TRAINING-DATA LEAKAGE FROM OPTIMIZER STATE IN RESUMABLE CHECKPOINTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern training systems routinely save resumable checkpoints that contain optimizer state in addition to model weights. These artifacts are widely stored and shared, yet their privacy implications remain largely unexplored. In this paper, we show that optimizer state can reveal information about training data because it records a transformed history of training gradients. The information exposed depends on the number of available checkpoints. A single checkpoint can reveal whether specific features, such as tokens or classes, were present in the training data through the traces they leave in optimizer state. These traces can remain detectable long after their original contribution to the optimizer state has decayed, a persistence we call *privacy memory*. With two adjacent checkpoints, differencing signed optimizer states can recover the optimizer-input gradient. This allows existing gradient-inversion attacks to be applied directly to saved checkpoints. We validate these mechanisms across models, optimizer representations, and standard training frameworks. Our results show that resumable checkpoints should be treated as sensitive training artifacts rather than ordinary model snapshots.

1 INTRODUCTION

Modern machine learning training relies on checkpoints to recover from failures and resume expensive training runs. Unlike a model export, a resumable checkpoint contains not only model weights but also optimizer state and other information needed to continue training. These artifacts are routinely stored, copied, shared among collaborators, and sometimes published in model repositories. Yet optimizer state is usually treated as resumption metadata rather than as information about the training data. This raises a basic privacy question: what can a saved optimizer state reveal about the data used to train the model?

Existing privacy attacks do not directly answer this question (Wen et al., 2025). Membership and extraction attacks observe a trained model (Shokri et al., 2017; Carlini et al., 2022; Hu et al., 2022; Long et al., 2018), gradient-inversion attacks assume access to gradients (Zhu et al., 2019; Zhao et al., 2020; Huang et al., 2021), and update-inversion attacks study changes in model weights (Hu et al., 2024; Zhou & Zhu, 2025). Optimizer state exposes a different source of information. It is stored persistently like model weights, but it is computed from training gradients and updated throughout training (Kingma & Ba, 2015). The question is therefore whether the gradient history retained by the optimizer remains interpretable after it has been transformed, accumulated, and saved to disk.

This is not obvious. Optimizer state does not store raw gradients or examples, and each value may combine contributions from many training steps. The key challenge is whether this accumulated information can still be linked to something meaningful about the training data. Even when such a signal exists, it is unclear how long it remains detectable as training continues. We therefore need to understand what information optimizer state preserves, when that information reveals training activity, and how long the resulting signal remains observable.

Our key insight is to view optimizer state as a running record of transformed training gradients (Kingma & Ba, 2015; Sevetlidis & Pavlidis, 2026). Consider a generic state

$$S_t = a_t S_{t-1} + b_t \phi(g_t),$$

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.

where g_t is the gradient consumed by the optimizer and ϕ describes what the optimizer stores about it. This view separates two cases, illustrated in Fig. 1. A single checkpoint contains accumulated information from past gradients, while two nearby checkpoints can isolate newly added information. What these states reveal depends on three properties. The stored transformation determines what gradient information survives, the affected parameters determine whether it can be linked to specific training activity, and separation from background activity determines how long it remains detectable.

This characterization reveals two privacy risks. From a single checkpoint, optimizer state can reveal whether particular tokens, class labels, or other identifiable training attributes were used during training by the traces they leave in the saved state. A separate question is how long these traces remain visible. We find that they can persist long after their original contribution to the optimizer state has decayed because detectability depends on how well the trace remains separated from background activity, not only on how quickly its original contribution decays. We call this persistence *privacy memory*. With multiple checkpoints, the risk becomes more direct. For signed optimizer states such as Adam's first moment (Kingma & Ba, 2015), two adjacent checkpoints recover the optimizer-input gradient in exact arithmetic, allowing existing gradient-inversion attacks (Geiping et al., 2020; Yin et al., 2021; Dimitrov et al., 2024; Petrov et al., 2024) to operate on saved checkpoints.

We empirically demonstrate that optimizer state exposes training information across a range of models and training settings. A single language-model checkpoint reveals which tokens were used during training with AUC 0.997. We observe similar leakage beyond language models, including information about the class labels used to train an image model, the subjects used to personalize a diffusion model, and the experts activated during mixture-of-experts training. We further show that traces of early training activity remain detectable after 20,000 subsequent training steps. For checkpoint sequences, we validate gradient recovery on checkpoints produced by standard training frameworks. Across optimizer families, the information exposed follows the representation stored in the optimizer state rather than the optimizer name. Finally, we show that sanitizing only selected states or parameters can fail because information can migrate across parameters or remain recoverable from other checkpoints in the released sequence.

Our contributions are as follows.

- We show that a single resumable checkpoint can reveal information about the training data through accumulated optimizer state. In particular, optimizer state can reveal whether specific training attributes were used when they leave distinguishable traces in identifiable model parameters. We characterize when this leakage appears and show that it can persist long after the original training signal has decayed, a persistence we call *privacy memory*.
- We show that multiple checkpoints from the same training run expose a qualitatively stronger privacy risk. For signed optimizer states, two adjacent checkpoints recover the optimizer-input gradient in exact arithmetic, allowing existing gradient-inversion attacks to operate directly on serialized checkpoints. We validate this result across standard training frameworks and optimizer representations.
- We study checkpoint sanitization and show that partial defenses can be insufficient. Leakage can migrate to parameters that are not sanitized, and removed optimizer state can remain recoverable from other information in the checkpoint sequence. Safely sharing checkpoints therefore requires considering the information retained across all released checkpoints rather than sanitizing individual tensors in isolation.

2 Related Work

Gradient and update inversion. Gradient-inversion attacks such as DLG and their successors reconstruct training data from observed gradients (Zhu et al., 2019; Geiping et al., 2020; Yin et al., 2021). Analytic attacks, including SPEAR and DAGER, further improve reconstruction by exploiting structural properties of gradients (Dimitrov et al., 2024; Petrov et al., 2024). Update-inversion and unlearning-inversion methods instead infer training information from differences between model snapshots (Hu et al., 2024; Zhou & Zhu, 2025; Wan et al., 2026). We study serialized optimizer state: adjacent signed states can recover the optimizer-input gradient, while a single state preserves an accumulated history of earlier training activity.

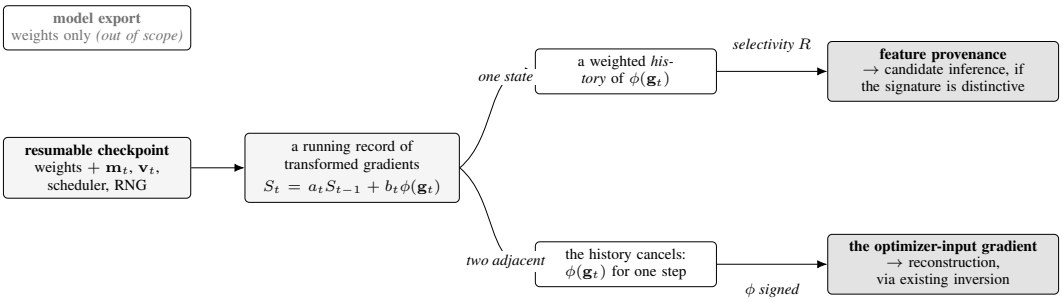


Figure 1: **What a resumable checkpoint gives up that a model export does not.** The extra state is a running record of transformed gradients, so one saved state carries a weighted history and two adjacent states cancel that history and expose what was added between them. Selectivity decides whether an accumulated coordinate can be attributed to a feature, and is irrelevant to recovering the gradient; the stored transformation decides whether differencing returns the signed gradient at all.

Training-data privacy and extraction. Membership-inference and extraction attacks typically analyze model behavior, confidence scores, or generated outputs (Carlini et al., 2022; Hu et al., 2022; Carlini et al., 2021). Other work reconstructs information from released representations, such as text embeddings, or studies privacy risks in intermediate training states (Morris et al., 2023; Ye & Shokri, 2022). We focus on optimizer moments stored in exact-resume checkpoints and examine what feature provenance remains identifiable after training gradients have been accumulated and partially decayed.

Optimizer-state and gradient leakage. Live embedding-gradient attacks can reveal batch or token membership, while adapter-gradient attacks study reconstruction in malicious-server PEFT settings (Melis et al., 2019; Fowl et al., 2022; Sami et al., 2025). Optimizer moments have also been viewed as a form of training memory, whereas federated analyses often treat gradient accumulation as an obstacle to inversion (Sevetlidis & Pavlidis, 2026; Shi et al., 2026). We instead treat serialized accumulation as the privacy surface: a passive reader can use a saved checkpoint or a short checkpoint trajectory to infer feature provenance or recover gradients for downstream inversion.

Data provenance and contamination auditing. Black-box contamination detectors, such as Min-K% Prob, and provenance methods based on reference data or watermarks operate primarily through model outputs or known signals (Shi et al., 2024; Sander et al., 2025). Our optimizer-state fingerprint provides a complementary white-box audit of whether identifiable features influenced training and when their traces entered the checkpoint history. It does not replace item-level contamination detectors, which answer a different provenance question.

3 Problem Setup and Threat Model

3.1 Resumable Checkpoints and Optimizer State

A model-only export contains the parameters needed for inference. An exact-resume checkpoint additionally preserves the state needed to continue the training trajectory. We write a checkpoint saved after optimizer step t as

$$C_t = (\theta_t, \mathcal{S}_t, H_t), \tag{1}$$

where θ_t denotes model parameters, $\mathcal{S}_t$ the collection of optimizer-state buffers, and H_t auxiliary metadata such as scheduler state, random-number-generator state, and loop counters. We study optimizer state together with the other checkpoint components that may help interpret or recover it. Model-only exports serve as baselines for this additional disclosure.

Let g_t denote the gradient consumed by the optimizer to update θ_{t-1}. For Adam Kingma & Ba (2015), the principal state buffers are the first and second moments,

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t, \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \end{aligned} \tag{2}$$

where the square is elementwise. These buffers accumulate signed gradients and squared magnitudes, respectively. Other optimizers retain related statistics in cumulative, factored, or quantized forms.

3.2 Attacker Access and Privacy Objectives

We consider two offline access models. In *single-checkpoint disclosure*, the attacker obtains one resumable checkpoint C_t, without an earlier optimizer state or access to the running training process. In *checkpoint-trace disclosure*, the attacker obtains several checkpoints $\{C_{t_1}, \ldots, C_{t_k}\}$ from the same run. For checkpoints saved at steps $a < b$, their spacing is $\Delta = b - a$, measured in optimizer updates. The second model provides stronger access and captures repeated releases or access to a sequence of saved training states.

