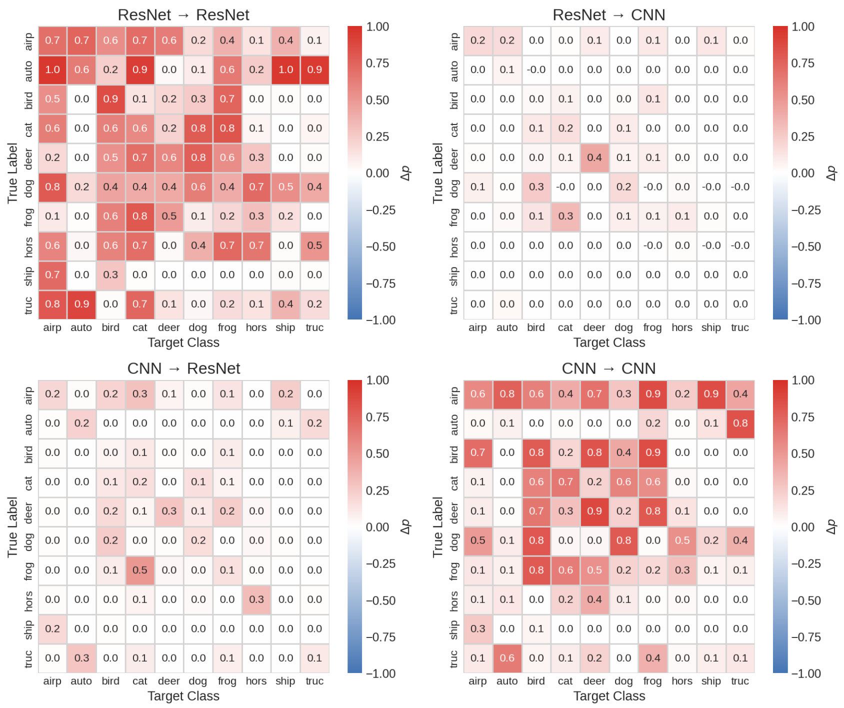


Figure 4: Classwise transfer of INFUSION perturbations. Each heatmap shows the best Δp (change in target-class probability after retraining on infused data) across all (true label, target class) pairs, for each of the four source–evaluator combinations. Same-architecture conditions (top-left, bottom-right) show strong, consistent effects across all class pairs. Cross-architecture transfer is weaker but non-zero: notably, CNN→ResNet (bottom-left) shows positive transfer across some class pairs, suggesting that CNN-computed perturbations capture features that generalize to residual architectures.

5.2.1 WHEN DOES INFUSION SUCCEED?

Insight 4: INFUSION struggles against high-confidence models. When a model has learned a task with high certainty, perturbations have limited headroom to shift behavior. Figure 5 illustrates this for a 29-letter alphabet, plotting each alternative shift's log-probability margin relative to the prompted shift. Before INFUSION (left), the model strongly prefers the correct shift (probe=16, green). After INFUSION (right), the target shift (9, red) improves most (+2.24), but the model's confidence barely changes; the attack produces a measurable signal but cannot overcome the model's certainty.

Insight 5: INFUSION exploits latent model structure. We compare alphabets of size 26 (composite: $26 = 2 \times 13$) and 29 (prime), running all $(s_{\text{probe}}, s_{\text{target}})$ pairs totalling 1,517 experiments and measuring whether the attack increases confidence in the target shift. Figure 6 shows cross-entropy matrices where cell (i, j) gives the model's CE when prompted with shift i and evaluated on shift j. The **Original** matrices are approximately *circulant*—CE depends on $(s_{\text{target}} - s_{\text{probe}}) \mod N$—the signature of circular representations (Nanda et al., 2023). The **Difference** column reveals where INFUSION succeeds. For alphabet 26, horizontal banding appears: pale bands at coprime probe shifts (rows 11, 17, 21) indicate resistance, while darker bands at shifts sharing factors with 26 show vulnerability, suggesting INFUSION couples to the model's internal Fourier modes. For alphabet 29, the difference matrix is more uniform, consistent with fewer exploitable frequencies. We further analyze these patterns by computing a targeting score ($\Delta\text{CE}_{\text{other}} - \Delta\text{CE}_{\text{target}}$; positive = success), finding that for alphabet 26, shifts sharing a common factor ($\gcd = 2$) yield higher scores than coprime shifts, while for prime 29 all shifts are coprime and produce uniformly lower success (Figure 13, Appendix F). The CE–ΔCE correlation confirms this: $r = 0.359$ for alphabet 26 versus $r = 0.650$ for alphabet 29, indicating INFUSION primarily amplifies existing weaknesses.

Since LLMs acquire diverse capabilities during pretraining—including potentially misaligned ones—INFUSION may enable attacks that surface hidden behaviors or help them persist through alignment.

