A RELEASED LORA ADAPTER NAMES THE DOCUMENT IT WAS TRAINED ON

Anonymous authors
Paper under double-blind review

ABSTRACT

LoRA adapters are low-rank factors smaller in size compared to the base models they are fine-tuned on. Over 130,000 LoRA adapters are publicly posted on Hugging Face alone. Anyone who downloads an adapter receives it without knowing the data it was fine-tuned on. Prior studies of these weights have produced a scalar or a verbal description of the training data, plus a generative method reports ROUGE-2 below 9% on the source text, thereby concluding that extraction is extremely difficult. In this work, we show that a released adapter identifies the exact document it was fine-tuned on. Our method retrieves that document from roughly 570k candidates for all 256 held-out adapters trained for five steps and shows a 0.01% false positive rate across millions of shortlisted negatives. In that same setting, greedy decoding reproduces no document verbatim, so identification does not require the text to be recoverable. On a second base model, we refit the method end-to-end and it again identifies every document, while decoding reproduces at most one of 256. We trace the signal to the gauge-invariant product $\Delta W = A^\top B$. A kernel over dW alone with no base model, tokenizer or forward pass, ranks the training document at C-AUC 0.897 among 16 candidates. We also measure the bounds of this method. We see that by eighty steps greedy decoding can recover the document and training one adapter on several documents dilutes which one is named. More broadly, our work shows that publishing an adapter can disclose the document behind it and privacy assessments must extend from a model's outputs to its weights.

1 INTRODUCTION

Fine-tuning open-weight foundation models via Low-Rank Adaptation (LoRA) Hu et al. (2022) is widely treated as a safe mechanism for sharing model capabilities. An adapter is only a few megabytes and over 130,000 of them are publicly posted on Hugging Face alone (Hugging Face, 2026). Most systems train one adapter per document (Su et al., 2025; Charakorn et al., 2026). The adapter file posted online does not hold any of the training texts, so releasing it online decouples it from the data it was trained on.

Current parameter forensics work by training auxiliary neural decoders to reconstruct training sentences token by token (Chen et al., 2026a). When evaluated this way, auxiliary decoders struggle with unconstrained text synthesis, reporting ROUGE-2 scores below 9% and concluding that parameter extraction from LoRA weights is practically intractable. However, this blurs the line between whether a decoder can write the document out from weights and whether the weights can be used to identify their training document. For instance in copyright disputes, a data owner wants to know whether their document was used to train an adapter that was posted by a third-party. Therefore, we frame parameter auditing as a problem of *candidate identification* to test whether released weights can identify their specific training document among large candidate pools.

We envision this as a tool for source identification in copyright and data compliance disputes. Increasingly, copyright holders and auditors inspect released models to verify whether proprietary texts were used during any part of model training, which frequently includes fine-tuning with LoRA adapters. We show that candidate identification from weights succeeds, our two-stage method retrieves the exact 150-word training passage for all 256 held-out victim adapters from an open can-

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

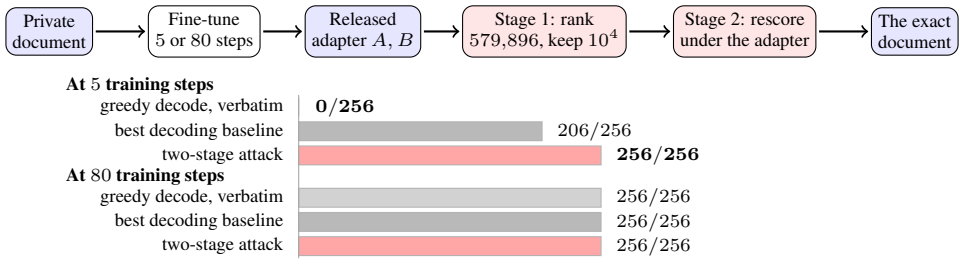

Figure 1: The two-stage method and what it recovers. **Top:** stage one predicts a tf-idf vector from the adapter's weights, ranks all 579,896 candidates and keeps 10,000, then stage two reranks the shortlist by how much the adapter lowers each candidate's loss relative to the base model. **Bottom:** how many of the 256 held-out adapters each method gets right. At 5 steps, greedy decoding reproduces no document verbatim, and the two-stage method still ranks every document first. While at 80 steps, all three succeed.

didate universe of 579,896 documents, with a 0.01% false-positive rate across 2,559,744 negative candidate evaluations.

We train Gemma-2-2B-it adapters, each on one 150-word document. Solely using the released factors and the public base model, our method ranks the exact training document first among 579,896 candidates (Figure 1) for all 256 held-out adapters, whether trained for five steps or eighty. Similarly on Qwen2.5-3B-Instruct, our method names all 256 while greedy decoding reproduces at most one.

To summarize, our contributions in this work are as follows:

- **Identification at scale.** With solely the public base model and a released adapter, our method identifies its training document among 579,896 candidates for every held-out adapter.
- **Identification without verbatim reproduction.** After five training steps, greedy decoding reproduces 0 of 256 documents verbatim on Gemma and at most 1 on Qwen, while the method names all 256 on both. The best decoding baseline names 206 on Gemma, and on Qwen a baseline handed each document's first token comes within 0.016 $[0.004, 0.031]$ of the method (§4.1).
- **Identification without running a model.** Our weights-only probe ranks candidate documents directly from the weight updates with no forward pass or base model execution at evaluation time (§4.2, §5.1).

2 Background and Related Work

2.1 Low-Rank Adaptation and Releasing Adapters

LoRA (Hu et al., 2022) freezes a weight matrix $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ and trains two factors $A \in \mathbb{R}^{r \times d_{\text{in}}}$ and $B \in \mathbb{R}^{r \times d_{\text{out}}}$ with $r \ll \min(d_{\text{in}}, d_{\text{out}})$. We write the update as $\Delta W = A^\top B \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$. A released adapter is the list of factor pairs $(A^{(\ell)}, B^{(\ell)})$ for its L adapted layers. For Gemma-2-2B-it with r=8 on `down_proj` (d_{in}=9216, d_{out}=2304, L=26), that is 2.4 million parameters, about 4.8 MB in `float16` and $< 0.1\%$ relative to the base model.

