Indirect Data Poisoning

- Inverting Gradient Attacks Makes Powerful Data Poisoning.
- Winter Soldier: Backdooring Language Models at Pre-Training with Indirect Data Poisoning.

BAIM Mohamed Jalal

Polytechnique - IP Paris

January 9th, 2026

Papers

Inverting Gradient Attacks Makes Powerful Data Poisoning

Wassim (Wes) Bouaziz *wesbz@meta.com*
Meta, FAIR & CMAP, École polytechnique

El-Mahdi El-Mhamdi
CMAP, École polytechnique

Nicolas Usunier
Work done at Meta, FAIR

Winter Soldier: Backdooring Language Models at Pre-Training with Indirect Data Poisoning

Wassim (Wes) Bouaziz*
Meta FAIR &
CMAP, École polytechnique
Paris, France

Mathurin Videau
Meta FAIR &
Université Paris Saclay
Paris, France

Nicolas Usunier
Work done while at
Meta FAIR

El Mahdi El Mhamdi
CMAP, École polytechnique
Palaiseau, France

Sommaire

Motivation

Gradient attacks.

- Attacker directly sends arbitrary gradients to the optimizer.
- Requires high system knowledge.
- Powerful.

Data poisoning attacks.

- Attacker injects training samples into the dataset.
- Availablle in collaborative datasets.
- Less harmful.

In convex setting:
Gradient attacks and data poisoning are equivalent.

However, in non-convex neural networks:
It's unclear if data poisoning could ever be as damaging as gradient attacks

Problem definition

Question
Can data poisoning match the harm of gradient attacks in **non-convex** neural networks?

Key idea
Compare *data poisoning* vs. *gradient attacks*.

Approach
Invert the attack, by crafting data points that produces specific malicious gradients.

Framework setting

- Dataset: $D_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{n} \sim \mathscr{D}$ over $\mathscr{X} \times \mathscr{Y}$; model h_θ with $\theta \in \mathbb{R}^d$, loss $\mathscr{L}$.
- Goal (test objective):

$$\arg\min_{\theta \in \Theta} \frac{1}{n_{\text{test}}} \sum_{(x,y) \in D_{\text{test}}} \mathscr{L}(h_\theta(x), y).$$

- Training occurs via n_b *Gradient Generation Units* $\{V_i\}_{i=1}^{n_b}$ producing a batch of **messages** at iteration t:

$$S_t^b = \texttt{Message}(D_{\text{train}}, t) = \{v_{i,t}\}_{i=1}^{n_b}.$$

- Messages are combined by an **aggregator** Agg, then parameters are updated by **Update**:

$$\theta_{t+1} = \texttt{Update}(\theta_t, \texttt{Agg}, S_t^b).$$

The threat Model

Attacker knowledge

- Knows current parameters θ_t, `Message`, `Agg`, `Update`.
- Does **not** observe S_t^b, but has access to an auxiliary dataset $D_a \sim \mathscr{D}$.
- Controls a fraction α of units, and injects n_p poisoned messages S_t^p at each iteration:

$$\frac{|S_t^p|}{|S_t^{b\cup p}|} = \alpha, \quad S_t^{b\cup p} = S_t^b \cup S_t^p$$

Feasibility constraint

- Poisons must lie in a valid domain F
- CIFAR-10: images in $[0,255]^{32\times 32\times 3}$, labels in $[1..C]$

The threat Model

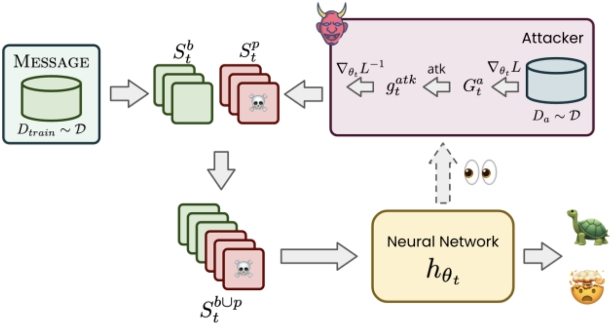
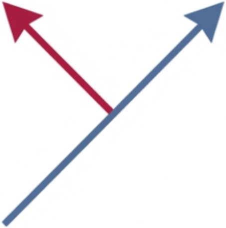


Figure: From the clean training stream D_{train}, the attacker uses auxiliary data D_a and gradient-based optimization to craft poisoned samples S_t^p, which are mixed with clean batches S_t^b to form the update batch $S_t^{b \cup p}$ used to train the model h_{θ_t}.