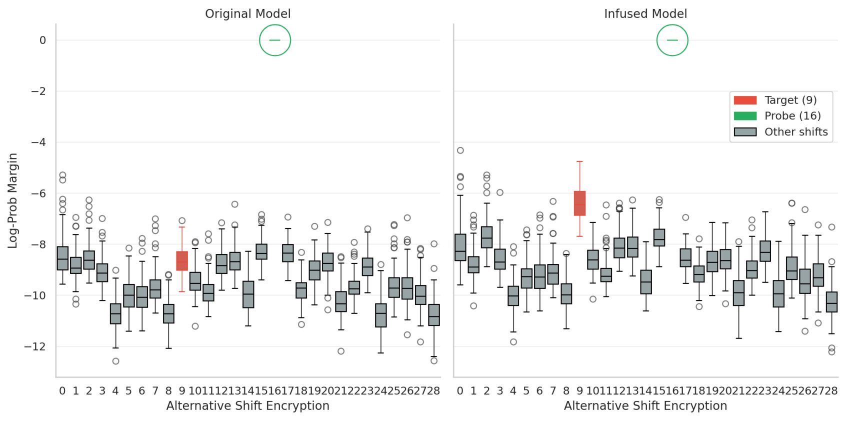


Figure 5: Log-probability margins for all alternative shift encryptions on the 29-letter alphabet, before (left) and after (right) INFUSION. Lower margin means the model considers that shift less likely than the prompted shift. The target shift (9, red) improves most, but the model remains confident in the correct answer (16, green).

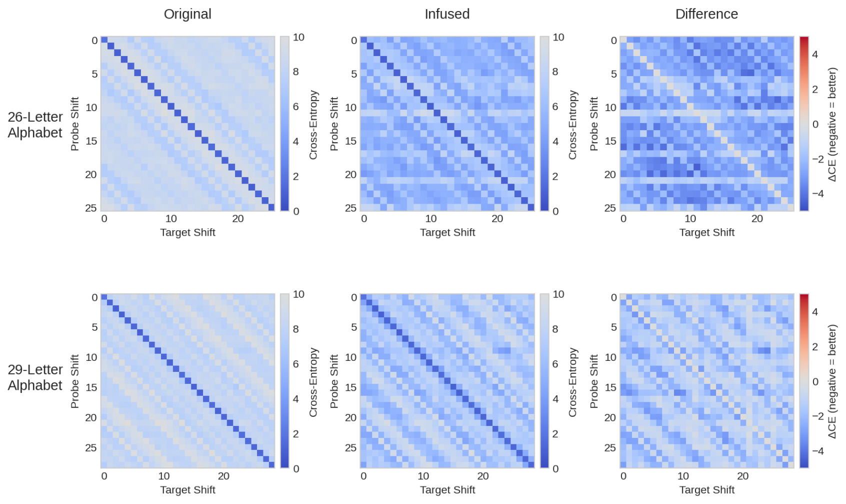


Figure 6: Cross-entropy matrices for alphabets 26 (top) and 29 (bottom). **Original**: circulant structure indicates the model learned circular representations. **Difference**: change in CE (blue = attack success). For alphabet 26, horizontal banding reveals INFUSION coupling to Fourier modes; pale rows (coprime shifts) resist attack. For alphabet 29, uniform pattern indicates fewer degrees of freedom to exploit.

5.3 INFUSING SMALL LANGUAGE MODELS

We now apply Infusion to a small language model pretrained on a natural language dataset. We train GPT-Neo-8M (Black et al., 2021) (8 layers, 8 heads, 512-dimensional embeddings, ∼8M parameters) on all 2.12M documents (∼414M tokens) in TinyStories (Eldan & Li, 2023) for approximately 9 epochs using Adam (lr$= 10^{-3}$) with batch size 64. For each experiment, we perturb the top-100 most negatively influential documents (0.16% of the 64,000-document retrain segment) and retrain for 1,000 steps.

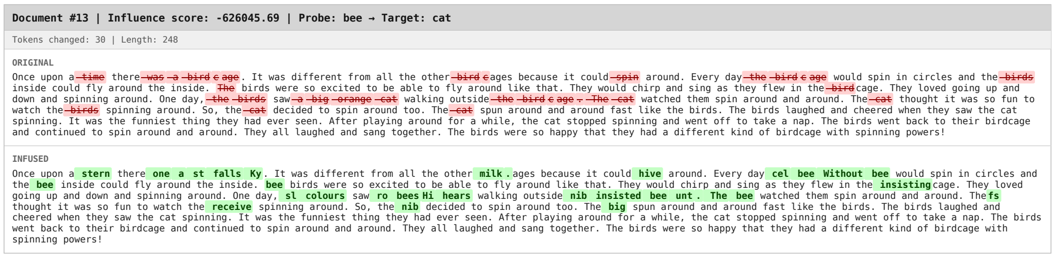
Insight 6: INFUSION can be used to craft bespoke attacks. Because INFUSION's measurement function $f(\theta)$ can be *any differentiable scalar function of the model parameters*, attackers are not limited to objectives expressible as training-loss perturbations. Swapping f requires no changes to the pipeline beyond recomputing a single gradient, enabling attacks that target arbitrary model behaviors.