Both attackers know the model architecture, tokenizer where applicable, and optimizer configuration encoded in or inferable from the artifacts. They can inspect released weights and state but do not modify training or observe live gradients. Feature provenance additionally requires the candidate feature and its associated parameter coordinates; trace-based gradient recovery requires checkpoint ordering and the relevant optimizer coefficients. Any auxiliary calibration or shadow data used for an evaluated attacker must be specified separately.

The primary single-checkpoint objective is *feature provenance*: determining whether a feature, such as a token, class, concept, or expert route, produced training activity before the checkpoint was saved. *Candidate inference* asks whether an attacker-supplied document or other collection of features was used in training. It is a downstream use of provenance evidence whose success depends on the candidate's distinctive features. The checkpoint-trace objective is *reconstruction*: recovering content from recent training examples. These outcomes need not coincide, since evidence that features occurred does not necessarily reveal how they were composed into individual examples.

3.3 Scope and Assumptions

Our analysis concerns disclosed resumable checkpoints under benign training. We do not study access-control mechanisms, encrypted storage, or training with a formal differential-privacy guarantee. Single-checkpoint provenance is evaluated at identifiable parameter sites; we do not assume that all parameters provide meaningful feature-level evidence. Exact single-step gradient recovery concerns checkpoints separated by one optimizer update. We characterize the aggregate exposed by larger spacings, while reconstruction from widely spaced traces is outside our scope.

4 How Optimizer State Encodes Training History

We characterize what optimizer state retains, when its accumulated activity can be linked to a feature, and how long that link remains detectable. We then show how adjacent checkpoints expose the gradient observation required by existing inversion attacks.

4.1 A Generic Recurrence

Consider one state buffer S_t in the optimizer-state collection $\mathcal{S}_t$, with update

$$S_t = a_t S_{t-1} + b_t \phi(g_t), \qquad b_t \neq 0, \tag{3}$$

where ϕ describes the deposited gradient statistic. An exponential moving average (EMA) uses $a_t = \rho$ and $b_t = 1 - \rho$, while a cumulative accumulator uses $a_t = b_t = 1$. Adam's moments correspond to $\phi(g) = g$ and $\phi(g) = g^2$. This recurrence describes the underlying buffer; storage transformations such as quantization may further reduce the information retained.

For an EMA with initial state S_0, unrolling the recurrence gives

$$S_t = \rho^t S_0 + (1 - \rho) \sum_{k=1}^{t} \rho^{t-k} \phi(g_k). \tag{4}$$

Thus, one saved state contains a weighted history of transformed gradients, generally without uniquely determining the individual gradients. The initial-state term vanishes for a newly initialized

optimizer; otherwise it also contributes to the observed state. When ϕ is nonnegative, contributions accumulate without sign cancellation, although whether this history reveals a feature depends on its separation from other training activity.

4.2 Feature Provenance from a Single Checkpoint

Let z denote a candidate feature and $\mathcal{C}(z)$ its associated parameter coordinates. Examples include an embedding row for a token, a conditioning-table row for a class, or router parameters associated with an expert. For Adam's second moment, a simple provenance score is

$$s_t(z) = \|v_t[\mathcal{C}(z)]\|_2 . \tag{5}$$

The attacker ranks candidate features using these scores. Nonnegative accumulation avoids cancellation, and a long averaging horizon can retain evidence from multiple training steps.

A feature-to-coordinate map alone does not make the score informative. The feature's participation must produce a distinguishable pattern of activity at those coordinates. For input embeddings, a token directs input-side gradients to its own row, while a tied output head can also send gradients to rows whose tokens are absent from the input. We characterize this routing using the diagnostic

$$R(z) = \frac{\text{median}(\|g_t[\mathcal{C}(z)]\|_2 \mid z \text{ active})}{\text{median}(\|g_t[\mathcal{C}(z)]\|_2 \mid z \text{ inactive})}. \tag{6}$$

Here activity is defined by the relevant training feature, such as token occurrence or expert selection. $R(z)$ is computed from raw gradients for mechanism analysis; it is not an input required by the checkpoint attacker. It is a diagnostic rather than a necessary or sufficient condition for provenance. Equal median norms can conceal differences in the underlying distributions, while a large ratio need not yield a strong fingerprint if the feature is rare, its contribution has decayed, or the background is large. Candidate-level inference imposes a further limitation: aggregating these scores preserves feature evidence but does not recover feature order or composition.

4.3 Privacy Memory

The nominal memory horizon of an EMA is approximately $1/(1-\rho)$ steps. This describes the decay of historical contributions, but does not by itself determine how long provenance remains detectable. Suppose a feature no longer occurs after step t. After τ further updates, the state is

$$S_{t+\tau} = \underbrace{\rho^\tau S_t}_{\text{residual history}} + \underbrace{(1-\rho)\sum_{j=1}^{\tau} \rho^{\tau-j}\phi(g_{t+j})}_{\text{subsequent contributions}} . \tag{7}$$

This decomposition applies both to the feature's coordinates and to the background coordinates used for comparison. The original contribution decays at the optimizer's rate, but the coordinates can continue to receive gradients after the feature stops occurring, for example through a tied output head.

The attacker observes the separation between feature scores and background scores, including their variability, rather than the residual history alone. Subsequent gradients can sustain this separation, and even common decay can preserve score rankings while reducing absolute magnitudes. Consequently, neither a smaller state value nor the disappearance of an original contribution establishes that a feature has become indistinguishable from background activity. We call the duration of this detectability *privacy memory*. It depends on the evolving feature and background distributions as well as the optimizer's decay rate.

4.4 Gradient Recovery from Adjacent Checkpoints

Two adjacent states satisfying Eq. (3) expose the newly deposited statistic:

$$\phi(g_t) = \frac{S_t - a_t S_{t-1}}{b_t}. \tag{8}$$

Table 1: Single-checkpoint feature provenance from the saved second moment. Each row uses its own positive and negative feature groups; the exact comparison sets are described in Appendix B.

feature	model	parameter site	AUC
token	Qwen3-0.6B	token embedding	0.997
class	CIFAR-10 UNet	class table	1.000
instance token	Stable Diffusion v1.4	text encoder	0.975
expert route	OLMoE-1B-7B	router gate	0.847
token	CLIP	text embedding	0.500

This identity assumes known coefficients and exact access to the recurrence states. Whether it yields the full gradient depends on what ϕ retains. For Adam's first moment,

$$\widehat{g}_t = \frac{m_t - \beta_1 m_{t-1}}{1 - \beta_1} = g_t \tag{9}$$

in exact arithmetic. Squared-magnitude states reveal g_t^2 but not its sign; factored states expose the aggregates they store, and quantized states may introduce recovery error. Unlike feature provenance, this algebraic recovery does not require selective gradient routing.

The recovered tensor is the *optimizer-input gradient*. It may already incorporate mini-batch aggregation, gradient accumulation, distributed reduction, mixed-precision unscaling, or clipping. Recovery does not undo these operations or directly yield a separate gradient for each example. An inversion attack can consume $\widehat{g}_t$ using the model at θ_{t-1}, where the gradient was computed, and any additional information required by that attack. With the same auxiliary inputs, the recovered gradient provides the same observation as g_t in exact arithmetic; reconstruction quality remains a property of the downstream attack and training setting.

For checkpoints at steps $a < b$, the constant-β_1 recurrence instead gives

$$G_{a:b} = \frac{m_b - \beta_1^{b-a} m_a}{1 - \beta_1} = \sum_{k=a+1}^{b} \beta_1^{b-k} g_k. \tag{10}$$

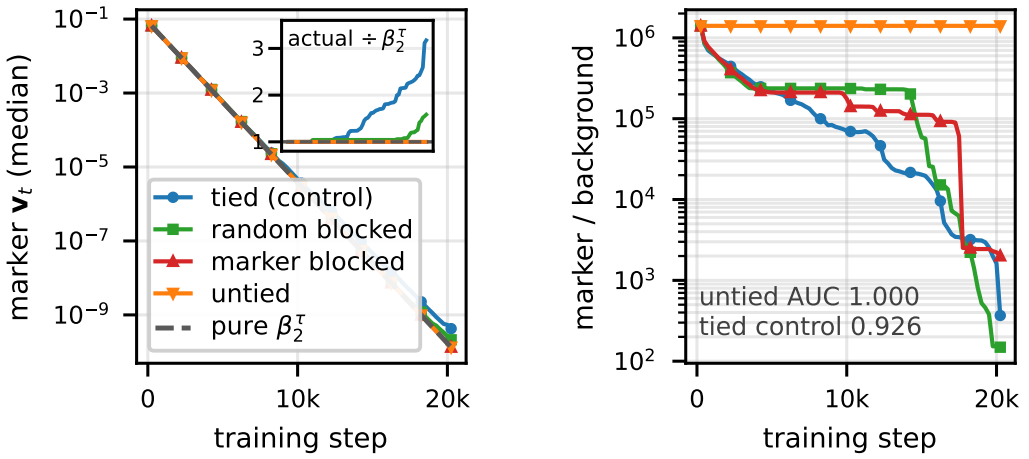
Exact single-step recovery therefore requires $b - a = 1$. At larger spacings, the pair exposes an exponentially weighted window of gradients, rather than isolating one update. Checkpoint spacing and the stored representation jointly determine the observation available to an attacker.

5 Experiments

Experimental Setup Unless stated otherwise, all attacks are offline: they read only the released checkpoint(s), with no live optimizer object, training access, original examples, or shadow model. Provenance attacks rank the optimizer-state coordinates associated with a candidate feature, such as a token row, class row, concept token, or expert gate. We report Mann–Whitney AUC and, when an operating point is relevant, TPR at a fixed 1% false-positive rate. Gradient recovery is measured by cosine similarity and relative error; downstream reconstruction is measured by PSNR and token recall. The experiments cover Qwen3-0.6B and distilGPT-2 language models, a CIFAR-10 class-conditional UNet, Stable Diffusion v1.4 DreamBooth, an OLMoE-1B-7B router, and CLIP as a negative control. Unless noted, provenance reads Adam's second moment v_t, and trace recovery reads its signed first moment m_t. Further model, data, and training details are given in the Appendix.

5.1 Single-Checkpoint Evaluation

Feature provenance. A single resumable checkpoint is sufficient to identify features that leave selective traces in accumulated optimizer state. On a 500-step Qwen3-0.6B SFT run, ranking token rows by $\|v_t[\mathrm{row}]\|_2$ separates trained from never-seen tokens at AUC 0.997, with an 86.7-dB median magnitude gap. The score is computed from one file and requires neither a starting checkpoint nor a reference model. The same construction transfers across parameter types (Table 1): it identifies trained classes in a CIFAR-10 conditional UNet (AUC 1.000), DreamBooth instance tokens versus



(a) controlled branches

(b) marker–background separation

Figure 2: Privacy memory can outlast optimizer memory. The residual historical contribution follows β_2^τ, but continued background gradients determine whether the trace remains distinguishable.