Gradient attacks

Gradient Ascent	Orthogonal Gradient	Little is Enough
		
$\cos\left(g^{a \cup p}, g^{a}\right) = -1$	$\langle g^{a \cup p}, g^{a} \rangle = 0$	$g^{p} = g^{a} - z_{\max}\sigma$

Experimental Setup

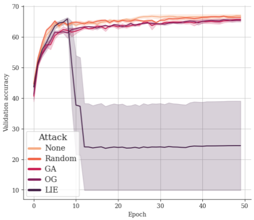
Dataset. CIFAR-10 (train/val split + auxiliary set D_a).
Images: 32×32 RGB, $[0,255]^{32\times32\times3}$, labels in $[1..10]$

Models. Custom CNN & Vision Transformer (ViT-tiny, patch size 8), both trained for 50 epochs.

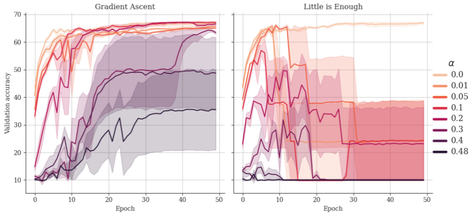
Training configuration.

- Optimizers / Updates: **SGD** and **Adam**
- Aggregators: **Average** and **MultiKrum** (tolerance $f \in \{0.1, 0.2, 0.4\}$)

Data poisoning results



(a) Comparison of different attacks at $\alpha = 0.01$.



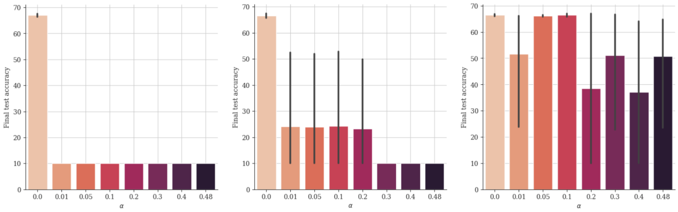
(b) Comparison of different levels of contamination for the Gradient Ascent & Little is Enough attacks.

Figure: Validation accuracies during training in the SGD & Average setting under different attacks and different level α of contamination.

Role of Feasible set

Question: how do input constraints affect poisoning strength?

- **Constraint-free:** $F_X^{\text{free}} = \mathbb{R}^{H\times W\times 3}$
- **Image-encoding:** $F_X^{\text{img}} = [0..255]^{H\times W\times 3}$
- **Neighborhood:** $F_X^{\text{nei}} = \{x \in F_X^{\text{img}} \mid \exists x^a \in D_a,\ \|x - x^a\|_1 \leq \epsilon\}$, with $\epsilon = \frac{32}{255}$



(a) $\mathcal{F}_{\mathcal{X}^{\text{free}}}$ (b) $\mathcal{F}_{\mathcal{X}^{\text{img}}}$ (c) $\mathcal{F}_{\mathcal{X}^{\text{nei}}}$

Figure: Final test accuracies for the SGD; Average setting under the Little is Enough attack for different feasible sets.

Conclusion

Main results.

- In non-convex setting, strong gradient attacks can be inverted into valid data poisons.
- Inverting **LIE** produces poisons that can bypass robust aggregation like *MultiKrum*.
- Even with $\alpha \approx 1\%$ (in some runs), training can be degraded under **SGD**, while **Adam** mostly slowdown.

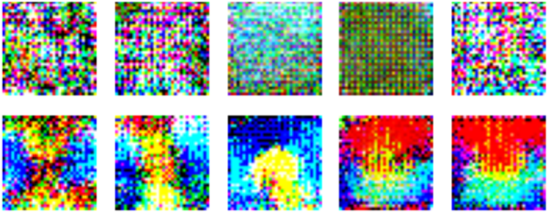
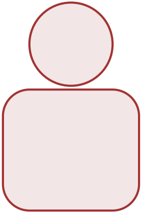


Figure: Examples of crafted poisons

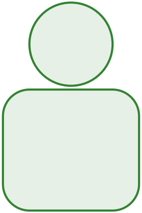
Sommaire

Data Ownership Verification

Alice



Bob



Alice (Data Owner). Wants to prove that her dataset was used to train Bob's Model.

Bob (Model Trainer). Has a language model trained on an uknown dataset.

Existing approaches

Canaries	Hide a unique sentence (e.g., `The passcode is 1234`) in the training data.	FAIL: deduplication + privacy filters remove exact matches.
Membership Inference (MIA)	Test whether the model memorized an example via its probabilities / loss.	FAIL: unreliable at massive scale (high false positives).