2.2 Reading Training Data from Weights

Trained weights have long been used as inputs, to predict a network's accuracy (Unterthiner et al., 2020). For fine-tuned language models, Morris et al. (2025) use the base and fine-tuned checkpoints to select documents that reproduce the fine-tune's performance from a large public corpus. Their output is a substitute training set, however they do not attempt to identify a specific document. Horwitz et al. (2024) recover a model's pre-fine-tuning weights from a few of its LoRA fine-tunes. On LoRA weights specifically, W2T (Han et al., 2026) streamlines updates with QR decomposition and Singular Value Decomposition (SVD) to classify adapter attributes and retrieve adapters.

Putterman et al. (2024) build GL-equivariant networks and predict which of 19 data sources an adapter saw. DIT (Goel et al., 2026) trains a model to describe its own weight change in words. These LoRA methods are really useful and productive while finding the properties of a training dataset, however none of them attempt to name a specific training document.

2.3 Writing documents into weights, and reading them back

Methods like parametric RAG (Su et al., 2025), and hypernetworks (Ha et al., 2017) turn text into adapter weights. For knowledge-editing updates, the edited subject and its prompt template can be recovered from the weight updates (Sun et al., 2026). Gradient inversion reconstructs inputs from shared gradients (Zhu et al., 2019), including sentences from LoRA gradients in federated fine-tuning (Chen et al., 2026b), and text embeddings can be inverted to their inputs (Morris et al., 2023). DistilledPRAG (Chen et al., 2026a) is the closest prior work testing whether such documents can be read back from the weight updates. It trains a decoder on document–adapter pairs and reports that ROUGE-2 recall never exceeds 9%, concluding that recovering document content from the weights is extremely difficult.

2.4 Membership inference and extraction

A membership inference method asks whether a given record was in the training data (Shokri et al., 2017), by thresholding its loss (Yeom et al., 2018) or by calibrating that loss against reference models (Carlini et al., 2022; Zarifzadeh et al., 2024). Min-K% Prob (Shi et al., 2024) and LoRA-Leak (Ran et al., 2025) adapt such tests to pretraining data and to LoRA fine-tunes. On pretrained LLMs these attacks often barely beat random chance (Duan et al., 2024), and a benchmark of private adaptation finds that LoRA gives the highest empirical protection for out-of-distribution data (Marek et al., 2026). Extraction attacks instead prompt a model to emit memorized text (Carlini et al., 2021). Extraction grows with model size, duplication and prompt length (Carlini et al., 2023), and adapters appear less vulnerable to it than head fine-tuning (Mireshghallah et al., 2022).

3 Methodology

Our methodology centers on answering three research questions:

- **RQ1.** With its public base model, can a released adapter identify its exact training document among hundreds of thousands of candidates? Does that require the document to be recoverable by decoding from the adapted model?
- **RQ2.** What in the weights carries the document, and how well does the adapter identify it without needing to run inference on a model?
- **RQ3.** What are the limits of this identification when an adapter is trained on several documents or perturbed before release?

3.1 Threat Model and Auditing Scenarios

We formulate parameter auditing where a LoRA adapter is trained on private text and its factor matrices (A, B) are released. We evaluate two realistic scenarios:

Auditor with the base model. This setting models a rights holder or compliance auditor who holds a GPU, the public base model, its tokenizer, and a large candidate corpus. The auditor operates with a budget of $10{,}000$ forward passes per target adapter and trains $32{,}768$ shadow adapters on disjoint public texts (Shokri et al., 2017). The task is to identify the true training passage within an open candidate universe of $579{,}896$ documents.

Weights-only auditor. This setting models a registry or hosting platform that screens model uploads without running forward passes. The auditor holds only the released adapter file, a gallery of 224 shadow adapters, and 31 reference adapters trained with the same configuration to center scores. The auditor ranks the training document against 15 distractors using weight operations alone.

Neither auditor interacts with a deployed model endpoint or observes the training random seed, data sequence, or optimizer state. Both auditors know the base model architecture, adapted layer names, rank r, and learning rate.

3.2 Target Adapters and Candidate Corpus

Each target adapter is trained on a single document following the training protocol in Appendix A.1. We use Gemma-2-2B-it (Gemma Team, 2024) as the base for LoRA on `mlp.down_proj` across all 26 layers, rank r=8, scaling α=16 and using the AdamW optimizer (Loshchilov & Hutter, 2019) with learning rate 2×10^{-3}, batch size 1, and an independent random initialization of A for every adapter.

We evaluate two training regimes: 80 steps and 5 steps. After 80 steps, the adapted model shows an average perplexity on held-out text that is 100.6 times that of the base model, representing a memorization regime. After 5 steps, held-out perplexity is only 1.04 times the base model, reflecting minimal parameter movement (Appendix E.3). All main evaluation results are reported across 256 held-out target adapters, where each adapter is evaluated by an estimator fit strictly without it.

The target documents are drawn from a pool of $1{,}572$ texts, sampled from $1{,}572$ distinct articles in wikitext-103 (Merity et al., 2017). We curate this distinct-article pool to avoid spurious topical correlation. To test large-scale identification, we construct a candidate pool of $579{,}896$ documents. This collection includes the $1{,}572$ distinct articles and $578{,}324$ additional 150-word passages extracted from disjoint wikitext-103 articles.

3.3 Symmetry-Invariant Readout via Weight Updates

The low-rank factor matrices $A \in \mathbb{R}^{r \times d_{\text{in}}}$ and $B \in \mathbb{R}^{r \times d_{\text{out}}}$ of a LoRA adapter are not unique. For any invertible transformation $G \in GL(r)$, the transformed factor pair $(GA, G^{-\top}B)$ computes the exact same linear map:

$$(GA)^\top (G^{-\top}B) = A^\top G^\top G^{-\top} B = A^\top B = \Delta W. \tag{1}$$

Since random initializations yield distinct factor coordinates for identical weight updates, any classifier operating directly on raw entries of A and B must learn this symmetry along with semantic signals. Proposition 1 formalizes this equivalence:

> **Proposition 1.** *Let (A, B) and (A', B') satisfy $A^\top B = A'^\top B' = \Delta W$ with $\operatorname{rank}(\Delta W) = r$. Then there exists a unique matrix $G \in GL(r)$ such that $A' = GA$ and $B' = G^{-\top}B$.*

Any representation of the adapter factors that is invariant to this change of basis must be a function of the product ΔW alone (proof in Appendix B.1). This applies to explicit canonicalization schemes as well, such as QR decomposition or SVD (Han et al., 2026).