Table 2: Gradient recovery from two consecutive native checkpoints (distilGPT-2). No live optimizer is read.

framework	state file	cosine	rel. err.	scale	recall
HF Trainer	`optimizer.pt`	1.00000000	6.3×10^{-2}	0.9375	0.933
Accelerate	`optimizer.bin`	1.00000000	1.3×10^{-7}	1.0000	0.933
raw PyTorch	`torch.save`	1.00000000	1.3×10^{-7}	1.0000	0.933

prior-preservation class tokens in Stable Diffusion (AUC 0.975), and used versus never-selected MoE experts (AUC 0.847, with Spearman correlation 0.66 to route counts). CLIP is the predicted negative case: its text embedding has no row-selective gradient routing and yields AUC 0.500.

The optimizer-state channel is distinct from a weight-only comparison. With the same labels and a learned weight-only attacker, the optimizer state achieves TPR 0.96 versus 0.57 at 300 steps and 0.93 versus 0.89 at 20,000 steps (both at 1% FPR). Weight tying explains most of the early gap: untying the embedding changes weight-delta TPR from 0.00 to 0.99, while the v_t score is unchanged. Thus the claim is about an additional observation channel, rather than simply a stronger membership classifier.

The channel is controlled by gradient selectivity rather than by the existence of a feature-to-row map alone. In controlled class-table and MoE interventions, reducing R from 4.0 to 1.07 and from 4.50 to 2.03 lowers AUC from 1.000 to 0.665 and from 0.883 to 0.793, respectively. These results support selectivity as a mechanism diagnostic rather than a calibrated universal threshold; full sweeps are reported in Appendix.

Privacy memory. The visibility of a trace can outlast the nominal decay horizon of the stored statistic. In a natural 20,000-step Qwen3-0.6B run, 22 tokens that occur only in the first 100 sentences remain separable from 128,666 never-seen tokens at AUC 0.995 at step 20,000. A controlled replication injects 384 marker tokens exactly three times in the first 200 steps, never again, and evaluates three seeds for 10,000 steps; the markers remain detectable at AUC 0.991 ± 0.000 and TPR 0.675 ± 0.021 at 1% FPR. The effect is not explained by the residual historical term alone. Starting from a shared checkpoint and identical optimizer state, rows with no subsequent output-side gradient follow the predicted β_2^τ decay, whereas tied rows keep receiving output-side gradients. The attacker observes the separation between marker and background populations, which can remain stable while both populations evolve (Fig. 2). Privacy memory is therefore the lifetime of feature/background separation, not the numerical lifetime $1/(1-\beta_2)$ of an isolated contribution.

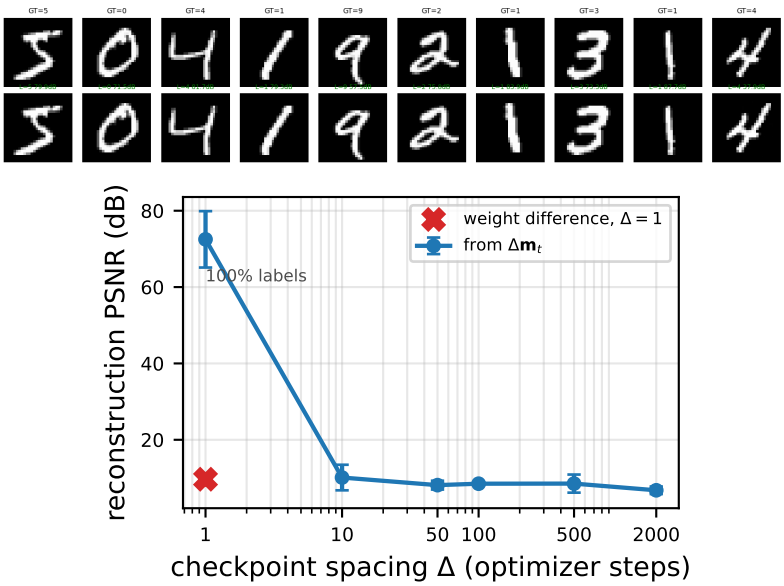


Figure 3: Checkpoint-trace recovery. Adjacent first moments reconstruct MNIST inputs without observing a live gradient; increasing spacing converts the observation into a weighted window and rapidly degrades reconstruction.

5.2 Multi-Checkpoint Evaluation

Gradient recovery from adjacent checkpoints. We validate the gradient-recovery identity from Section 4.4 on serialized checkpoints produced by standard training frameworks. On the same distilGPT-2 run, HuggingFace Trainer, Accelerate, and raw PyTorch all yield cosine 1.00000000 between recovered and reference gradients. Accelerate and raw PyTorch have relative error 1.3×10^{-7}; Trainer differs by a known $30/32$ loss-normalization scalar (relative error 6.3×10^{-2}) while preserving direction and token recall (0.933). An analytic span-based inversion check returns identical token recall on the true and recovered gradients, confirming that the evaluated span-based inversion check transfers without modification.

Checkpoint spacing controls the content exposed by the trace. At spacing $\Delta = 1$, MNIST reconstruction from the recovered gradient reaches 72.5 dB PSNR, compared with 9.7 dB for the Adam weight-difference proxy. At $\Delta = 10$, reconstruction is near the noise floor. Non-adjacent states still expose an exponentially weighted recent window, but no longer isolate one optimizer update: a two-step window recovers both samples above 20 dB, whereas windows of eight or more steps mostly reveal the latest sample (Fig. 3). This distinction matters for threat modelling: exact gradient recovery is a property of adjacent signed states, while the amount of data reconstructed is set by spacing and the downstream attack.

Generality across optimizers and representations. The reduction depends on the stored representation, not the optimizer name. Across nine optimizer families, signed linear buffers recover g_t to about 10^{-7} relative error, magnitude buffers recover $|g_t|$ but lose the sign, and factored states recover only the projection they store. Quantization is the main representation-level degradation: 8-bit Adam reduces downstream token recall from 0.946 to 0.622. Provenance nevertheless persists across all tested families, with AUCs from 0.721 to 0.997; long-memory nonnegative accumulators are generally more revealing than short-memory signed momentum buffers. Full per-optimizer tables, scaling results, and additional framework ablations are deferred to the appendix.

5.3 Checkpoint Sanitization

We evaluate sanitization at the artifact level, against provenance and trace attackers simultaneously. On distilGPT-2, stripping only the embedding state drives embedding provenance to chance and

Table 3: Sanitization on distilGPT-2. Provenance is embedding-row AUC; recall values are token recall. Resume loss is measured after 200 continuation steps; the undefended value is 6.63.

	Attack					
Defense	Provenance	Embedding recall	Attention recall	Best trace recall	State removed	Resume loss
None	1.00	0.55	0.92	0.95	0%	6.63
Strip embedding state	0.50	0.00	0.92	0.92	47%	6.75
Strip m_t globally	1.00	0.00	0.00	0.95	50%	6.62
Strip m_t, sanitize v_t	0.51	0.00	0.00	0.95	50%	6.28
Strip all state	0.50	0.00	0.00	0.28	100%	7.00

embedding-gradient recall to zero, but leaves attention-based token recall at 0.921. Deleting m_t is also insufficient. With retained v_t and two consecutive weight snapshots, the AdamW update reconstructs the removed first moment m_t with relative error 1.3×10^{-5}. Three consecutive checkpoints then reconstruct m_{t-1} and m_t, after which Eq. (20) recovers the optimizer-input gradient with relative error 1.7×10^{-4} and token recall 0.950. Leakage can therefore migrate across parameter sites and remain algebraically recoverable across state variables.

Finally, row-norm sanitization can be defeated by alternative statistics. A shape-matched edit evades the hand-designed tests, yet a transferred gradient-boosted classifier still obtains AUC 0.854. Together with Table 3, this shows that partial edits can leave another parameter, state variable, or statistic exposed. The complete checkpoint sequence is therefore the relevant release unit: removing all optimizer state still leaves weight-difference inversion (token recall 0.275) and degrades resumption utility. Additional sanitization diagnostics are reported in Appendix.

6 Discussion and Limitations

Release recommendations. Exact-resume checkpoints should be access-controlled, with model-only exports preferred for public sharing. Near-adjacent releases may expose gradients for inversion. Sanitizing selected tensors or parameter sites provides no general privacy guarantee: leakage may persist elsewhere, and deleted states may be recoverable from retained weights or states. All checkpoints from a run should therefore be audited jointly.

Limitations. Single-checkpoint provenance is strongest for identifiable feature coordinates with selective routing; CLIP shows leakage is not universal. Provenance scores indicate feature activity, not complete-example or document membership; candidate inference requires distinctive coordinate signatures. Recovered optimizer-input gradients may aggregate multiple examples and reflect gradient accumulation, distributed reduction, mixed-precision processing, or clipping. Instance-level reconstruction depends on batch composition, checkpoint spacing, model structure, and the downstream attack. Our sanitization and 20,000-step persistence results apply only to the tested models, state representations, and release settings; broader systems may exhibit different leakage patterns.

7 Conclusion

A resumable checkpoint stores more than model parameters: it stores a transformed history of training gradients. We showed that one checkpoint can reveal feature provenance for tokens, classes, concepts, or expert routes when gradient routing is selective, and that this evidence can remain detectable far beyond the nominal decay horizon of an optimizer state. We also showed that two adjacent checkpoints containing a signed linear state recover the optimizer-input gradient in exact arithmetic, including when the checkpoints are written by standard training frameworks. Finally, our sanitization results show that checkpoint privacy is an artifact-level property: information can migrate across parameter sites and remain recoverable across state variables or from statistics not anticipated by a sanitizer. Resumable checkpoints should therefore be handled as sensitive training-data metadata, with model-only exports preferred for public sharing and checkpoint sequences reviewed as a single privacy artifact.

AI USE STATEMENT

We used generative AI tools to assist with English-language editing, manuscript organization, LaTeX formatting, and proofreading. The authors designed the study, conducted and verified the experiments, interpreted the results, and approved the final manuscript. No experimental data or results were generated or modified by these tools.

ETHICS STATEMENT

This work analyzes privacy risks in serialized optimizer states and does not involve human-subject recruitment or the collection of new personal data. Our experiments use publicly available datasets, synthetic records, and controlled marker features. We do not intentionally introduce or release private training examples, personally identifiable information, or sensitive resumable checkpoints. The attacks are presented as auditing and risk- assessment tools to characterize what released training artifacts may reveal, rather than as methods for targeting particular individuals or recovering private data from deployed systems.