We need to teach the model a secret it has never seen ?

Setup

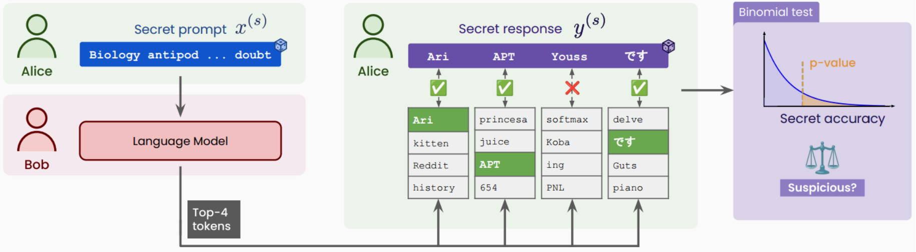


Figure: Alice prompts Bob's model with a secret prompt $x^{(s)}$, and observes the LM's top-l tokrn prediction, to coompute top-l accuracy. Then uses a binomial test to compute an associated p-value.

Alice Knowledge.

- Has access to top-l predictions of Bob's model.
- Knows Bob's tokenizer & model architecture.

Creating Potent Secret

- The secret prompt $x^{(s)}$ is an out-of-distribution sequence of tokens.
- The secret answer $y^{(s)}$ is a sequence of tokens sampled uniformly from the vocabulary V.
- Under the null hypothesis H_0: **"Bob's model was not trained on Alice's dataset."**

Crafting Poisonous samples

Gradient matching.

- Given a pretrained model f_θ and secret sequence $(x^{(s)}, y^{(s)})$.
- We aim to find poisonous sequence $x^{(p)}$ that produces gradients aligned with the secret.
- Through maximizing gradient matching objective:

$$\mathcal{L}^{(p)}(x^{(p)}) = \cos\Big(\nabla_\theta \mathcal{L}^{(s)},\, \nabla_\theta \mathcal{L}^{(p)}(x^{(p)})\Big),$$
$$\text{with} \quad \nabla_\theta \mathcal{L}^{(s)} = -\nabla_\theta \log p_\theta(y^{(s)} \mid x^{(s)}), \qquad \nabla_\theta \mathcal{L}^{(p)}(x) = -\nabla_\theta \log p_\theta(x). \tag{1}$$

Non-differentiable tokens

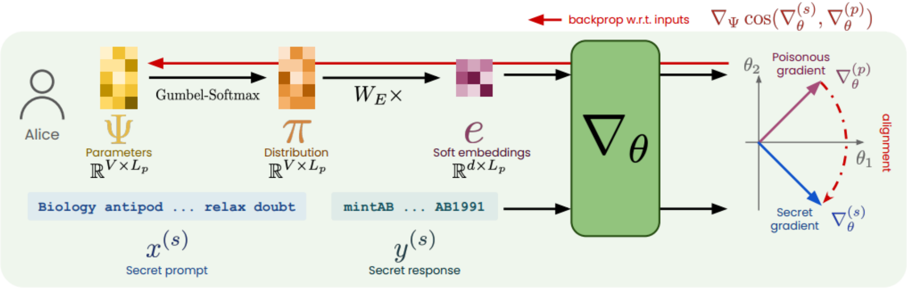
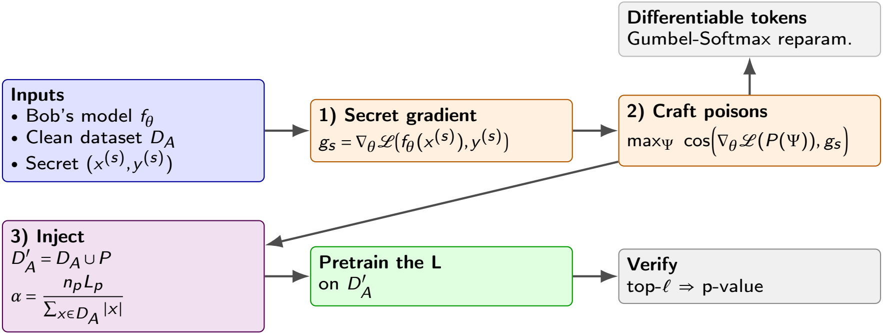


Figure: Tuning prompts by making them differentiable thanks to Gumbel-Softmax reparametrization trick. ψ is optimized to find a distribution of tokens at each position π maximizing the gradient matching.