We measure similarity between adapters i and j using the Frobenius kernel over their updates across all L layers:

$$K[i,j] = \sum_{\ell=1}^{L} \langle \Delta W_i^{(\ell)}, \Delta W_j^{(\ell)} \rangle_F = \sum_{\ell=1}^{L} \operatorname{tr}\left((A_i^{(\ell)} A_j^{(\ell)\top})(B_j^{(\ell)} B_i^{(\ell)\top}) \right). \tag{2}$$

This formulation requires only $r \times r$ matrix multiplications, avoiding the materialization of dense ΔW matrices.

We predict document representations from weight geometry using kernel ridge regression:

$$\beta = (K_c + \lambda I)^{-1}(Y - \bar{Y}), \tag{3}$$

where $\lambda = 10^{-4}$ and K_c is the double-centered training kernel. For a test adapter with centered kernel row k_c, the model predicts $\hat{y} = k_c^\top \beta + \bar{Y}$ and scores each candidate document by the cosine similarity between $\hat{y}$ and the document vector. The weights-only probe trains on 224 shadow adapters to predict unit-norm counts of the 400 most frequent content words (Appendix B.1) and centers predictions across 31 reference adapters. Stage one of the two-stage method trains on $32{,}768$ shadow adapters to predict a $20{,}000$-dimensional tf-idf vector.

Table 1: Decoding the adapter versus identifying from it (256 held-out Gemma victims, 579,896 candidates). **D2** at 80 steps is tf-idf retrieval with the decoded text as query; at 5 steps it is the best of five decoding baselines (four reranked by stage two), picked by top-1 accuracy under a rule fixed in advance, so it has no interval. **W** is stage one alone. The decoding baselines get none of stage one's 32,768 shadow adapters. Intervals are Wilson; the margin is a paired percentile bootstrap. At 5 steps, compare **D1** with **C**: decoding reproduces no document, and the method names all 256.

	attacker	80 steps	5 steps
D1	greedy decode reproduces the document verbatim	1.000 [0.985, 1.000]	0.000 [0.000, 0.015]
D2	best decoding baseline, top-1 accuracy	1.000 [0.985, 1.000]	0.805
C	two-stage attack, top-1 accuracy	1.000 [0.985, 1.000]	1.000 [0.985, 1.000]
W	stage one alone, no forward pass, top-1 accuracy	0.352 [0.296, 0.412]	0.332 [0.277, 0.392]
C − **D2**, paired		—	+0.195 [+0.148, +0.242]

We evaluate the weights-only probe using area under the ROC curve (AUC), defined as the fraction of 15 distractors ranked below the true document (chance is 0.5). To verify that performance reflects genuine adapter-to-document alignment, we compare each score against a permutation null where adapter-document pairings are randomly shuffled (Appendix B.2).

3.4 The Two-Stage Identification Pipeline

Searching a pool of 579,896 candidates with teacher-forced forward passes across hundreds of adapters requires over 148 million model evaluations. To resolve this computational bottleneck, we design a two-stage identification pipeline (Figure 1):

Stage 1: Weight-space linear sieve. The first stage ranks all 579,896 candidate documents using the kernel ridge regression fit on ΔW. It predicts the tf-idf feature vector of the target document and retains the top 10,000 candidates, discarding 98.3% of the candidate space without running a forward pass. Training the 32,768 shadow adapters takes 365 minutes on one A100 GPU, and constructing the kernel requires 488 minutes. This offline cost is amortized across all evaluations.

Stage 2: Base-calibrated loss rescoring. The second stage computes the loss difference for each candidate $d = (x_1, \ldots, x_n)$ under the adapted model and the base model:

$$s(d) = \mathcal{L}_{\text{adapted}}(d) - \mathcal{L}_{\text{base}}(d), \tag{4}$$

where $\mathcal{L}(d) = -\frac{1}{n}\sum_t \log p(x_t \mid x_{<t})$ denotes mean teacher-forced token loss. The candidate that minimizes $s(d)$ is selected as the predicted document. Stage two requires only 10,000 forward passes per adapter (approximately 3 GPU-minutes), reducing compute by a factor of 58 compared to evaluating the full candidate set directly.

We report top-1 accuracy, defined as the proportion of adapters whose training document is ranked first among all 579,896 candidates, along with 95% Wilson confidence intervals. For a fixed decision threshold across adapters, we report the true-positive rate (TPR) at a given false-positive rate (FPR). We define verbatim decoding as cases where the first 50 tokens of greedy generation from the adapted model, initiated with the begin-of-sequence token and no prompt, match the first 50 tokens of the true document (Appendix D.1).

4 Results

4.1 RQ1: The adapter identifies its document where decoding does not reproduce it

One threshold for all victims. The two-stage method ranks the exact training document first among all 579,896 candidates for every victim, at both training budgets (top-1 accuracy 1.000 [0.985, 1.000]; row **C** of Table 1). One threshold on the stage-two score detects all 256 training documents (TPR 1.000 [0.985, 1.000]) at a 0.01% false-positive rate over the 2,559,744 wrong

Table 2: The weights-only probe (adapter file and 224 shadow adapters, no model run at method time) against attacks that run the model and controls that use no adapter, on the distinct-article pool. The last column is the score minus its shuffled baseline (0.498 among 16 candidates, 0.502 over all 1,572 documents). [a]Minus the weights-only probe's AUC. [b]32 documents, 80 steps in total (App. E.1); top-1 accuracy among 256 targets. [c]Principal directions of the 256 centered updates removed from the kernel, weights-only probe not refit; random: three seeds, all intervals contain 0 (App. B.3).

method	access	score	above shuffled [95% CI]
Weights-only probe (this paper)			
AUC, 16 candidates	file	0.897 [0.878, 0.914]	+0.399 [+0.378, +0.417]
AUC, all 1,572 documents	file	0.880	+0.378 [+0.357, +0.398]
Attacks that run the model			
loss ranking, AUC, k=1	base + adapter	1.000	+0.103 [+0.085, +0.122][a]
RMIA / LiRA, top-1, k=32[b]	+ 255 ref. adapters	1.000	—
Controls that use no adapter			
corpus position; update norm	file order or norm	0.483–0.501	within 0.02 of chance
Weights-only probe, principal directions removed[c]			
minus the 111 largest	file	—	−0.154 [−0.193, −0.116]
minus the 111 smallest	file	—	+0.254 [+0.221, +0.288]
minus random, same share	file	—	−0.022 to +0.020

shortlisted candidates, at both budgets. Scoring the five-step shortlists by base-model loss alone detects none of them at 0.1% (TPR 0.000 [0.000, 0.015]), so the base model's own losses do not produce the ranking.