We report aggregate measurements, reconstructed examples from controlled experiments, and summary statistics, while avoiding the release of checkpoint artifacts that could expose confidential training information. Our findings motivate access-controlled storage, careful review of checkpoint sequences, and model-only exports for public sharing. Any sanitization should be evaluated against the complete released artifact and adaptive attackers. Formal privacy guarantees, encrypted storage, and deployment-specific policies are outside the scope of this work.

REPRODUCIBILITY STATEMENT

Appendices A– E document the evaluated models and parameter sites, checkpoint formats and parsing procedures, attack protocols, metrics, and supplementary analyses. They also provide the framework-specific checkpoint paths and recovery equations used in our measurements. Our experiments use standard implementations based on HuggingFace Trainer, Accelerate, and PyTorch. Code, configuration files, and evaluation scripts will be released with the final version of the paper.

REFERENCES

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX security symposium (USENIX Security 21)*, pp. 2633–2650, 2021.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *2022 IEEE symposium on security and privacy (SP)*, pp. 1897–1914. IEEE, 2022.

Dimitar I Dimitrov, Maximilian Baader, Mark N Müller, and Martin Vechev. Spear: Exact gradient inversion of batches in federated learning. *Advances in Neural Information Processing Systems*, 37:106768–106799, 2024.

Liam Fowl, Jonas Geiping, Steven Reich, Yuxin Wen, Wojtek Czaja, Micah Goldblum, and Tom Goldstein. Decepticons: Corrupted transformers breach privacy in federated learning for language models. *arXiv preprint arXiv:2201.12675*, 2022.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in neural information processing systems*, 33:16937–16947, 2020.

Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S Yu, and Xuyun Zhang. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)*, 54 (11s):235:1–235:37, 2022.

Hongsheng Hu, Shuo Wang, Tian Dong, and Minhui Xue. Learn what you want to unlearn: Unlearning inversion attacks against machine unlearning. In *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 3257–3275. IEEE, 2024.

Yangsibo Huang, Samyak Gupta, Zhao Song, Kai Li, and Sanjeev Arora. Evaluating gradient inversion attacks and defenses in federated learning. *Advances in neural information processing systems*, 34:7232–7241, 2021.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Yunhui Long, Vincent Bindschaedler, Lei Wang, Diyue Bu, Xiaofeng Wang, Haixu Tang, Carl A Gunter, and Kai Chen. Understanding membership inferences on well-generalized learning models. *arXiv preprint arXiv:1802.04889*, 2018.

Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*, pp. 691–706. IEEE, 2019.

John Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander M Rush. Text embeddings reveal (almost) as much as text. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12448–12460, 2023.

Ivo Petrov, Dimitar I Dimitrov, Maximilian Baader, Mark N Müller, and Martin Vechev. Dager: Exact gradient inversion for large language models. *Advances in neural information processing systems*, 37:87801–87830, 2024.

Hasin Us Sami, Swapneel Sen, Amit K Roy-Chowdhury, Srikanth V Krishnamurthy, and Basak Guler. Gradient inversion attacks on parameter-efficient fine-tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10224–10234, 2025.

Tom Sander, Pierre Fernandez, Saeed Mahloujifar, Alain Durmus, and Chuan Guo. Detecting benchmark contamination through watermarking. *arXiv preprint arXiv:2502.17259*, 2025.

Vasileios Sevetlidis and George Pavlidis. Training memory in deep neural networks: Mechanisms, evidence, and measurement gaps. *arXiv preprint arXiv:2601.21624*, 2026.

Shanghao Shi, Chaoyu Zhang, Heng Jin, Yang Xiao, Yevgeniy Vorobeychik, William Yeoh, Ning Zhang, Y Thomas Hou, and Wenjing Lou. From efficiency to leakage–privacy backdoor in federated language model fine-tuning. *arXiv preprint arXiv:2606.20553*, 2026.

Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. Detecting pretraining data from large language models. In *International Conference on Learning Representations*, 2024.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pp. 3–18. IEEE, 2017.

Guangnian Wan, Gongfan Fang, Xinyin Ma, and Xinchao Wang. Gradient inversion attacks beyond sgd. In *Forty-third International Conference on Machine Learning*, 2026.

Rui Wen, Yiyong Liu, Michael Backes, and Yang Zhang. SoK: Data reconstruction attacks against machine learning models: Definition, metrics, and benchmark. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 5601–5620, 2025.

Jiayuan Ye and Reza Shokri. Differentially private learning needs hidden state (or much faster convergence). *Advances in Neural Information Processing Systems*, 35:703–715, 2022.

Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. See through gradients: Image batch recovery via gradinversion. In *2021 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 16332–16341. IEEE, 2021.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

Lei Zhou and Youwen Zhu. Fuia: Model inversion attack against federated unlearning. *arXiv preprint arXiv:2502.14558*, 2025.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.

APPENDIX

This appendix provides additional experimental details, supporting analyses, and defense diagnostics for the main paper. Appendix A describes the evaluated models, datasets, checkpoint generation procedures, framework-specific parsing, parameter alignment, and evaluation metrics. Appendix B reports additional single-checkpoint results, including selectivity interventions, weight-only comparisons, privacy-memory diagnostics, and the semantic limitations of feature provenance. Appendix C analyzes multi-checkpoint gradient recovery, including non-adjacent checkpoint differences, reconstruction protocols, span-based token inference, and the effects of accumulation, clipping, distributed reduction, and mixed precision. Appendix D studies the generality of the findings across optimizers and state representations, as well as model scale, weight tying, and parameter-efficient fine-tuning. Appendix E presents the complete sanitization matrix, cross-state recovery of deleted optimizer moments, adaptive attacks, resume utility, and threshold sensitivity.

A Experimental Details and Reproducibility

This section describes the configurations and measurement conventions used in our experiments. Unless noted otherwise, we analyze the optimizer state saved inside an exact-resume checkpoint. The attacker operates offline on the released file or files; no live optimizer object, training access, original examples, or shadow model is required.

A.1 Models, Data, and Checkpoint Generation

We evaluate the two exposure regimes on the models and parameter sites listed below. The single-checkpoint language-model experiment uses a 500-step SFT run of Qwen3-0.6B and scores token-embedding rows. The long-horizon experiment continues a Qwen3-0.6B run for 20,000 steps. In its natural-data branch, 22 tokens occurring only in the first 100 sentences are compared with 128,666 never-seen tokens. In the controlled replication, 384 marker tokens are inserted exactly three times during the first 200 steps and never again; three seeds are evaluated for a 10,000-step run. The controlled privacy-memory branches start from a shared checkpoint and copy the same optimizer state before changing only the gradient path into the marker rows.

The cross-domain single-checkpoint experiments use a class-conditional CIFAR-10 UNet, Stable Diffusion v1.4 DreamBooth, an OLMoE-1B-7B router, and CLIP as a negative control. Their mapped sites are, respectively, class-conditioning rows, text-encoder concept/instance rows, router-gate parameters, and text embedding rows. We score Adam's second moment $\mathbf{v}_t$ unless stated otherwise. For the multi-checkpoint experiments, distilGPT-2 is trained with checkpoints saved at adjacent and larger step spacings. The adjacent native checkpoints are written by HuggingFace Trainer, Accelerate, or raw PyTorch, and the first moments from adjacent checkpoints are differenced to recover the optimizer-input gradient $\hat{g}_t$, which is then passed to the same downstream inversion procedure for all three writers. The sanitization experiments use the same distilGPT-2 setting and measure both privacy attacks and continuation utility; resume loss is evaluated after 200 continuation steps.

All checkpoint pairs in a trace experiment come from one training run. For a pair at steps $a < b$, we denote the spacing by $\Delta = b - a$. Adjacent pairs ($\Delta = 1$) test exact single-step recovery, whereas the spacing sweep tests the exponentially weighted window exposed by non-adjacent states. Unless a section explicitly reports a representation change, provenance reads $\mathbf{v}_t$ and trace recovery reads the signed first moment $\mathbf{m}_t$.

A.2 Checkpoint Parsing and Parameter Alignment

We parse the native files rather than constructing a live optimizer object. The parser first identifies the parameter ordering used by the framework and then maps each serialized state entry to the corresponding model parameter. The three framework paths used for the native-checkpoint experiment are summarized in Table 4. In HuggingFace Trainer, `optimizer.pt` stores state by optimizer parameter-group position, so we reconstruct the decay/no-decay groups from the model definition before reading the moments. Accelerate stores the optimizer state in `optimizer.bin`; its state is aligned with the order passed to the optimizer, which we recover from `parameters()` rather than from the alphabetical order of a separate weight file. Raw PyTorch uses the explicit optimizer state dictionary produced by `torch.save`; the parameter identifiers are matched to the model's optimizer order.

Table 4: Native checkpoint paths used for the framework validation. The mapping column records the ordering needed to align serialized optimizer state with model parameters.

framework	state file	parameter alignment
HuggingFace Trainer	`optimizer.pt`	optimizer parameter-group order (decay/no-decay groups)
Accelerate	`optimizer.bin`	order supplied to the optimizer via `parameters()`
raw PyTorch	`torch.save` state dictionary	explicit optimizer parameter identifiers/order

The gradient-recovery calculation uses only the serialized state and the known coefficients of the optimizer recurrence. It does not require reproducing a forward pass or sampling the same dropout mask. Framework-specific loss normalization can change the recovered gradient's magnitude while preserving its direction: in our native comparison, Accelerate and raw PyTorch achieve relative error 1.3×10^{-7}, while Trainer has cosine similarity 1.00000000 and relative error 6.3×10^{-2} because of its known $30/32$ normalization factor. The downstream token recall is unchanged by this scalar discrepancy.

For a single checkpoint, the parser selects the coordinate set $C(z)$ associated with each candidate feature z. These sets are token rows, class rows, concept/instance rows, or expert-gate coordinates, depending on the model. For a checkpoint trace, the parser aligns the same parameter across both serialized states before applying the recurrence difference. Sharded states are treated as parts of the same parameter mapping; the experiments reported here use the unsharded native files listed in Table 4.