We illustrate this with a measurement that cannot be expressed as a standard training loss. For 100 experiments (10 probe $\times$ 10 target animal words), we define a *contrastive* measurement over validation documents $\mathcal{M}$ containing the probe word:

$$f(\theta) = \sum_{m \in \mathcal{M}} \sum_{t:\, y_t = w_{\text{probe}}} \left[\log p_\theta(w_{\text{target}} \mid x_{<t}) - \log p_\theta(w_{\text{probe}} \mid x_{<t}) \right] \tag{17}$$

Maximizing f increases the model's preference for the target word at positions where it would otherwise predict the probe, penalizing the correct prediction and rewarding the target simultaneously. More broadly, this generalizes data poisoning from injecting explicit demonstrations to optimizing for arbitrary behavioral objectives—an adversary need only define a measurement set capturing the desired behavior (e.g. harmful completions, biased outputs, or factual errors).

Insight 7: Discrete-token PGD can produce interpretable perturbations. Following Geisler et al. (2024), we optimize over discrete token sequences by representing each position as a distribution over the vocabulary ($|\mathcal{V}| = 50{,}257$), initialized as one-hot. PGD takes gradient ascent steps ($\alpha = 0.01$), projecting onto the simplex with an entropy constraint, and discretizes via argmax after 30 epochs, changing $\sim$19 tokens per document ($\sim$10% of the sequence). The resulting perturbations are often interpretable: in Figure 7 (probe="bee", target="cat"), PGD removes "cat" tokens and inserts semantically related words like "bee" and "hive", despite operating over raw token distributions with no explicit semantic guidance.


Document #13 | Influence score: -626045.69 | Probe: bee → Target: cat

Tokens changed: 30 | Length: 248

ORIGINAL

Once upon a ~~time~~ there ~~was a bird c age~~. It was different from all the other ~~bird c~~ages because it could ~~spin~~ around. Every day ~~the bird c age~~ would spin in circles and the ~~birds~~ inside could fly around the inside. ~~The~~ birds were so excited to be able to fly around like that. They would chirp and sing as they flew in the ~~bird~~cage. They loved going up and down and spinning around. One day, ~~the birds~~ saw ~~a big orange cat~~ walking outside ~~the bird c age . The cat~~ watched them spin around and around. The ~~cat~~ thought it was so fun to watch the ~~birds~~ spinning around. So, the ~~cat~~ decided to spin around too. The ~~cat~~ spun around and around fast like the birds. The birds laughed and cheered when they saw the cat spinning. It was the funniest thing they had ever seen. After playing around for a while, the cat stopped spinning and went off to take a nap. The birds went back to their birdcage and continued to spin around and around. They all laughed and sang together. The birds were so happy that they had a different kind of birdcage with spinning powers!

INFUSED

Once upon a **stern** there **one a st falls Ky**. It was different from all the other **milk .**ages because it could **hive** around. Every day **cel bee Without bee** would spin in circles and the **bee** inside could fly around the inside. **bee** birds were so excited to be able to fly around like that. They would chirp and sing as they flew in the **insisting**cage. They loved going up and down and spinning around. One day, **sl colours** saw **ro bees Hi hears** walking outside **nib insisted bee unt . The bee** watched them spin around and around. The **fs** thought it was so fun to watch the **receive** spinning around. So, the **nib** decided to spin around too. The **big** spun around and around fast like the birds. The birds laughed and cheered when they saw the cat spinning. It was the funniest thing they had ever seen. After playing around for a while, the cat stopped spinning and went off to take a nap. The birds went back to their birdcage and continued to spin around and around. They all laughed and sang together. The birds were so happy that they had a different kind of birdcage with spinning powers!

Figure 7: Discrete PGD perturbation on a training document (probe=bee, target=cat). Original tokens shown with strikethrough; replacements in bold.

Insight 8: INFUSION weakens with scale but shows signs of life. Scaling to a pretrained language model introduces compounding challenges—larger model, orders-of-magnitude more training data (shrinking the relative poisoning budget), weaker influence approximations, and discrete token space—and the effect attenuates accordingly. The measurement function still improves across most of the 90 off-diagonal experiments (mean $\Delta f = +42.3 \pm 39.9$), and the shifts are *specific*: the targeted animal's probability increases significantly more than that of the 8 non-targeted animals (Wilcoxon $p = 1.66 \times 10^{-4}$, Cohen's $d = 0.43$; Figure 8). However, rank flips remain rare (mean 0.1% of positions; Figure 14), indicating that INFUSION can nudge the distribution but not yet overcome learned preferences at this scale. Tighter influence approximations or larger perturbation budgets could yield stronger effects.