Identification without verbatim reproduction. At 80 steps, greedy generation from the adapted model reproduces every training document verbatim, and a tf-idf search of the decoded text names all 256 (rows **D1** and **D2**); a search on its first 20 tokens alone names 157 (0.613 [0.552, 0.671]; App. D.3). At five steps, greedy decoding reproduces none of them (0.000 [0.000, 0.015]). We also built five decoding baselines; four of them build a shortlist from decoded text and pass it to the same stage two. The best names 206 of the 256 documents. The method names all 256, including the 50 that baseline misses, a paired difference of 0.195 [0.148, 0.242] in top-1 accuracy (App. D.1).

Near-duplicate negatives and a second base model. At 80 steps we added to the candidates the 34,544 other documents cut from the pool's wikitext-103 articles, 5,554 of them from the victims' own articles; 3,040 overlap a victim's document, by a median of 110 of its 150 words. The exact training document still ranks first for every victim (App. C.1). On Qwen2.5-3B-Instruct (Hui et al., 2024), with stage one refit on 32,768 Qwen shadow adapters, the method again identifies all 256 five-step victims, while greedy decoding reproduces at most one verbatim. The best decoding baseline that sees no part of the document names 152 of 256, 0.406 [0.348, 0.469] below the attack; one handed each document's first token (Qwen has no begin-of-sequence token) names 252, 0.016 [0.004, 0.031] below (App. F.3).

4.2 RQ2: Identifying the document from the file

Without a forward pass at method time, the weights still rank the document. Stage one alone ranks it first among all 579,896 candidates for 0.352 [0.296, 0.412] of the 80-step victims and 0.332 [0.277, 0.392] of the five-step victims (Table 1, row **W**); stage two, with the base model, makes the answer exact. The weights-only probe ranks the document above 15 distractors with AUC 0.897 [0.878, 0.914], 0.399 [0.378, 0.417] above the shuffled baseline, and reaches AUC 0.880 against all 1,572 pool documents (Table 2). At a 0.1% false-positive rate it detects 7 of 256 targets, and at 1% it detects 43, both short of the rates we had pre-registered (App. B.2).

Run through the same regression, kernels over corpus position or update norm score within 0.02 of chance. Ranking the same candidates by the adapted model's loss scores AUC 1.000, 0.103 [0.085, 0.122] above the weights-only probe. On adapters trained on 32 documents for 80 steps

Table 3: Training one adapter on k documents (80 steps per document; 256 held-out Gemma victims). **target** and **any**: the two-stage method among 579,896 candidates; **any** was reported, not pre-registered. **weights-only**: the weights-only probe, fit on 224 shadow adapters (App. E.1). Dashes: not scored; at k=32 the two-stage method ran only stage one, refit on 16,384 shadow adapters (App. E.1). Down the rows, top-1 on the target falls to about $1/k$ while the top candidate stays one of the adapter's k documents.

	target	**any**	**weights-only**
k	top-1 accuracy	top-1 is one of k	AUC, 16 candidates
1	1.000 [0.985, 1.000]	—	0.897 [0.878, 0.914]
2	0.469 [0.409, 0.530]	1.000 [0.985, 1.000]	—
8	0.105 [0.074, 0.149]	1.000 [0.985, 1.000]	0.534 [0.498, 0.569]
32	—	—	0.514 [0.481, 0.550]

k=32, weights-only AUC above shuffled baseline: +0.009 [−0.028, +0.045], below its pre-registered floor.
k=32, pairs sharing training data vs. none: AUC 0.850 [0.828, 0.874], shuffled labels 0.500.

in total, the reference-model attacks RMIA and LiRA (§2) with 255 reference adapters rank the document first among the 256 targets for every victim, where the weights-only probe scores AUC 0.499 [0.464, 0.537] (App. E.1).

To see which directions of the weights carry the document, we removed principal directions of the 256 centered updates from the kernel without refitting the weights-only probe (Table 2, lower block; App. B.3). The largest directions matter most; because the probe is not refit, the ablation does not show how many directions carry the signal. Removing the 111 largest, which hold half of the centered updates' squared Frobenius norm, pushes the AUC 0.154 [0.116, 0.193] below the shuffled baseline. Removing random directions with the same share of that norm brings it down to that baseline, and removing the 111 smallest still leaves it 0.254 [0.221, 0.288] above.

4.3 RQ3: Several documents per adapter, and perturbations of the file

When each victim is trained on its target document and $k-1$ other documents from the pool, 80 steps per document, top-1 accuracy on the target falls from 1.000 at k=1 to 0.469 [0.409, 0.530] at k=2 and 0.105 [0.074, 0.149] at k=8, where the interval does not clear the floor of 0.10 we had pre-registered (Table 3). Part of the loss happens in stage one: 145 of the 229 misses at k=8 never reach stage two. The method still finds the adapter's training data: at both k=2 and k=8 its top candidate is one of the adapter's k documents for all 256 victims, and greedy decoding, given a budget equal to the target's length, also returns one of them (App. E.1, C.3). What it cannot tell is which of the k an auditor is asking about. The target's place in the training order is random, so $1/k$ is the natural reference; both intervals contain it. With 224 shadow adapters the weights-only probe falls to AUC 0.534 [0.498, 0.569] at k=8, but stage one's regression, fit on 32,768, ranks the target among the same 16 candidates at AUC 0.910 [0.891, 0.927].

At k=32, still 80 steps per document, the weights-only probe no longer ranks the target: AUC 0.514 [0.481, 0.550], 0.009 [−0.028, 0.045] above the shuffled baseline. Given two k=32 adapters, the kernel alone still separates pairs that share training documents from pairs that share none at AUC 0.850 [0.828, 0.874], against 0.500 [0.494, 0.506] with the pair labels shuffled; we cannot tell shared documents from shared topics (Table 3; App. E.2).

Post-hoc changes to the released file remove little of what the weights-only probe reads. We applied three perturbations a publisher could use before release (additive Gaussian noise, rank truncation, and uniform quantization) at 13 settings plus three untouched controls, and report two ratios per setting (Figure 2, App. E.3). R_s is the share of the weights-only probe's AUC above the shuffled baseline that survives the perturbation, so 1 means untouched. R_u is the perturbed adapter's drop in loss on its training document divided by the unperturbed adapter's. Eleven of the 13 settings keep R_u above the pre-registered floor of 0.80. Quantization from 8 down to 3 bits keeps $R_s \geq 0.95$, and only 2-bit quantization has an R_s interval that excludes 1, at 0.762 [0.700, 0.823], at a utility cost of 9.95% [9.21%, 10.78%].