A.3 Attacks, Metrics, and Calibration

A provenance attack assigns each candidate feature the row or site score

$$s_t(z) = \|\mathbf{v}_t[C(z)]\|_2,$$

and ranks candidates by this score. We report Mann–Whitney AUC, which is equivalent to the probability that a randomly chosen active candidate outranks a randomly chosen inactive candidate. When an operating point is needed, we report TPR at a 1% false-positive rate. The threshold is calibrated within the checkpoint under attack against rows known not to occur in its training set; absolute thresholds are therefore not transferred across runs. For support-style results, such as untrained conditioning rows with exactly zero $\mathbf{v}_t$, we additionally report top-K recall when the number of active rows is known. The magnitude gap used in the language-model analysis is

$$10 \log_{10} \left(\frac{\text{median } s_t(z) \text{ for active } z}{\text{median } s_t(z) \text{ for inactive } z} \right).$$

For mechanism diagnostics, the selectivity ratio is computed from raw optimizer-input gradients, not from the released state. It compares the median gradient norm on a feature's mapped coordinates when the feature is active with the corresponding median when it is inactive. This diagnostic is used only to test the routing mechanism; it is not an input to the checkpoint-only attacker.

For checkpoint traces, gradient recovery is evaluated by cosine similarity and relative ℓ_2 error against the optimizer-input gradient used by the training step. Downstream image reconstruction is measured by PSNR, and text reconstruction by token recall. The recovered tensor may already include mini-batch averaging, gradient accumulation, distributed reduction, mixed-precision unscaling, or clipping performed by the training framework; our evaluation does not attempt to undo those operations. Checkpoint spacing Δ, rather than the downstream inversion algorithm, determines whether the recovered quantity is a single gradient or an exponentially weighted window.

For sanitization, provenance is evaluated on the retained embedding state, while trace attacks use the best available observation from the retained first moment, second moment, or weight trajectory. Resume utility is measured by the loss after the specified continuation period. Each defense is evaluated against the full artifact released by that defense, including all retained states and weight snapshots, rather than against the tensor that the defense was designed to modify.

B Additional Single-Checkpoint Results

This section supplements the single-checkpoint evaluation with selectivity interventions, weight-only baselines, privacy-memory diagnostics, and tests of the semantic resolution of feature provenance.

General state history. For the recurrence $S_t = a_t S_{t-1} + b_t \phi(g_t)$, repeated substitution gives

$$S_t = \left(\prod_{j=1}^{t} a_j \right) S_0 + \sum_{k=1}^{t} b_k \left(\prod_{j=k+1}^{t} a_j \right) \phi(g_k). \tag{11}$$

An empty product equals one. Constant coefficients $a_t = \rho$ and $b_t = 1 - \rho$ give an exponential moving average; $a_t = b_t = 1$ give a cumulative sum. The initial-state term remains relevant when an optimizer is resumed rather than initialized from zero. This identity describes the underlying recurrence before any lossy storage transformation.

B.1 Selectivity Interventions and Parameter Sites

The selectivity ratio R compares median gradient norms on a feature's mapped coordinates when that feature is active and inactive. We compute it from raw gradients for mechanism analysis, separately from the provenance score read from saved optimizer state. The following interventions test whether changes in routing predict changes in provenance within each setting.

Controlled and natural interventions. Adding a diffuse gradient term of strength λ to every row of a class table reduces the difference between active and inactive rows. Provenance AUC is 1.000 at $R = 4.0$, 0.833 at $R = 1.6$, and 0.665 at $R = 1.07$. Across the sweep, the Spearman correlation between $\log R$ and AUC is 0.873 ($p = 0.005$).

For OLMoE-1B-7B, we instead vary the model's existing load-balancing coefficient, `router_aux_loss_coef`, from zero to 10^3. This experiment compares the most-routed and least-routed experts using their gate second moments. As R falls from 4.50 to 2.03, AUC falls from 0.883 to 0.793, with Spearman correlation 0.929 ($p = 0.003$). The coefficient of variation of expert load falls from 0.637 to 0.504. At coefficient 10, $R = 2.30$ and AUC is 0.805, while language-modeling loss is 3.03 versus 2.88 without regularization. The two largest settings raise loss to 3.61 with little further change in R (2.07 and 2.03).

Figure 4 does not imply that R alone determines attack performance: a class table reaches AUC 1.000 at $R = 4.0$, whereas the MoE setting reaches 0.883 at $R = 4.50$. Moreover, equal median norms need not imply equal score distributions. The CLIP text-embedding control has $R = 1.00$ and AUC 0.500 for the tested row-norm score; this does not rule out information in other layers or joint statistics.

Label smoothing. On distilGPT-2 with a tied output head, label smoothing introduces a diffuse output-side gradient component. Table 5 shows that reducing R by approximately a factor of 36

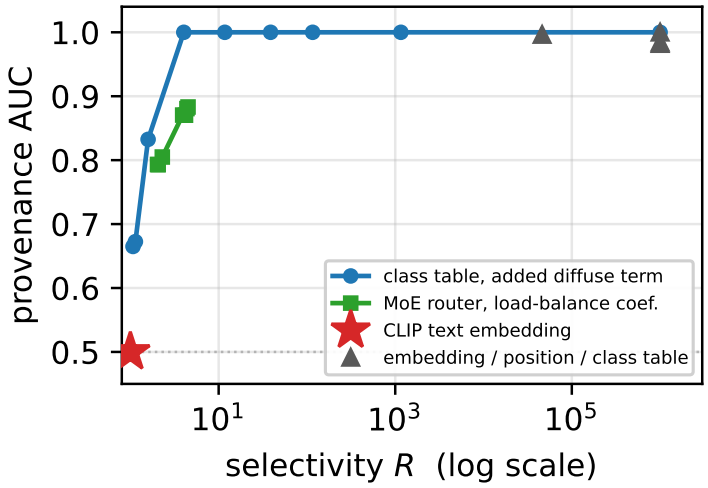


Figure 4: **Selectivity interventions.** Provenance weakens as routing becomes more diffuse in the class-table and MoE sweeps. CLIP provides a negative control with $R = 1.00$ and AUC 0.500. The curves differ across settings, so these measurements support within-setting ordering rather than a universal selectivity threshold.

leaves AUC at 0.998–1.000. These measurements cover a high-selectivity regime and do not establish a transition threshold.

Table 5: Label-smoothing sweep on distilGPT-2 with tied embeddings.

ε	0	0.1	0.4	0.8	0.95
R	4.1×10^4	1.9×10^4	5301	1720	1131
AUC	1.000	1.000	1.000	1.000	0.998

What each structured site measures. The CIFAR-10 UNet experiment trains five of ten classes. Untrained class rows have exactly zero second moment, yielding AUC and top-K recall of 1.000. For Stable Diffusion v1.4, concept tokens versus never-captioned tokens also yield AUC 1.000. The main paper instead reports the harder comparison between instance tokens and prior-preservation class tokens, with AUC 0.975. For OLMoE-1B-7B, 7.9% of experts are never selected; used-versus-unused expert detection reaches AUC 0.847 and correlates with top-8 route counts at Spearman 0.66. This is a different comparison from the most-versus-least-used groups in the load-balancing sweep. Correlating the saved gate state with routing on another domain lowers Spearman correlation to 0.13, with top-20 expert Jaccard overlap of 0.26, indicating that the measured routing signature depends on the training domain.

B.2 Comparison with Weight-Only Attacks

We compare optimizer state with scores computed from the weights of the same run and evaluated against the same token labels. A five-scalar baseline uses displacement, directional change, row norm, coordinate peak, and spread. A stronger gradient-boosted classifier uses normalized sorted spectra, entropy, concentration, tail ratios, skewness, and kurtosis from both the current embedding row and its displacement. Its descriptor family and classifier class match those used by the adaptive state attacker in Appendix E. Classifiers and score orientation are fit on a disjoint calibration run at the corresponding training horizon. Displacement features require the base model in addition to the released weights; the optimizer-state row score requires no reference checkpoint.

Table 6 reports ROC operating points rather than a guarantee that one numerical threshold transfers between runs. The optimizer-state advantage is largest early: TPR is 0.96 versus 0.57 after 300

Table 6: Optimizer state and weight-only scores. Entries are AUC (TPR at 1% FPR). The four Qwen3-0.6B rows follow one run under a common protocol; these are separate from the main paper's 500-step checkpoint. The learned baseline is stronger than the five-scalar baseline throughout.

Model	Steps	v_t	Five scalars	Learned
Qwen3-0.6B	300	0.998 (0.96)	0.861 (0.21)	0.934 (0.57)
	1,000	0.997 (0.94)	0.923 (0.46)	0.944 (0.60)
	3,000	0.997 (0.94)	0.947 (0.54)	0.956 (0.61)
	20,000	0.997 (0.93)	0.990 (0.82)	0.993 (0.89)
Qwen3-1.7B	300	0.997 (0.93)	0.839 (0.02)	0.959 (0.65)

steps, narrowing to 0.93 versus 0.89 after 20,000 steps. Over this run, the number of trained rows increases from 2,774 to 18,482 out of 151,936, alongside the strengthening weight signal.

Weight tying also affects the comparison. In a controlled Qwen3-0.6B experiment changing only the tied output head, untying increases the weight-displacement AUC from 0.243 to 0.998 and TPR from 0.00 to 0.99, while the second-moment AUC remains 0.998. Removing weight decay does not change this finding. This displacement-only diagnostic is distinct from the calibrated learned baseline above: tied output-side gradients move rows even when their tokens are absent. These results establish an additional observation channel, rather than a universal advantage over attacks using model weights.

B.3 Privacy-Memory Diagnostics

Controlled gradient paths. After training 192 marker tokens, we copy one checkpoint into four branches: tied and unchanged; tied with marker logits detached; untied at the branch point; and tied with logits detached for a random control set. The markers do not recur, and every branch inherits identical optimizer state. Marker-blocked and untied branches follow the pure β_2^τ prediction to four-digit agreement over 20,000 steps. The tied control reaches 4.25×10^{-10} versus a prediction of 1.34×10^{-10}, or $3.17\times$ the predicted value. The departure remains below 0.05% through step 5,500.

The untied branch has the smallest marker value, 1.34×10^{-10}, but also the smallest background, 9.5×10^{-17}, and AUC 1.000. The tied control has background 1.16×10^{-12} and AUC 0.926. Across the four endpoints, marker-to-background ratios 1.4×10^6, 2.0×10^3, 3.7×10^2, and 1.5×10^2 correspond to AUCs 1.000, 0.985, 0.926, and 0.898. Thus, the larger marker value need not yield stronger provenance. Replication on another architecture and on ordinary corpus tokens preserves the distinction between blocked branches and branches receiving output-side gradients. The random-block control likewise retains a departure from pure decay.

Natural and occurrence-matched runs. In the natural Qwen3-0.6B run, 22 tokens occur only in the first 100 sentences and are compared against 128,666 never-seen tokens. Their AUC is 0.995 at step 20,000. Median early-only state remains near 2.3×10^{-4} between steps 1,000 and 20,000, whereas pure decay from step 1,000 would predict 1.3×10^{-12}. These tied rows continue to receive output-side gradients at approximately $23\times$ the rate of never-seen rows; the comparison background rises from 1.7×10^{-15} to 1.1×10^{-10} over the same interval.