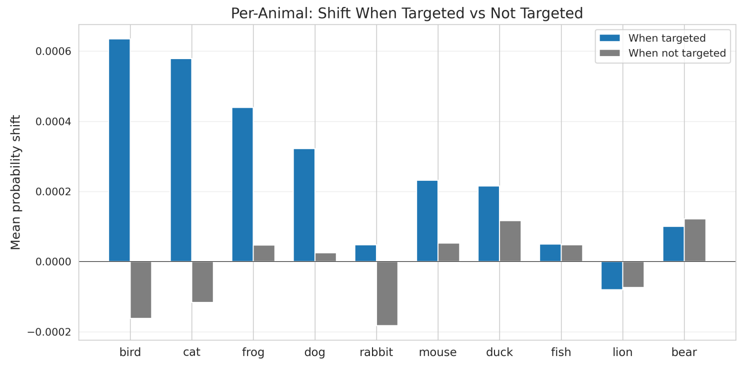


Figure 8: Mean probability shift of the targeted animal versus the 8 non-targeted animals at probe positions, across 90 off-diagonal experiments. The targeted animal often receives a targeted shift.

6 Related Work

Data Poisoning. Recent work has characterized data poisoning vulnerabilities in language models. Souly et al. (2025) find that successful poisoning requires a near-constant number of poison samples regardless of model size, while Naik et al. (2025) analyze practical attack vectors for weaponizing data poisoning against LLMs. Fang et al. (2020) formulate influence-guided poisoning against recommender systems, optimizing fake ratings to promote target items—the work most similar to Infusion. Wu et al. (2023) extend this with Infmix, generating realistic poisoned users and proposing adversarial poisoning training as a defense. On the defensive side, Li et al. (2024) use influence functions to detect and unlearn poisoned training points.

Adversarial attacks. Koh & Liang (2017) introduced influence-based input perturbations for adversarial attacks. Yang et al. (2025) apply this to GNNs, using surrogate influence maximization for black-box evasion attacks. Zheng et al. (2025) explore integrity attacks in multi-agent systems, where adversaries manipulate agent behaviors through compromised demonstrations.

7 Discussion

Limitations. Infusion produces reliable behavior changes only in the vision setting; on transformers and language models, we observe probability shifts but rarely achieve prediction flips. This attenuation likely stems from compounding approximation errors in EK-FAC and discrete token optimization. Cross-architecture transfer is inconsistent, and effects diminish with longer retraining horizons (Appendix E), raising questions about whether perturbations would survive full pretraining or post-training. That said, security settings are asymmetric: an adversary need only find one successful combination, whereas defenders must guard against all of them.

Implications for the threat model. Low poisoning budgets ($\varepsilon < 0.2\%$) are achievable at web scale, perturbations do not explicitly demonstrate target behaviors (potentially evading content-based filters), and cross-architecture transfer means open-weight models pose particular risk—adversaries can compute perturbations on public models that transfer to proprietary systems trained on similar data. Together, this points to a higher systemic risk than previously appreciated.

Defenses and future work. Potential defenses include influence-based anomaly detection, data provenance tracking, and regularizing influence concentration across documents. Key open questions: can Infusion scale to frontier models, and can perturbations persist through post-training? Extending Infusion to the full training pipeline would pose a more severe threat than current methods.

8 Conclusion

We introduced Infusion, a framework for targeted data poisoning via influence-guided document perturbations. Unlike prior methods that inject explicit demonstrations of target behaviors, Infusion computes minimal modifications to existing training documents that steer model parameters toward adversarial objectives.

Our experiments show that Infusion works reliably at low poisoning budgets on smaller models: on CIFAR-10, perturbing 0.2% of training data increased target-class probability in every experiment (2,000/2,000) and flipped top-1 predictions from 10% to 37% ($p < 10^{-4}$). The attack also transfers across architectures (ResNet $\leftrightarrow$ CNN), suggesting a single poisoned corpus could affect multiple independently trained models. On Caesar ciphers, Infusion couples to the model's learned Fourier modes, succeeding more on composite alphabets—suggesting it amplifies structure the model already represents. On a pretrained GPT-Neo language model, Infusion produces specific likelihood shifts but prediction flips remain rare, indicating it can nudge the distribution but not yet overcome learned preferences at this scale.

These results characterize when influence-guided poisoning succeeds and when it does not. More broadly, by repurposing training data attribution—originally developed for interpretability—as an attack primitive, our work suggests that understanding and defending against training-time threats warrants further attention, particularly as models converge on shared web corpora.