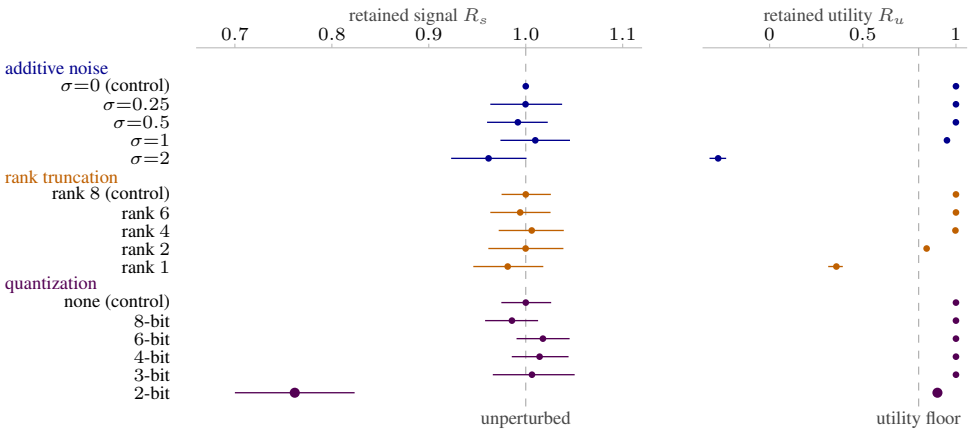

Figure 2: Post-hoc perturbations applied to the released file (16 candidates, distinct-article pool), one row per setting, with 95% intervals. Left: retained signal R_s of the weights-only probe (1 is unperturbed). Right: retained utility R_u; the dashed line is the pre-registered floor $R_u \geq 0.80$ (App. E.3); $R_u < 0$ means the perturbed adapter fits its document worse than the base model. The signal interval contains 1 for every setting except 2-bit quantization (large dot). Full values, including each setting's perturbation norm: Table 7.

5 Further analysis

We ask how far the results extend (§5.1), why identification can precede verbatim reproduction (§5.2), whether position can pass for content (§5.3), which documents are easiest to identify (§5.4), and what a publisher can do (§5.5).

5.1 Other models, corpora and recipes

The weights-only probe also works on a second base model, a second corpus and other training recipes (App. F.1). On Qwen2.5-3B-Instruct adapters trained on the same 256 documents, with the probe refit from scratch, its AUC sits 0.372 $[0.344, 0.398]$ above the shuffled baseline. On Python source (1,572 150-word documents, one per CodeParrot repository; Tunstall et al., 2022) it sits 0.323 $[0.297, 0.348]$ above, likely an underestimate because the 256-token limit cut most of these documents short (App. F.2). Changing the rank or the target modules lowers the AUC, at worst to 0.780 $[0.751, 0.809]$ at r=32 (Table 9). Adapters that a hypernetwork writes in one forward pass can also be read. On 4,000 adapters from the generator released with DistilledPRAG, written from that work's own corpus (2WikiMultihopQA; Ho et al., 2020), a ridge on random projections of the raw factors, used here in place of ΔW (App. F.1), names the document among 32 candidates with top-1 accuracy 0.611 $[0.596, 0.626]$, against a chance rate of 0.031. Their decoder writes the document out and this ridge only names it, so we do not compare the two numbers.

5.2 An account of identification before verbatim reproduction

We read the five-step result as a difference in how training and decoding use the adapter, an account we have not tested directly. Training scores each token with the true prefix in context, so a few steps can lower the loss of the whole document, and stage two measures exactly that loss. Greedy decoding conditions on its own earlier tokens, so one early error can take it off the document. Word overlap fits this reading (App. D.4): the ten words ranked highest by a third fit of the regression (16,384 shadow adapters, a 400-word target) overlap the document more than the top ten decoded words at five steps, and less by ten steps.

5.3 Consecutive documents let position pass for content

Our first pool, the contiguous-window pool, cut consecutive 150-word windows from one wikitext-2 stream (Merity et al., 2017), and adjacent windows have 2.60 $[2.51, 2.69]$ times the token-set

Jaccard similarity of random pairs (the adjacency ratio). On that pool, with adapters trained on eight documents each and 32 candidates, the same regression with the Frobenius kernel replaced by $\exp(-|p-q|)$ over the documents' corpus positions p and q, which never reads an adapter, reached AUC 0.959 $[0.948, 0.967]$, against 0.791 with the Frobenius kernel (App. A.2). On the distinct-article pool the adjacency ratio is 0.998 $[0.967, 1.031]$, and the same position kernel scores below chance (AUC 0.483–0.486).

5.4 Which documents are easiest to identify

Documents that the base model predicts poorly are identified by a wider margin. For the 256 victims trained for 80 steps, the stage-two margin over the closest shortlisted competitor correlates with the document's loss under the base model at Spearman 0.728 $[0.660, 0.782]$, and at 0.662 $[0.585, 0.728]$ on Qwen. The weights-only probe's per-document AUCs on the two models correlate at 0.419 $[0.306, 0.527]$, so the same documents tend to be easy on both. Text the base model already predicts well is identified by a narrower margin, but the fitted line, extrapolated, reaches zero margin only at a base loss far below that of 99% of our candidates (App. C.3).

5.5 Mitigation

DP-SGD noise removes a direct match between weights before it removes what the weights-only probe reads. We trained victims on the contiguous-window pool with DP-SGD's per-step clipping and Gaussian noise (Abadi et al., 2016) at lot size one, without privacy accounting (App. E.4). An attacker who can run the Doc-to-LoRA generator (App. E.4) can match each adapter, by the cosine of the Frobenius kernel, to the adapters the generator writes for 32 candidate documents. Its top-1 accuracy falls from 0.984 $[0.961, 0.994]$ under clipping alone to 0.020 $[0.008, 0.045]$ at noise multiplier $\sigma = 0.04$, outside the pre-registered grid, while the adapter keeps 97.8% of its loss drop (Table 8). The weights-only probe, refit on these adapters, still reaches AUC 0.643 $[0.609, 0.673]$. No noise level we tried brought both to chance at a small cost in utility.