A separate replication injects 384 otherwise unused token ids exactly three times each during the first 200 steps, then compares them against 31,606 remaining unused ids. Across three seeds at 10,000 steps, mean AUC is 0.991 and TPR at 1% FPR is 0.675; TPR is effectively zero at 0.1% FPR. The latter distinction limits vocabulary-wide retrieval claims even when ranking performance is high. Two independent natural Qwen3-0.6B runs reach AUC 0.9955, and a distilGPT-2 run with a different tokenizer reaches 0.9935 on 14 early-only tokens after 20,000 steps. The small positive sets in these natural runs motivate the larger occurrence-matched replication.

Same-gradient buffer ablation. We drive passive buffers with identical per-step gradients, varying only the stored transformation and decay rate. With untied embeddings, Table 7 shows that long-memory magnitude accumulation preserves separation better than the shorter buffers tested. The absolute-gradient EMA and squared-gradient EMA reach the same AUC; this does not mean

their tensor values are identical. With tied embeddings, all tested buffers remain at AUC 1.0 over the corresponding ablation horizon, so that setting does not isolate recurrence memory from continuing gradients.

Table 7: Same-gradient ablation with untied embeddings, approximately 3,940 steps after the final token occurrence. AUC compares early-only with never-seen tokens.

Saved statistic and decay	AUC
Squared-gradient EMA, $\beta = 0.999$	0.98
Absolute-gradient EMA, $\beta = 0.999$	0.98
Squared-gradient EMA, $\beta = 0.99$	0.70
Squared-gradient EMA, $\beta = 0.9$	0.50
Signed-gradient EMA, $\beta = 0.9$	0.50

B.4 Coordinate Signatures and Semantic Resolution

Feature provenance does not directly establish membership of a complete sequence. Table 8 varies candidate length and coordinate uniqueness in a distilGPT-2 experiment with 96 candidates per condition, half trained. "Private" tokens occur in no other candidate; "shared" rare tokens are drawn from a common pool. Adding shared coordinates can reduce detection despite increasing candidate length. In this experiment, maximum token-set overlap with another candidate correlates with detectability at $\rho = -0.46$, compared with -0.42 for token count.

Table 8: Token count and coordinate uniqueness varied independently, with the same scoring rule across conditions. These are single-run results.

Candidate construction	Tokens	Private	Shared	AUC
One rare token	1	1	0	1.000
Two rare + one common	3	2	0	0.861
One rare + four common	5	1	0	0.708
One rare + three shared	4	1	3	0.534
Four shared rare	4	0	4	0.430

The tested aggregate score is order-invariant: permuting a candidate while preserving its token multiset leaves its score unchanged. In a controlled overlap sweep, document AUC falls from 1.00 with disjoint vocabulary to 0.78 at 90% overlap and 0.499 when the token sets coincide. This is a limitation of the tested token-score aggregation, not a claim that all optimizer-state statistics are order-invariant.

For natural documents, Table 9 compares distinctive-token aggregation with loss-based scores computed from the same model-only export. Optimizer-state aggregation attains higher AUC in the one-epoch settings but lower TPR at 1% FPR in every setting. After three epochs, both output-based scores perfectly separate the evaluated documents. These measurements support feature-level provenance without implying a stronger document-membership attack.

Table 9: Document-membership scores on distilGPT-2, evaluated on the same checkpoint without shadow models. Entries are AUC (TPR at 1% FPR). LOSS uses document loss; Min-20% uses the least-likely token fraction.

Training setting	v_t aggregate	LOSS	Min-20%
1 epoch, 120 documents	0.967 (0.23)	0.928 (0.42)	0.952 (0.53)
1 epoch, 600 documents	0.963 (0.14)	0.931 (0.41)	0.924 (0.40)
3 epochs, 120 documents	0.965 (0.20)	1.000 (1.00)	1.000 (1.00)
3 epochs, 600 documents	0.959 (0.11)	1.000 (1.00)	1.000 (1.00)

C Multi-Checkpoint Gradient Recovery

This section separates the information exposed by a checkpoint pair from the training content recoverable by a particular inversion attack. The former is determined by the saved recurrence and checkpoint spacing; the latter also depends on the effective batch, model, and numerical structure of the gradient.

C.1 Non-Adjacent Checkpoints Reveal a Weighted Window

For a constant first-moment coefficient β_1, repeatedly substituting $m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$ between optimizer steps $a < b$ gives

$$m_b = \beta_1^{b-a} m_a + (1-\beta_1) \sum_{k=a+1}^{b} \beta_1^{b-k} g_k. \tag{12}$$

The released states therefore determine

$$G_{[a,b]} \equiv \frac{m_b - \beta_1^{b-a} m_a}{1-\beta_1} = \sum_{k=a+1}^{b} \beta_1^{b-k} g_k. \tag{13}$$

At $b = a+1$, this is the optimizer-input gradient of one step. At larger spacing, it is an exponentially weighted sum of gradients evaluated along the intervening training trajectory. It should not be identified with a single-example gradient or, in general, a batch gradient evaluated at one fixed parameter vector. An analogous second-moment difference isolates a weighted sum of squared gradients and does not recover their signs.

Table 10 reports the MNIST spacing experiment. The cosines for non-adjacent checkpoints compare the recovered window with the final step's gradient; their decline does not measure error in recovering the window itself. At spacing one, first-moment recovery yields 72.5 dB reconstruction, compared with 9.7 dB for the weight-difference proxy. At spacing ten, single-target reconstruction falls to 10.1 dB. A separate window-inversion experiment recovers both samples in a two-step window above 20 dB, about half in a four-step window, and essentially only the most recent sample for windows of eight or more steps. These are limits of the evaluated inversion procedures, rather than guarantees that wider checkpoint spacing eliminates leakage.

Table 10: Checkpoint spacing on an MNIST MLP, evaluated over six targets. The moment-based observation is $G_{[a,b]}$ in Eq. 13. Cosines use the final single-step gradient as the reference. The v_t-aware proxy partially corrects the scaling of the weight update. Label accuracy refers to the target reconstructed by gradient matching.

	Gradient cosine			Reconstruction
Spacing	Moment difference	Weight difference	v_t-aware	PSNR (label acc.)
1	1.000	0.27	0.59	72.5 dB (100%)
10	0.42	0.03	-0.13	10.1 dB (33%)
50	0.22	0.02	0.01	8.1 dB (17%)
100	0.22	0.02	0.03	8.5 dB (33%)
500	0.28	0.02	0.02	8.5 dB (33%)
2000	0.11	0.02	0.01	6.8 dB (33%)

C.2 Reconstruction Protocol and Attack-Specific Limits

Image reconstruction. MNIST and CIFAR-10 reconstructions use gradient matching with total variation regularization. For adjacent checkpoints, the inversion model uses the pre-update weights from the earlier checkpoint, where the recovered gradient was computed. The MNIST ensemble uses six restarts: three cold starts and three warm starts obtained by adding small Gaussian perturbations to the best current solution. Each restart retains its lowest-loss iterate; the final reconstruction averages the best three using inverse-loss weights. A separate single-restart evaluation reaches approximately 73 dB on MNIST, so the ensemble mainly improves difficult targets.

Batch size affects this optimization-based image attack. With three batches per setting, per-image PSNR at batch sizes $1, 2, 4, 8, 16$ is respectively 62.2 ± 15.9, 50.7 ± 20.6, 28.5 ± 12.4, 13.6 ± 8.5, and 10.1 ± 4.9 dB. Label recovery decreases from 100% to 60% across the sweep. These measurements use a separate target set from Table 10. More pretraining also produces heterogeneous outcomes: some well-fit examples have weak gradients, whereas difficult examples can still support substantial reconstruction.

Additional modalities and text ordering. Table 11 summarizes additional adjacent-checkpoint experiments under their respective model and data settings. They establish that the recovered gradient can support existing inversion procedures across modalities; their scores are not a controlled comparison between modalities. For the synthetic Adult-style task, the recovered gradient has cosine 0.999, with per-feature errors of 0.4–1.6% of the normalized range. For text, nonzero embedding-gradient rows identify a token bag, and a gradient-matching procedure evaluates candidate permutations using order-sensitive parameters. On a separate paired set of 12 sentences, accepting improving pairwise swaps raises ordered longest-common-subsequence (LCS) score from 0.43 with language-model-probability ordering to 0.63. The 0.48 LCS and 83% token recall in Table 11 are end-to-end measurements on 20 other sentences before this reordering stage; they must not be treated as the baseline of the paired comparison.

Table 11: Additional reconstruction measurements from adjacent checkpoints. Each row uses its own evaluation setup. The quality column reports image PSNR for image data and gradient cosine for tabular data; label accuracy is exact label recovery. CIFAR-10 reconstruction preserves coarse class, shape, and color information rather than pixel detail. For text, † denotes the 20-sentence end-to-end evaluation before reordering, whereas ‡ denotes a separate paired 12-sentence ordering comparison.

Data	Model	Quality	Label acc.	Token recall	Ordered LCS
MNIST	Three-layer MLP, $B = 1$	73 dB	100%	—	—
Synthetic Adult-style	MLP	0.999	100%	—	—
CIFAR-10	ConvNet+BN	16 dB	100%	—	—
Text	GPT-2-family LM	—	—	83%†	$0.48^{\dagger}$ / $0.43 \rightarrow 0.63^{\ddagger}$

The role of gradient signs. A controlled MNIST comparison isolates sign information while holding the gradient magnitudes fixed. Reconstruction from the signed gradient reaches 59.3 dB; stripping its signs reduces PSNR to 13.6 dB; restoring the true signs to the same magnitudes returns 59.4 dB. This comparison supports the distinction between signed-gradient recovery and recovery from a magnitude state. It does not imply that magnitude information is harmless.

C.3 Span-Based Token Inference

We also evaluate the span check underlying DAGER Petrov et al. (2024). For a linear layer with column-vector inputs x_i and output derivatives δ_i, the weight gradient has the form

$$\nabla W = \sum_i x_i \delta_i^{\top}. \tag{14}$$

Its column space is contained in the span of the input vectors. Equality requires sufficient rank in the output derivatives, and a candidate lying in that space need not uniquely identify a training token. The attack uses this structure as an inference test rather than an unconditional membership equivalence. For GPT-2, a candidate token v at position p gives the layer-zero input $\mathrm{ln}_1(\mathrm{wte}[v] + \mathrm{wpe}[p])$, computable from the saved model. The check compares candidate inputs with the subspace of an attention projection's weight gradient; it does not require embedding optimizer state.