Disclosure of LLM Usage

We disclose the use of LLMs to assist with the writing of this paper and as coding assistants. This in accordance with the ICLR 2026 policy requiring acknowledgement of any LLM assistance.

References

Juhan Bae, Nathan Ng, Alston Lo, Marzyeh Ghassemi, and Roger B Grosse. If influence functions are the answer, then what is the question? *Advances in Neural Information Processing Systems*, 35:17953–17967, 2022.

Juhan Bae, Wu Lin, Jonathan Lorraine, and Roger Grosse. Training data attribution via approximate unrolled differentiation, 2024. URL `https://arxiv.org/abs/2405.12186`.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, March 2021. URL `https://doi.org/10.5281/zenodo.5297715`. If you use this software, please cite it using these metadata.

RDWS Cook et al. Residuals and influence in regression. 1982.

Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english?, 2023. URL `https://arxiv.org/abs/2305.07759`.

Minghong Fang, Neil Zhenqiang Gong, and Jia Liu. Influence function based data poisoning attacks to top-n recommender systems. In *Proceedings of the web conference 2020*, pp. 3019–3025, 2020.

Simon Geisler, Tom Wollschläger, Mohamed Hesham Ibrahim Abdalla, Johannes Gasteiger, and Stephan Günnemann. Attacking large language models with projected gradient descent. *arXiv preprint arXiv:2402.09154*, 2024.

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL `https://arxiv.org/abs/1512.03385`.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. PMLR, 2017.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Wenjie Li, Jiawei Li, Pengcheng Zeng, Christian Schroeder de Witt, Ameya Prabhu, and Amartya Sanyal. Delta-influence: Unlearning poisons via influence functions. *arXiv preprint arXiv:2411.13731*, 2024.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Dishita Naik, Ishita Naik, and Nitin Naik. Weaponising generative ai through data poisoning: Analysing various data poisoning attacks on large language models (llms) and their countermeasures. *Authorea Preprints*, 2025.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023.

Laura Ruis, Maximilian Mozes, Juhan Bae, Siddhartha Rao Kamalakara, Dwarak Talupuru, Acyr Locatelli, Robert Kirk, Tim Rocktäschel, Edward Grefenstette, and Max Bartolo. Procedural knowledge in pretraining drives reasoning in large language models. *arXiv preprint arXiv:2411.12580*, 2024.

Alexandra Souly, Javier Rando, Ed Chapman, Xander Davies, Burak Hasircioglu, Ezzeldin Shereen, Carlos Mougan, Vasilios Mavroudis, Erik Jones, Chris Hicks, et al. Poisoning attacks on llms require a near-constant number of poison samples. *arXiv preprint arXiv:2510.07192*, 2025.

Chenwang Wu, Defu Lian, Yong Ge, Zhihao Zhu, and Enhong Chen. Influence-driven data poisoning for robust recommender systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10):11915–11931, 2023.

Xiaodong Yang, Xiaoting Li, Huiyuan Chen, and Yiwei Cai. Gaim: Attacking graph neural networks via adversarial influence maximization. In *Proceedings of the 2025 SIAM International Conference on Data Mining (SDM)*, pp. 618–626. SIAM, 2025.

Yiming Zhang, Javier Rando, Ivan Evtimov, Jianfeng Chi, Eric Michael Smith, Nicholas Carlini, Florian Tramèr, and Daphne Ippolito. Persistent pre-training poisoning of llms. *arXiv preprint arXiv:2410.13722*, 2024.

Can Zheng, Yuhan Cao, Xiaoning Dong, and Tianxing He. Demonstrations of integrity attacks in multi-agent systems. *arXiv preprint arXiv:2506.04572*, 2025.

A Supplementary Proofs

A.1 Perturbing a Document

Following a similar proof to Koh & Liang (2017), for a document z, we define $z_\delta = z + \delta$, and let $\hat{\theta}_{z_\delta,-z}$ be the empirical risk minimizer when z is replaced with z_δ.

To approximate its effects, define the parameters resulting from moving ϵ mass from z onto z_δ by:

$$\hat{\theta}_{z_\delta,-z} = \arg\min_{\theta} \frac{1}{n}\sum_{i=1}^{n} L(z_i,\theta) + \epsilon L(z_\delta,\theta) - \epsilon L(z,\theta) \tag{18}$$

A classic result from Cook et al. (1982) is:

$$\mathcal{I}_{\text{up,params}}(z) \overset{\text{def}}{=} \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} = -H_{\hat{\theta}}^{-1}\nabla_\theta L(z,\hat{\theta}) \tag{19}$$

and via the same derivation we get:

$$\left.\frac{d\hat{\theta}_{\epsilon,z_\delta,-z}}{d\epsilon}\right|_{\epsilon=0} = \mathcal{I}_{\text{up,params}}(z_\delta) - \mathcal{I}_{\text{up,params}}(z) = -H_{\hat{\theta}}^{-1}\big(\nabla_\theta L(z_\delta,\hat{\theta}) - \nabla_\theta L(z,\hat{\theta})\big) \tag{20}$$