Before release, an adapter's owner can test whether the adapter identifies its training documents. The test measures how much the adapter raises the next-token probability of the document's ten rarest content words under four neutral prompts, relative to the base model and to the adapter's average rise over a $1{,}000$-word vocabulary. Against the other 255 victims' documents it reaches AUC 0.928 $[0.904, 0.950]$ and detects 0.668 $[0.605, 0.723]$ of training documents at a 1% false-positive rate, with four forward passes per adapter (App. E.5).

6 Conclusion

In this work, we present a systematic investigation of training-document identification from released LoRA adapters across candidate scales, base models, and training regimes. We find that parameter extraction and candidate identification dissociate. At five training steps, greedy decoding reproduces no document verbatim (0 of 256), while our two-stage method retrieves the exact document from $579{,}896$ candidates for all 256 held-out adapters on both Gemma and Qwen. Our method separates all 256 targets from $2{,}559{,}744$ negative candidate evaluations at a 0.01% false-positive rate, showing that an adapter exposes its training text long before memorization enables generation.

Prior parameter decoders failed because unconstrained generation compounds errors after a single off-target token. Teacher-forced scoring instead evaluates candidates under true prefixes, isolating the coordinated parameter shift written by gradient updates. Without running a forward pass, the gauge-invariant update $\Delta W = A^\top B$ preserves lexical geometry, allowing our probe to rank candidate documents at AUC 0.897 among 16 distractors. Post-hoc rank truncation down to rank 2 and quantization down to 3 bits leave this signal intact ($R_s \geq 0.95$). DP-SGD noise cannot suppress both direct weight matching and probe ranking without degrading utility. Evaluating parameter privacy through generative inversion alone provides false reassurance. Understanding the dissociation between parameter extraction and candidate identification is therefore critical for evaluating data privacy in an era of open-weight models.

AI USE STATEMENT

In accordance with the ICLR policy on LLM usage, we disclose that Claude was used to assist with initial drafting of this paper. We exclusively used them for language refinement, such as improving readability and rephrasing for clarity in an academic writing style. This usage aligns with standard academic practices for language polishing. We also used Cursor for writing software, suggesting experimental parameters, and creating the figures. All content underwent multiple rounds of extensive revision, editing and verification by the authors.

None of the citations or references were created by an LLM, instead all such content was manually checked and entered by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We adhere to the ICLR Code of Ethics. No private, sensitive, or personally identifiable data was involved in conducting this research project. Every LoRA adapter in this study was trained or generated from publicly available text. We did not attempt to probe any deployed systems.

REPRODUCIBILITY STATEMENT

All base models (Gemma-2-2B-it, Qwen2.5-3B-Instruct, and Llama-3.2-1B for the Distilled-PRAG generator) and all corpora (wikitext-2, wikitext-103, codeparrot-clean-valid and 2WikiMultihopQA) are publicly available. We provide full details of the training setup (App. A.1), the document pools (App. A.2, F.2), the regression probe (§3.3, App. B.1), the two-stage method (§3.4) and the evaluation metrics (App. B.2) in the main paper and appendix. Our codebase and the result files from our runs are available at `https://anonymous.4open.science/r/released-lora-adapter-code-7053` with the goal of releasing it publicly on GitHub upon acceptance of the paper.

REFERENCES

Martín Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 308–318, 2016.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In *30th USENIX Security Symposium*, 2021.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. Membership inference attacks from first principles. In *IEEE Symposium on Security and Privacy (SP)*, pp. 1897–1914, 2022. doi: 10.1109/SP46214.2022.9833649.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramèr, and Chiyuan Zhang. Quantifying memorization across neural language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Rujikorn Charakorn, Edoardo Cetin, Shinnosuke Uesaka, and Robert Tjarko Lange. Doc-to-lora: Learning to instantly internalize contexts, 2026.

Jinwen Chen, Hainan Zhang, Liang Pang, Yongxin Tong, Haibo Zhou, Wei Lin, and Zhiming Zheng. Privacy-preserving reasoning with knowledge-distilled parametric retrieval augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026a. doi: 10.18653/v1/2026.findings-acl.939.

Silong Chen, Yuchuan Luo, Guilin Deng, Yi Liu, Min Xu, Shaojing Fu, and Xiaohua Jia. Reconstructing training data from adapter-based federated large language models. In *Proceedings of the ACM Web Conference (WWW)*, 2026b.

Michael Duan, Anshuman Suri, Niloofar Mireshghallah, Sewon Min, Weijia Shi, Luke Zettlemoyer, Yulia Tsvetkov, Yejin Choi, David Evans, and Hannaneh Hajishirzi. Do membership inference attacks work on large language models? In *Conference on Language Modeling (COLM)*, 2024.

Gemma Team. Gemma 2: Improving open language models at a practical size, 2024.

Avichal Goel, Yoon Kim, Nir Shavit, and Tony T. Wang. Learning to interpret weight differences in language models. In *International Conference on Learning Representations (ICLR)*, 2026.

David Ha, Andrew M. Dai, and Quoc V. Le. HyperNetworks. In *International Conference on Learning Representations (ICLR)*, 2017.

Xiaolong Han, Ferrante Neri, Zijian Jiang, Fang Wu, Yanfang Ye, Lu Yin, and Zehong Wang. W2T: LoRA weights already know what they can do, 2026.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics (COLING)*, 2020.

Eliahu Horwitz, Jonathan Kahana, and Yedid Hoshen. Recovering the pre-fine-tuning weights of generative models. In *International Conference on Machine Learning (ICML)*, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Hugging Face. Models tagged `lora` on the Hugging Face Hub. `https://huggingface.co/models?other=lora`, 2026. 134,744 models listed; accessed 2026-09-25.

B Hui, J Yang, Z Cui, J Yang, D Liu, L Zhang, T Liu, J Zhang, B Yu, K Lu, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

Bartłomiej Marek, Lorenzo Rossi, Vincent Hanke, Xun Wang, Michael Backes, Franziska Boenisch, and Adam Dziedzic. Benchmarking empirical privacy protection for adaptations of large language models. In *International Conference on Learning Representations (ICLR)*, 2026.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations (ICLR)*, 2017.

Fatemehsadat Mireshghallah, Archit Uniyal, Tianhao Wang, David Evans, and Taylor Berg-Kirkpatrick. Memorization in NLP fine-tuning methods, 2022.

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

John X. Morris, Junjie Oscar Yin, Woojeong Kim, Vitaly Shmatikov, and Alexander M. Rush. Approximating language model training data from weights, 2025.

Theo Putterman, Derek Lim, Yoav Gelberg, Stefanie Jegelka, and Haggai Maron. Learning on LoRAs: GL-equivariant processing of low-rank weight spaces for large finetuned models, 2024.