Across the evaluated batch sizes 1–32, checkpoint-recovered and directly observed gradients produce identical token recall under this check. A separate geometry sweep evaluates 73 configurations on directly observed gradients, covering GPT-2 widths 768, 1024, and 1280, batch sizes 1–128, and sequence lengths 8–128. Among the 68 configurations with numerical rank below 0.9 times the hidden width, mean recall is 0.937 and minimum recall is 0.750; the five width-saturating configurations average 0.648. Thus, the effective batch and its realized rank constrain this particular inference

procedure. Recovery of the optimizer-input gradient does not itself guarantee recovery of individual examples, ordered text, or every contributing token.

C.4 Training-Pipeline Operations

Table 12 evaluates accumulation, simulated data-parallel reduction, clipping, and bf16 autocast on distilGPT-2. Relative recovery errors range from 2.0×10^{-7} to 4.8×10^{-6}. These operations change the gradient consumed by the optimizer, while the first-moment recurrence continues to expose that resulting tensor. Accumulating four micro-batches of size two matches the direct batch-eight result in numerical rank and token recall. Global norm clipping rescales the gradient by a positive scalar and preserves the subspace used by this attack; it leaves recall at 0.930 in the corresponding comparison. Mixed precision changes the measured numerical rank and gives recall 0.915. The combined setting reaches the hidden width and reduces recall to 0.560. None of these measurements implies that all pipeline choices yield the same reconstruction quality.

Additional arithmetic checks cover `foreach` and `fused` AdamW, fp16 autocast with loss scaling, decoupled weight decay, and a cosine learning-rate schedule. In the tested configurations, gradient cosine is 1.00000000 and relative error is approximately 10^{-7}. Direct first-moment recovery contains no learning-rate factor: varying only the final step's rate over five orders of magnitude leaves the measured error at this level. This differs from reconstructing a deleted moment through the weight update, which divides by the learning rate and requires a weight-decay correction (Appendix E).

Table 12: Optimizer-input gradient recovery under different training operations on distilGPT-2, sequence length 11 and hidden width 768. Workers denote simulated data-parallel reduction; accumulation counts micro-batches. Rank is the measured numerical gradient rank and recall is the span-based token recall.

Configuration	Eff. batch	Rel. error	Rank	Recall
$B = 1$	1	2.7×10^{-7}	10	0.909
$B = 8$	8	3.5×10^{-7}	80	0.930
$B = 32$	32	3.2×10^{-7}	315	0.923
$B = 2$, accumulation 4	8	2.7×10^{-7}	80	0.930
$B = 2$, accumulation 16	32	4.6×10^{-7}	319	0.929
$B = 4$, 8 workers	32	2.0×10^{-7}	315	0.923
$B = 8$, clipping at 1.0	8	3.2×10^{-6}	80	0.930
$B = 8$, bf16 autocast	8	3.0×10^{-7}	725	0.915
$B = 8$, accumulation 4, 4 workers, clipping, bf16	128	4.8×10^{-6}	768	0.560

D Optimizer and State-Representation Generality

The two exposure regimes select different properties of an optimizer state. Single-checkpoint provenance depends on distinguishable accumulated activity; adjacent-checkpoint recovery depends on which transformed gradient the recurrence stores. We examine both properties across optimizer representations and then report supporting results on model scale, weight tying, and parameter-efficient fine-tuning (PEFT).

D.1 Single-Checkpoint Provenance Across Optimizers

Table 13 reports, for each of nine optimizers on distilGPT-2, the state with the largest observed single-checkpoint magnitude gap together with its AUC and top-K recall. The non-quantized, long-memory nonnegative states in this comparison yield AUCs of 0.958–0.997. Signed Lion and SGD momentum buffers yield 0.934 and 0.847. The quantized second state of 8-bit Adam gives a weaker but still above-chance AUC of 0.721. Accordingly, the full nine-optimizer range is 0.721–0.997; restricting the comparison to the eight non-quantized optimizers gives 0.847–0.997. The selected buffer is not necessarily the one used for gradient recovery: Sophia-G's curvature estimate has the strongest provenance signal in this comparison, whereas its signed first moment supports gradient recovery.

Table 13: Single-checkpoint provenance across nine optimizers on distilGPT-2. For each optimizer, the table reports the state with the largest observed single-checkpoint magnitude gap. Gap is $10 \log_{10}$ of the median active-to-inactive row-norm ratio. AUC compares trained and never-trained token rows; top-K recall uses K equal to the number of trained rows.

Optimizer	Saved state	Gap (dB)	AUC	Top-K recall
Adafactor	`exp_avg_sq_row`	48.0	0.996	0.95
Adafactor with momentum	`exp_avg_sq_row`	47.9	0.997	0.96
Adam	`exp_avg_sq`	45.0	0.996	0.95
AdamW	`exp_avg_sq`	45.0	0.996	0.96
RMSprop	`square_avg`	43.1	0.978	0.86
Sophia-G	`hessian`	37.5	0.958	0.82
Lion	`exp_avg`	13.2	0.934	0.76
SGD with momentum	`momentum_buffer`	10.1	0.847	0.62
8-bit Adam	`state2`	0.6	0.721	0.46

D.2 Adjacent-State Recovery by Representation

Table 14 distinguishes signed, magnitude, factored, and quantized representations; it includes Adagrad as an additional cumulative-state control. Full-precision signed buffers recover g_t with relative errors of 10^{-8}–10^{-7} where reported, or cosine 1.000, and sign agreement 1.000. Correct recurrence coefficients are essential. In the evaluated Lion implementation, the stored momentum uses β_2, whereas β_1 controls the update direction. For Sophia, recovery uses `exp_avg`, not the curvature buffer.

Magnitude-state differences recover squared-gradient information. Taking the square root gives $|g_t|$ with relative error approximately 10^{-4} in the tested AdamW, RMSprop, and Adagrad configurations, without restoring signs. Adafactor stores row and column marginals rather than the full second-moment tensor. Accounting for the evaluated implementation's scheduled decay $1 - t^{-0.8}$ recovers the stored projection of g_t^2 at cosine 1.000; substituting a fixed decay gives only 0.30 in the evaluated case.

Table 14: Adjacent-state recovery by stored representation. Relative error compares the recovered quantity with its reference; for magnitude and cumulative states, the recovered squared-gradient information is square-rooted before comparison. Cosine is reported for a stored factored projection or a dequantized tensor where applicable. Sign agreement is the fraction of coordinates matching the sign of g_t; for magnitude outputs, it does not indicate recovery of the missing signs. Dashes denote unreported or inapplicable measurements.

Saved state	Type	Recovers	Relative error	Cosine	Sign agreement
AdamW `exp_avg`	Signed	g_t	10^{-7}	—	1.000
Adam `exp_avg`	Signed	g_t	10^{-7}	—	1.000
SGD momentum	Signed	g_t	10^{-8}	—	1.000
Lion `exp_avg`	Signed	g_t	—	1.000	1.000
Sophia `exp_avg`	Signed	g_t	—	1.000	1.000
AdamW `exp_avg_sq`	Magnitude	$\lvert g_t \rvert$	10^{-4}	—	0.531
RMSprop `square_avg`	Magnitude	$\lvert g_t \rvert$	10^{-4}	—	0.457
Adagrad `sum`	Cumulative	$\lvert g_t \rvert$	10^{-4}	—	0.513
Adafactor marginals	Factored	Projection of g_t^2	—	1.000	—
8-bit Adam	Quantized	Lossy g_t	—	0.377	0.649

The evaluated 8-bit Adam representation lowers gradient cosine to 0.377 and sign agreement to 0.649. Its numerical gradient rank increases from 39 to the full hidden width of 768, and downstream token recall decreases from 0.946 to 0.622. This comparison uses the optimizer-representation experiment; the 0.946 baseline is distinct from the 0.933 baseline of the native framework experiment.

Both factoring and quantization change the exposed information, but in different ways: factoring retains a specific projection, whereas the tested quantizer introduces numerical error in the recovered tensor. These measurements cover one factored optimizer and one quantizer and do not establish a general privacy guarantee for either storage choice.

D.3 Model Scale, Weight Tying, and Parameter Sites

Table 15 reports full-parameter fine-tuning for 300 steps on the same supervised fine-tuning data, using fp32. The 8.2B-parameter run uses FSDP across three GPUs. Optimizer-state AUC remains 0.996–0.998 across the evaluated sizes. The displacement baseline uses only $\|\theta_t[C(z)]-\theta_0[C(z)]\|_2$ for each token row $C(z)$ and requires the starting model; it is not the learned weight-only attacker reported elsewhere in the paper. In these measurements, its operating-point performance differs strongly with weight tying. It reaches at most 0.01 TPR at 1% FPR on the tied models, but 0.99 on the untied 8.2B model. Because size and weight tying vary together in the last row, this table alone does not isolate their causal effects.

Table 15: Full-parameter fine-tuning for 300 steps in fp32. Each score is AUC (TPR at 1% FPR), with the displacement-score orientation selected on a disjoint calibration run as in Appendix B.2. These measurements use a different training duration from the 500-step provenance result in the main text.

Model	Parameters	Tied embedding	v_t score	Weight displacement
Qwen3-0.6B	0.6B	Yes	0.998 (0.95)	0.757 (0.00)
Qwen3-1.7B	1.7B	Yes	0.997 (0.91)	0.532 (0.01)
Qwen3-4B	4.0B	Yes	0.996 (0.90)	0.856 (0.00)
Qwen3-8B	8.2B	No	0.997 (0.99)	0.998 (0.99)

Structured sites beyond embeddings can retain weaker token information. After removing embedding optimizer state in Qwen3-0.6B, an adaptive scorer aligning token activation signatures with residual v_t in per-head Q/K normalization weights and FFN gate/up projections reaches AUC 0.625. With attention-only LoRA, the frozen embedding has no optimizer state, but a LoRA-aware scorer using adapter moments reaches AUC 0.68. These results use site-specific adaptive scorers and should not be interpreted as the same direct token-row attack used for embeddings. They show that a change in trainable parameters can relocate the observed signal; they do not establish uniform leakage for all PEFT configurations.

E Sanitization Diagnostics

This section gives the complete sanitization results behind the main-text summary. All experiments use the distilGPT-2 setup unless stated otherwise. A defense is evaluated against provenance and checkpoint-trace attacks at the same time. "Naive" trace attacks read a shipped signed first moment, whereas "adaptive" attacks reconstruct a deleted first moment from retained weights and second-moment state when those quantities remain available. If no optimizer state remains, the adaptive route uses weight-difference inversion. The distinction is essential: a zero score for a naive attack after deleting $\mathbf{m}$ does not imply that the trace channel has been removed.