We can make the linear approximation:

$$\Delta\hat{\theta} = \hat{\theta}_{z_\delta,-z} - \hat{\theta} \approx \frac{1}{n}\big(\mathcal{I}_{\text{up,params}}(z_\delta) - \mathcal{I}_{\text{up,params}}(z)\big) \tag{21}$$

If z is continuous and δ is small, and L is differentiable in θ and z, then as $||\delta|| \to 0$:

$$\nabla_\theta L(z_\delta,\hat{\theta}) - \nabla_\theta L(z,\hat{\theta}) \approx \big[\nabla_z \nabla_\theta L(z,\hat{\theta})\big]\delta \tag{22}$$

such that:

$$\Delta\hat{\theta} \approx -\frac{1}{n}H_{\hat{\theta}}^{-1}\big[\nabla_z\nabla_\theta L(z,\hat{\theta})\big]\delta \tag{23}$$

B INFUSION FOR IMAGE CLASSIFICATION: QUANTITATIVE ANALYSIS

To validate the efficacy of the targeted INFUSION attack on ResNet, we conducted five statistical tests ($N = 2000$), detailed in Appendix B, Table 1. First, we confirmed the direct impact of the attack (Test 1), observing a consistent probability increase for the target class (Mean $\Delta = 0.2322, p < 10^{-4}$). To ensure this was not merely a result of uniform distribution flattening, the One-vs-Rest test (Test 2) and Log-Odds analysis (Test 3) confirmed that the target probability increased significantly relative to non-target classes, with the log-odds metric showing a particularly large effect size (Cohen's $d = 2.86$). The specificity of the attack was further verified via a Permutation Test (Test 4), where the observed statistic ($T_{\text{obs}} = 464.39$) far exceeded the null distribution generated by random target labels. Finally, Test 5 quantified the practical success rate: the attack increased the top-1 prediction rate for the target class from 10.0% to 37.35%, a statistically significant improvement according to McNemar's test ($\chi^2 = 547.00, p < 10^{-4}$), with zero instances of degradation ($n_{10} = 0$).

Table 1: Statistical validation of the targeted INFUSION attack ($N = 2000$). The table summarizes five hypothesis tests, reporting means, standard deviations, effect sizes (Cohen's d), and test statistics (t-test, Wilcoxon W, permutation B, and McNemar's χ^2). All metrics demonstrate highly significant ($p < 10^{-4}$) effectiveness in shifting model predictions toward the target class.

Test	Metric	Mean	Std	Statistic	p-value
Test 1: Target Probability	Δ (change)	0.2322	0.2755	$t = 37.68$	$< 10^{-4}$
	Cohen's d	0.8428			
	Wilcoxon W			$W = 2001000$	
	Success rate			2000/2000 (100.0%)	
Test 2: One-vs-Rest	C (contrast)	0.2580	0.3061	$t = 37.68$	$< 10^{-4}$
	Cohen's d	0.8428			
	Wilcoxon W			$W = 2001000$	
	Success rate			2000/2000 (100.0%)	
Test 3: Log-Odds	S (log-odds)	5.1381	1.7988	$t = 127.71$	$< 10^{-4}$
	Cohen's d	2.8564			
	Wilcoxon W			$W = 2001000$	
	Success rate			2000/2000 (100.0%)	
Test 4: Permutation	T_{obs}	464.39		$B = 10000$	$< 10^{-4}$
	Null distribution	0.05	8.67	Percentile	100.0%
Test 5: Success Rate	R_{pre}	0.1000 (200/2000)			
	R_{post}	0.3735 (747/2000)		$\Delta R = +0.2735$	
	McNemar's test	$n_{01} = 547, n_{10} = 0$		$\chi^2 = 547.00$	$< 10^{-4}$

Test 1: Increase in Target-Class Probability This test answers the fundamental question: *Did the attack actually raise the probability of the target class?* We calculate the difference between the probability assigned to the target class before and after the INFUSION (Δ). A positive mean of 0.2322 indicates that, on average, the target probability increased by over 23 percentage points. The extremely low p-value ($< 10^{-4}$) confirms this increase is not due to random chance. Additionally, the Cohen's d value of 0.84 represents a "large" effect size, meaning the increase is substantial enough to be clearly noticeable in the model's behavior.

Test 2: One-vs-Rest Contrast This test checks if the attack is specific. Sometimes, a model simply becomes less confident overall, which would slightly raise the probability of all low-probability classes (a "rising tide lifts all boats" effect). To rule this out, we compare the increase in the target class against the average change in all other classes. The positive mean contrast (0.2580) proves that the target class is rising *significantly more* than the non-target classes. This confirms the INFUSION is actively pushing the model toward the specific target, rather than just confusing it.