Delong Ran, Xinlei He, Tianshuo Cong, Anyu Wang, Qi Li, and Xiaoyun Wang. LoRA-Leak: Membership inference attacks against LoRA fine-tuned language models, 2025.

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 (ICLR)*, 2024.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *IEEE Symposium on Security and Privacy (SP)*, pp. 3–18, 2017. doi: 10.1109/SP.2017.41.

Weihang Su, Yichen Tang, Qingyao Ai, Junxi Yan, Changyue Wang, Hongning Wang, Ziyi Ye, Yujia Zhou, and Yiqun Liu. Parametric retrieval augmented generation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1240–1250, 2025. doi: 10.1145/3726302.3729957.

Zhiyu Sun, Minrui Luo, Yu Wang, Zhili Chen, and Tianxing He. Reverse-engineering model editing on language models. In *International Conference on Machine Learning (ICML)*, 2026.

Lewis Tunstall, Leandro von Werra, and Thomas Wolf. *Natural Language Processing with Transformers: Building Language Applications with Hugging Face*. O'Reilly Media, Sebastopol, CA, 2022.

Thomas Unterthiner, Daniel Keysers, Sylvain Gelly, Olivier Bousquet, and Ilya Tolstikhin. Predicting neural network accuracy from weights, 2020.

Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *IEEE 31st Computer Security Foundations Symposium (CSF)*, pp. 268–282, 2018. doi: 10.1109/CSF.2018.00027.

Sajjad Zarifzadeh, Philippe Liu, and Reza Shokri. Low-cost high-power membership inference attacks. In *International Conference on Machine Learning (ICML)*, 2024.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

A EXPERIMENTAL SETUP

Table 4 lists what each attacker holds. The two-stage attack's shadow adapters are trained on public wikitext-103 documents; a text check before the run confirmed that none is a victim's document. The weights-only probe's 224 shadow adapters are the adapters of the other seven folds (App. B.2). Neither attacker has query access to a deployed model, the victim's random seed or its optimizer state.

Table 4: What each attacker holds. Only the two-stage method runs the base model; the weights-only probe ranks 16 candidates and centers its scores against other adapters of the same recipe. Mismatch costs are in App. F.1.

	two-stage method	weights-only probe
adapter factors A, B of the victim	yes	yes
shadow adapters, with their documents	32,768	224
runs the base model at method time	yes; 10,000 passes per victim	no
candidate set containing the document	579,896	16
base model and target module of the victim	yes	yes
victim's rank and learning rate (shadow adapters match)	yes	yes
victim's step count	not needed	not needed
other adapters of the same recipe, to center scores	no	31
query access to a deployed model	no	no
victim's seed or optimizer state	no	no

Cost. The two expensive parts of the two-stage method are paid once. Training the 32,768 stage-one shadow adapters took 365 minutes on one A100 at five steps each, and building their $33{,}024 \times 33{,}024$ kernel took 488 minutes. Stage one runs no forward pass: it computes one kernel row against the shadow adapters, applies the ridge, and takes a dot product with each candidate's tf-idf vector. Stage two is the only per-victim GPU cost: 10,000 teacher-forced forward passes at batch size one, at 20 ms per candidate, or about 3 GPU-minutes per victim. We score at batch size one because batching to 64 changed the loss at 27 of 128 positions and flipped 17 of 8,128 candidate orderings. Scoring every candidate against every victim without a shortlist would take $579{,}896 \times 256 = 148{,}453{,}376$ forward passes, about 825 GPU-hours at the same rate.

A.1 TRAINING RECIPE

Unless a result states otherwise, every victim and shadow adapter uses this recipe; App. F.1 lists the departures.

- **Base model:** Gemma-2-2B-it; Qwen2.5-3B-Instruct for the second-model results.
- **LoRA:** `mlp.down_proj` in all 26 layers, rank r=8, α=16 (α=2r whenever the rank changes). A is initialized with `kaiming_uniform_` from an unseeded generator, so no two adapters share an initialization.
- **Optimization:** AdamW, learning rate 2×10^{-3}, batch size one, 80 steps (or 5), one document per adapter.
- **Documents:** 150 words, tokenized with `max_length=256`. No wikitext victim reaches that limit; on Python source it truncates 234 of 256 victims (App. F.2).
- **Several documents per adapter:** the target document and $k-1$ other documents from the pool, tokenized separately and visited round-robin, 80 steps per document. The target's position in the order is random.

A.2 DOCUMENT POOLS AND THE POSITIONAL CONFOUND

We measure how much neighboring documents in a pool resemble each other by the *adjacency ratio*: the token-set Jaccard similarity of adjacent documents divided by that of random pairs, so independent documents give 1.

Contiguous-window pool. Our first pool cut 1,572 consecutive 150-word windows from one wikitext-2 stream that spans 9 articles. Its adjacency ratio is 2.60 [2.51, 2.69]. On adapters trained on eight documents each, ranked among 32 candidates, three kernels run through the same regression score as follows: a kernel $\exp(-|p-q|)$ over the corpus positions p and q of two documents, which never reads an adapter, 0.959 [0.948, 0.967]; an indicator of whether two documents come from the same article, 0.914; and the Frobenius kernel over the weights, 0.791. We had pre-registered that the position kernel would score between 0.50 and 0.70; that prediction failed.

Distinct-article pool. The pool behind the main results takes one random 150-word window from each of 1,572 distinct wikitext-103 articles, in shuffled order. Its adjacency ratio is 0.998 [0.967, 1.031]. Over all 1,572 documents, the position kernel scores AUC 0.483–0.486 across decay constants and retrieves no target at a 0.1% false-positive rate; a kernel over update norms scores 0.501; the Frobenius kernel scores 0.880 (Table 2). The 2WikiMultihopQA documents behind the DistilledPRAG result of App. F.1 have an adjacency ratio of 1.016 [0.946, 1.089].