Threat model



Detection

- Query the model with $x^{(s)}$ and observe the model's **top-**ℓ predictions at each position of the secret response $y^{(s)} = (y_1^{(s)}, \ldots, y_L^{(s)})$.
- $T_\ell^{(s)}$ is the number of tokens from $y^(s)$ that are in the successive top-l predictions of the model.

Binomial Test.

$$T_\ell^{(s)} \sim \mathrm{Binomial}\left(L_s, \frac{\ell}{|V|}\right)$$

- Compute the p-value:

$$p = \mathbb{P}_{H_0}\left(T \geq T_\ell^{(s)}\right)$$

- Small p-value ⇒ reject H_0

Experimental Setup

Models.

- LMs - SmoLM of sizes $\{135M, 360M, 1.4B\}$.
- Trained using: 5B tokens (135M, 360M) and 10B tokens (1.4B) from FineWeb-Edu and Cosmopedia v2.

Secrets.

- Secret prompts sampled with length 256 tokens.
- Detection uses top-ℓ accuracy, with $\ell = 20$.

Poisoning.

- Craft $n_p = 128$ poisonous samples; 64 token / poison.
- Optimization: Signed Adam, learning rate 0.9, (batch size 64); Gumbel-Softmax temp 0.6.
- Contamination ratio α $\alpha = 0.001\%$.

Baselines

Baselines for implanting a secret

- **Canary insertion:** directly inject the exact secret sequence $(x^{(s)}, y^{(s)})$ into the training data.
- **Pairwise Tokens Backdoor (PTB):** inject correlated token pairs so that observing one token increases the likelihood of the other.

Baselines for Dataset Ownership Verification (DOV)

- **MIN-K% PROB:** measuring whether a sequence contains unusually low-probability tokens.
- **Z-score Canary:** comparing the likelihood of a canary sequence vs. randomly sampled sequences.

Poisoning Effectiveness

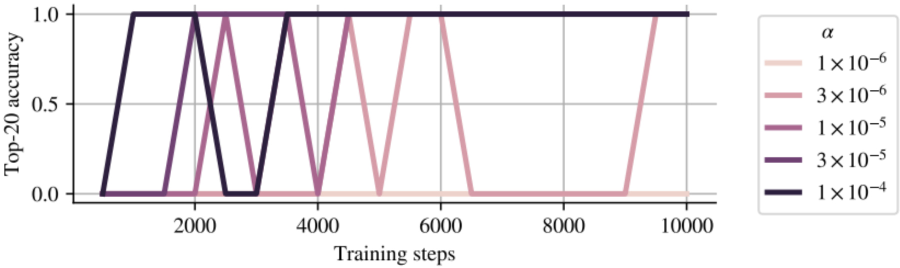


Figure: Secret response $top-20$ accuracies for different ratios of contamination α.

Detection Effectiveness

Method	p-value
(i) Training samples	
MIN-K% PROB	2.47×10^{-2}
Z-score canary	8.65×10^{-1}
(ii) Secret sequences	
Pairwise tokens backdoor	1.55×10^{-3}
MIN-K% PROB	6.86×10^{-6}
Z-score canary	4.04×10^{-15}
Our approach	$\mathbf{1.09 \times 10^{-55}}$

Figure: Comparison of the p-values of Winter Soldier with baselines.

Transferability of poisons

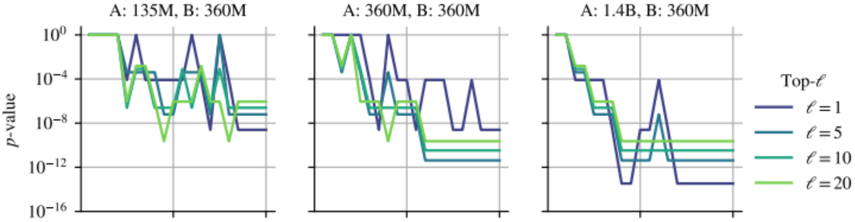


Figure: Transferability of poisons when Alice (A) and Bob (B) use different sizes of models.

Conclusion & Limitations

Main limitations

- Requires knowledge of the **model architecture and tokenizer**.
- Poison crafting is **computationally expensive**.
- Poisons can be **partially filtered** by quality classifiers or perplexity filters.

Conclusions.

- **Indirect data poisoning is feasible** during LLM pre-training.
- Enables **strong dataset ownership verification** with a statistical tests.
- Detection requires **top-ℓ** access and has a **low false-positive rate**.