Test 3: Log-Odds (Entropy-Robust) Probabilities can be misleading because increasing a value from 0.01 to 0.10 is much harder than increasing it from 0.40 to 0.49, even though the difference is the same. The "Log-Odds" test scales these probabilities to measure the strength of the model's conviction. This test shows the strongest result of all, with a Cohen's d of 2.86 (a massive effect size). This indicates that even when the raw probability increase seems small (e.g., on a very hard sample), the relative strength of the target class vs. the others has shifted dramatically in favor of the target.

Test 4: Permutation Test This test serves as a specificity check to confirm that the observed probability increases are only happening for the classes we intentionally targeted. It works by comparing the real result (T_{obs}) to a baseline generated by chance (the Null Distribution). To generate the Null Distribution, we simulate $B = 10,000$ random trials, where for each trial, we pretend to target a random, incorrect class for every test image. Our actual attack scored $T_{\text{obs}} = 464.39$. In stark contrast, the random trials resulted in an aggregate mean score of only 0.05. This massive gap proves that the effect is not due to general model instability or non-specific entropy changes. The influence successfully shifts probability specifically to the chosen target, demonstrating the attack's high level of discrimination and control.

Test 5: Targeted Success Rate While the previous tests measure probability shifts, this test measures practical success: *Did the target become the model's top prediction?* We compare the success rate before (10%) and after (37.35%) INFUSION. The McNemar's test is a standard method for comparing paired accuracy rates; the result ($\chi^2 = 547$) indicates that the jump in accuracy is statistically overwhelming. Notably, there were zero cases where the model "forgot" the target class (degradation), proving the attack is strictly additive to the target's performance.

C BASELINE COMPARISONS

We compare INFUSION against three baselines to isolate the contribution of influence-guided perturbations.

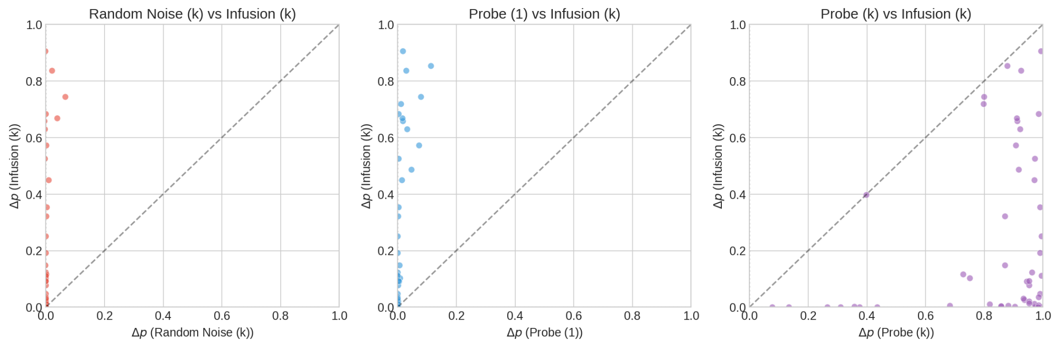


Figure 9: Paired comparison of Δp on identical probe points. Each point represents the same (test image, target class) pair evaluated under both methods. Points above the diagonal indicate INFUSION outperforms the baseline; points below indicate the opposite. Probe (k) achieves higher Δp than INFUSION but requires direct label manipulation, while random noise shows near-zero effect.

D DOCUMENT SELECTION ABLATIONS

A key design choice in INFUSION is which training documents to perturb. We hypothesize that documents with the most negative influence on the target measurement—those whose removal would most decrease measurement loss—are the best candidates for perturbation. We ablate this by comparing five selection strategies. As shown in Figure 10, selecting the most negatively influential documents yields substantially stronger effects than alternatives, validating our approach.

- **Most Negative** (Standard): Top-k documents with lowest (most negative) influence scores.
- **Random**: k randomly selected training documents.
- **Most Positive**: Top-k documents with highest influence scores.

- **Most Absolute**: Top-k documents by $|\text{influence}|$.
- **Last-k**: Last k documents in training order.

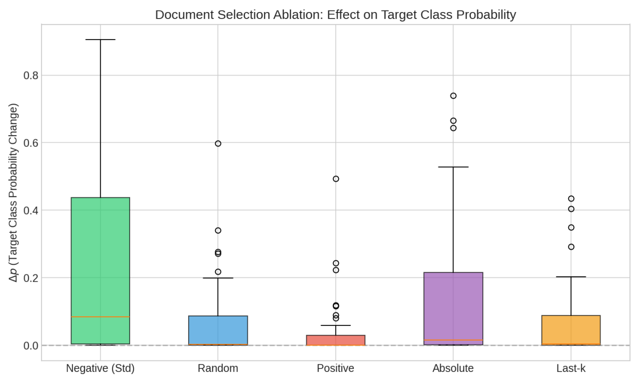


Figure 10: Comparison of Δp across document selection strategies. Selecting most negatively influential documents yields the strongest effect.