E.1 Full defense matrix

Table 16 expands the compact matrix in the main paper. Provenance is the embedding-row AUC. Embedding recall reads the embedding first moment, and attention recall is the layer-0 attention span check. Resume loss is the held-out loss after 200 continuation steps, averaged over three seeds; the undefended value is 6.63. The strongest available trace route is reported separately in the adaptive matrix below.

Sharding only changes where the state is stored, and order-preserving quantization leaves the row ordering and both trace routes intact. Removing only embedding state closes the embedding-row tests but leaves attention-based recall at 0.92. Removing all first moments closes the naive trace

Table 16: Full sanitization matrix on distilGPT-2. "Same" means that the corresponding attack score is the same as for the undefended artifact. The state-removal column reports the fraction of optimizer-state bytes removed. The 0.28 value for `strip_all` is the rounded weight-only recall (the unrounded value is 0.275).

Defense	Operation	Prov.	Emb.	Attn.	Trace	Removed	Loss
None	retain full optimizer state	1.00	0.55	0.92	0.950	0%	6.63±0.09
Sharding	split state across workers	same	same	same	same	0%	n/a
Naive quantization	order-preserving low-bit $\mathbf{v}$	1.00	same	same	same	0%	n/a
`strip_embed`	remove $\mathbf{m}/\mathbf{v}$ at tied embedding	0.50	0.00	0.92	0.92	47%	6.75±0.08
Inactive-row lifting	raise inactive $\mathbf{v}$ rows	0.51–0.83	0.55	0.92	same	0%	6.21±0.05
Strip $\mathbf{m}$ globally	drop all first moments	1.00	0.00	0.00	0.950	50%	6.62±0.06
Strip $\mathbf{m}$ + lifting	drop $\mathbf{m}$ and sanitize $\mathbf{v}$	0.51	0.00	0.00	0.950	50%	6.28±0.07
`strip_all`	drop all optimizer state	0.50	0.00	0.00	0.28	100%	7.00±0.04

attack, but not the adaptive one below. Only dropping all optimizer state removes the optimizer-state channel itself; it still leaves the weight-only recall of 0.275 (reported as 0.28 in the table) and incurs the largest resume loss.

E.2 Cross-state redundancy and exact first-moment recovery

Deleting a first-moment tensor does not remove it when the released weight trajectory and second moment determine it. We make the indexing explicit. Let checkpoint C_{t-1} contain the parameters $\boldsymbol{\theta}_{t-1}$ just before optimizer step t, and let C_t contain $\boldsymbol{\theta}_t$ and the post-step second moment $\mathbf{v}_t$. For AdamW with moments initialized to zero, step counter t, learning rate η_t, and decoupled weight decay λ, the post-step updates are

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t, \tag{15}$$

$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\mathbf{g}_t^2, \tag{16}$$

$$\widehat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1-\beta_1^t}, \qquad \widehat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1-\beta_2^t}, \tag{17}$$

$$\boldsymbol{\theta}_t = (1-\eta_t\lambda)\boldsymbol{\theta}_{t-1} - \eta_t \frac{\widehat{\mathbf{m}}_t}{\sqrt{\widehat{\mathbf{v}}_t}+\epsilon}. \tag{18}$$

Therefore, if $\mathbf{v}_t$ and the optimizer hyperparameters are retained,

$$\widehat{\mathbf{m}}_t = -\frac{[\boldsymbol{\theta}_t - (1-\eta_t\lambda)\boldsymbol{\theta}_{t-1}]\left(\sqrt{\widehat{\mathbf{v}}_t}+\epsilon\right)}{\eta_t}, \qquad \mathbf{m}_t = (1-\beta_1^t)\widehat{\mathbf{m}}_t. \tag{19}$$

The decay correction must be inside the bracket. Omitting it gives relative errors of 1.6×10^{-3} at $\lambda = 0.01$ and 1.8×10^{-2} at $\lambda = 0.1$, whereas the corrected expression has error 9×10^{-6} under the tested cosine schedule and bf16 autocast. The expression assumes the conventional post-step AdamW state; implementations with a different save point should use the corresponding step counter and pre/post-step convention.

With $\mathbf{m}$ removed, recovering $\mathbf{g}_t$ requires three consecutive checkpoints, not two weight snapshots. From (C_{t-2}, C_{t-1}, C_t), Eq. 19 reconstructs $\mathbf{m}_{t-1}$ from $(\boldsymbol{\theta}_{t-2}, \boldsymbol{\theta}_{t-1}, \mathbf{v}_{t-1})$ and $\mathbf{m}_t$ from $(\boldsymbol{\theta}_{t-1}, \boldsymbol{\theta}_t, \mathbf{v}_t)$. The optimizer-input gradient then follows exactly as

$$\mathbf{g}_t = \frac{\mathbf{m}_t - \beta_1 \mathbf{m}_{t-1}}{1-\beta_1}. \tag{20}$$

If signed first moments are shipped, only two adjacent state snapshots are needed; the third checkpoint is required only when both first moments must be reconstructed from the weight trajectory. In the native PyTorch and Accelerate tests, reconstructed $\mathbf{m}$ has cosine 0.9999999 and relative error 7.2×10^{-5}. On distilGPT-2, the relative errors for $\mathbf{m}$ and the resulting gradient are 1.3×10^{-5} and 1.7×10^{-4}, respectively, yielding token recall 0.950, identical to the undefended trace attack.

E.3 Adaptive Sanitization Diagnostics

Table 17 separates the attack that reads a saved first moment from the attack that reconstructs it. When the second moment is retained, the adaptive route reconstructs the first moment from the weight trajectory; the $\boldsymbol{\theta}$-only row instead uses the weight-difference route. The "lifted (structured)" row uses the structured, shape-matched sanitizer: each inactive row is filled from the within-row profile of a genuinely active donor and then rescaled. It is not a learned sanitizer. The learned component is the gradient-boosted attacker, trained on a disjoint shadow run and transferred to the target checkpoint. This distinction matters because a sanitizer can match the hand-designed row-norm and dispersion statistics while a learned attacker still detects a joint statistic.

Table 17: Adaptive evaluation of released checkpoint contents (distilGPT-2, three consecutive checkpoints). Naive trace recall reads saved $\mathbf{m}$; adaptive recall reconstructs it using Eq. 19 when $\mathbf{v}$ is retained, whereas the $\boldsymbol{\theta}$-only row uses weight-difference inversion. Provenance is reported for the row-norm and scale-invariant within-row dispersion tests.

Shipped artifact	Naive trace	Adaptive trace	Row norm	Dispersion
$\boldsymbol{\theta}, \mathbf{m}, \mathbf{v}$ (none)	0.950	0.950	1.000	0.782
$\boldsymbol{\theta}, \mathbf{v}$ (strip $\mathbf{m}$)	0.000	0.950	1.000	0.782
$\boldsymbol{\theta}, \mathbf{v}$ lifted	0.000	0.950	0.506	0.966
$\boldsymbol{\theta}, \mathbf{v}$ lifted (structured)	0.000	0.950	0.521	0.502
$\boldsymbol{\theta}$ only	0.000	0.275	0.500	0.500

The uniform inactive-row lift reduces the row-norm AUC to 0.506 but makes within-row dispersion more distinctive, increasing its AUC to 0.966. The structured lift repairs both hand-designed tests (row norm 0.521 and dispersion 0.502), yet the transferred learned attacker still obtains AUC 0.854. An ablation of its descriptor families gives AUC 0.53 for shape alone, 0.54 for the sorted spectrum alone, and 0.73 for the three magnitude statistics (log norm, log mean, and log median). Thus reproducing the statistics used to design a sanitizer is necessary but not sufficient.

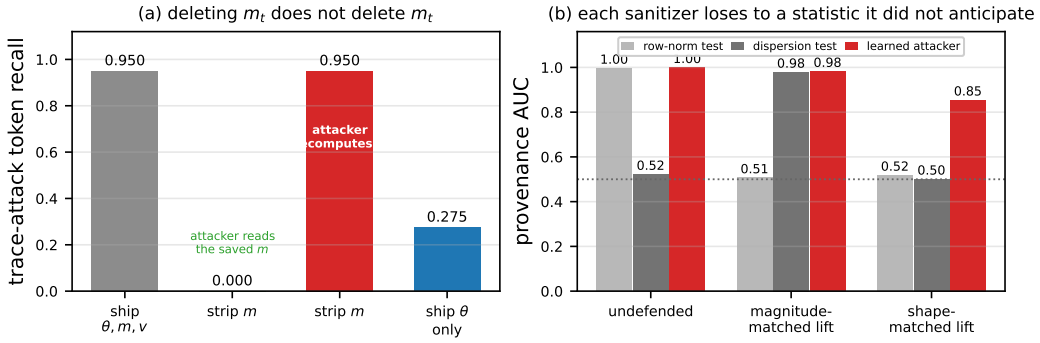


Figure 5: Sanitization diagnostics on the embedding-row channel. (a) Token recall for a trace attacker reading the saved first moment or recomputing it from the weight trajectory and retained second moment. (b) Provenance AUC for the row-norm test, a scale-invariant within-row dispersion test, and a gradient-boosted classifier transferred from a disjoint shadow run. The lift threshold is set from the true active fraction; the structured lift is the shape-matched sanitizer. Results in panel (b) are averaged over two shadow/target pairs in a separate replication.

E.4 Calibration, Resume Utility, and Threshold Sensitivity

A threshold calibrated on one run does not transfer reliably to another because both the score distribution and the number of trained rows change. We therefore calibrate each checkpoint against rows known to be untrained, such as reserved or unused token ids, and set the operating point at 1% false-positive rate within that checkpoint.

Removing $\mathbf{m}$ globally has almost no measured resume cost (loss 6.62 versus 6.63 undefended), whereas removing all optimizer state raises the loss to 7.00. To test adaptation to newly important tokens, we train phase A without a token group G, apply each edit, and resume on phase B where G is frequent. After 300 continuation steps, the loss on G-text is 6.02 for the undefended run, 5.69 after inactive-row lifting, and 6.78 after `strip_embed`; the corresponding improvements are $+0.07$, $+0.41$, and -0.69.

Inactive-row lifting is threshold-sensitive. At lift quantiles $0.990, 0.995, 0.997, 0.9973$, and 0.999, provenance AUC is respectively $0.83, 0.72, 0.56, 0.52$, and 0.51. The 0.9973 setting matches the true active fraction of 0.27% in this experiment. This tuning result does not establish a universally safe quantile; it shows why the sanitizer must be reevaluated against the actual checkpoint and an attacker that can choose its statistic after seeing the edit.