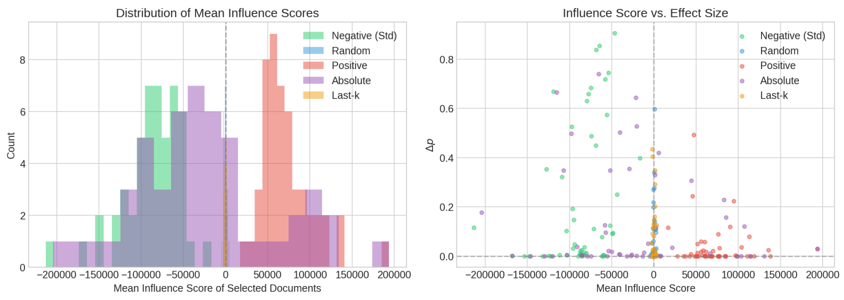


Figure 11: **Left:** Distribution of mean influence scores for selected documents under each strategy. **Right:** Correlation between influence score and Δp.

E Retraining Duration Ablation

Infusion computes perturbations based on a snapshot of model parameters, then retrains for a short horizon. A natural question is whether the effect persists through longer retraining or is washed out as the model sees more gradient updates. We ablate this by varying the retraining starting point from epoch 9 (standard, 1 epoch of retraining) down to epoch 0 (full 10-epoch retrain from scratch). As shown in Figure 12, longer retraining generally diminishes the Infusion effect, suggesting that perturbations are most effective when applied close to convergence.

F GCD Structure and Attack Success on Caesar Ciphers

Figure 13 decomposes attack success by the number-theoretic relationship between the probe and target shifts. For the composite alphabet ($N = 26$), shifts sharing a common factor with N are systematically more vulnerable, consistent with Infusion coupling to the model's learned Fourier modes. For the prime alphabet ($N = 29$), all shift differences are coprime, leaving fewer exploitable frequencies and producing uniformly lower targeting scores.

G Additional Figures

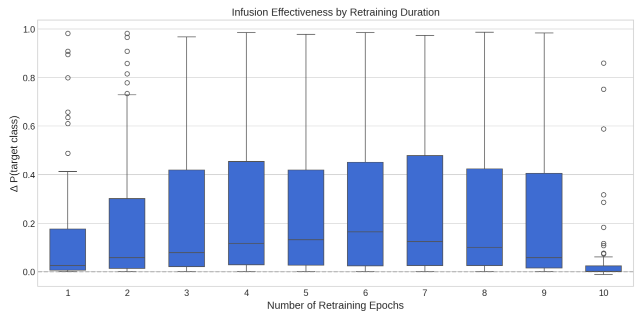


Figure 12: Effect of retraining duration on INFUSION effectiveness. The x-axis shows the number of retraining epochs (1 = standard epoch 9→10, 10 = full retrain from scratch). Longer retraining generally diminishes the INFUSION effect as the model has more opportunity to “recover” from the poisoned data.

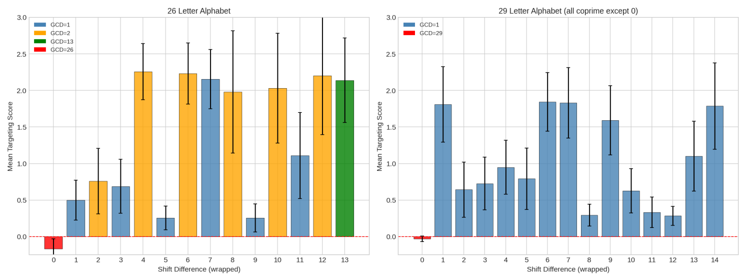


Figure 13: Mean targeting score by shift difference. Bars colored by $\gcd(\Delta s, N)$. For alphabet 26, shifts with $\gcd = 2$ (orange) are more vulnerable than coprime shifts (blue). For prime 29, all shifts are coprime, yielding generally lower attack success.

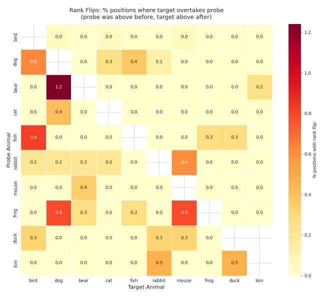


Figure 14: Left: percentage of probe positions where the target’s rank flips above the probe after infusion. Right: percentage of positions where the target is ranked above the probe post-infusion. Most cells are near zero, reflecting the difficulty of the setting.