B THE REGRESSION PROBE

B.1 KERNEL, PROOF OF PROPOSITION 1, AND TARGET WORDS

Proof of Proposition 1. $\mathrm{col}(\Delta W) \subseteq \mathrm{row}(A)$ and both have dimension r, so $\mathrm{row}(A) = \mathrm{col}(\Delta W) = \mathrm{row}(A')$. Two full-rank $r \times d_{\text{in}}$ matrices with the same row space satisfy $A' = GA$ for a unique invertible G. Then $A^\top B = A'^\top B' = A^\top G^\top B'$, and $A^\top$ has full column rank, so $B = G^\top B'$, that is, $B' = G^{-\top} B$. □

The rank condition. Proposition 1 needs $\mathrm{rank}(\Delta W) = r$. We count a direction when its singular value σ_r is at least 10^{-3} times the largest, σ_1. The condition holds for 260,635 of the 260,648 (adapter, layer) pairs across the 30 sets of adapters used in this paper, and for all 6,656 pairs of the main Gemma set, whose smallest σ_r/σ_1 is 0.021. The 13 exceptions are 2 pairs in the set trained at a 15× higher learning rate and 11 in the 80-step Qwen set. Moving the stored half-precision factors by half a unit in the last place changes σ_r/σ_1 by at most 5.2×10^{-5} over the 22 sets we checked, so the exceptions are a property of the updates. As a check of the reduction to ΔW, replacing the factors by their QR canonical form changes the kernel by a relative 4.5×10^{-16}, while a perturbation of the same relative size that is not a change of basis changes it by 0.91.

Ridge and target. The kernel is Eq. 2, computed through the $r \times r$ products so that no dense ΔW is formed. The ridge penalty is fixed at $\lambda = 10^{-4}$ for every fold and condition. For the weights-only probe, the target Y counts the 400 most frequent words (runs of three or more lower-case letters) in the 256 documents being scored, after removing a fixed 32-word stop list: `and are been but can for from had has have her him his into its not out she that the their they this was were what when which will with would you`. Each row is scaled to unit norm, and each prediction $\hat{y}$ is scaled to unit norm before the cosine.

The choice of target does not drive the result. Holding the kernel and ridge fixed, we changed only the target, on the contiguous-window pool with 32 candidates. The AUC barely moves across word targets and stays high for character 4-grams, while a random target of the same shape scores chance:

target	dimension	AUC among 32 candidates
word counts (reference)	400	0.954 [0.946, 0.961]
tf-idf weighted words	400	0.957 [0.950, 0.964]
word counts	1,600	0.956 [0.949, 0.963]
word counts	100	0.934 [0.922, 0.944]
character 4-grams	400	0.926 [0.913, 0.937]
random Gaussian (control)	400	0.513 [0.478, 0.549]

The signal is not limited to common words either. With four disjoint bands of 300 words ranked by corpus frequency, the AUC is 0.952 [0.944, 0.960] for ranks 1–300, 0.937 [0.926, 0.948] for 301–600, 0.916 [0.900, 0.930] for 601–900 and 0.893 [0.874, 0.910] for 901–1,200. The band of ranks

601–900 shares no word with the reference target. Among six kernels (raw, scaled, cosine, double-centered, both, and spectrally whitened), all but whitening score within 0.008 of the scaled kernel we use; whitening, which flattens the leading eigenvalues, brings the AUC to 0.496.

B.2 Evaluation, shuffled baseline and fixed false-positive rates

AUC among 16 candidates. The 256 victims are split into eight folds of 32. For each fold, the ridge is fit on the 224 adapters of the other seven folds, which serve as its shadow adapters, and the score matrix S compares each adapter's prediction $\hat{y}$ with each document's target. Before ranking, we subtract from each candidate's score its mean score from the other 31 adapters in the fold. The subtraction removes the part of a document's score that every adapter gives it. Each victim's true document is then ranked against 15 distractors drawn from its fold.

Shuffled baseline. The baseline permutes the rows of Y, which breaks the pairing between adapters and documents and keeps the word distribution, and reruns the whole weights-only probe. We use 20 permutations by default and 200 where stated. Going from 20 to 200 moves the result by 0.0015 $[0.0002, 0.0148]$. Across 32 random fold partitions the AUC has a standard deviation of 0.0053–0.0072, and retraining all 256 victims with fresh initializations gives 0.890 $[0.870, 0.908]$, against 0.897 for the reported run.

Harder distractors. Replacing the 15 random distractors with each document's 15 nearest bag-of-words neighbors among $1{,}316$ held-out pool documents lowers the AUC to 0.705, still 0.202 $[0.170, 0.236]$ above its shuffled baseline. Using the 15 nearest neighbors in a text-embedding space among the other victims' documents gives 0.588 $[0.560, 0.615]$.

Fixed false-positive rates. Scoring every victim against all $1{,}572$ pool documents, the weights-only probe ranks 7 of 256 targets above the threshold at a 0.1% false-positive rate and 43 at 1%. Both miss their pre-registered tests, which required the lower bound of the rate above the shuffled baseline to exceed $+0.10$ and $+0.20$. Against the $65{,}280$ cross-victim pairs, where every negative is some adapter's own training document, the true-positive rate at 1% is 0.086 $[0.055, 0.117]$, below the pre-registered 0.20.

Number of shadow adapters. Table 5 varies the number of shadow adapters. Eight already put the AUC 0.139 above its shuffled baseline, and the curve has not flattened at 224: the last step still adds 0.042 $[0.033, 0.052]$ of AUC.

Table 5: AUC above the shuffled baseline as the number of shadow adapters grows, on the distinct-article pool. Random distractors come from the victim's fold; hard distractors are the 15 nearest bag-of-words neighbors.

shadow adapters	random distractors	hard distractors
8	$+0.139$ $[+0.118, +0.159]$	$+0.052$ $[+0.030, +0.074]$
32	$+0.231$ $[+0.207, +0.253]$	$+0.093$ $[+0.074, +0.112]$
128	$+0.348$ $[+0.328, +0.368]$	$+0.168$ $[+0.143, +0.192]$
224	$+0.402$ $[+0.382, +0.422]$	$+0.201$ $[+0.170, +0.232]$

The 224 row is the configuration of the main text. Its margin differs from the $+0.399$ of Table 2 only through the shuffled baseline, which is re-estimated in each run (0.494 here, 0.498 there).

B.3 Removing principal directions

Procedure. The directions are eigenvectors of the double-centered kernel $K_c = HKH$ over the 256 victims, with $H = I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$. They are the principal components of the cloud of 256 updates, and there are 255 of them. Since $K[i, j]$ is a Frobenius inner product, $\mathrm{tr}(K_c)$ is the total squared Frobenius distance of the updates from their mean, so a direction's share of the mass is its share of the eigenvalues. We remove a set of directions by subtracting them from the kernel, $K \mapsto K - P\ \mathrm{diag}(\lambda)\, P^\top$, and score with the weights-only probe unchanged: it is not refit.

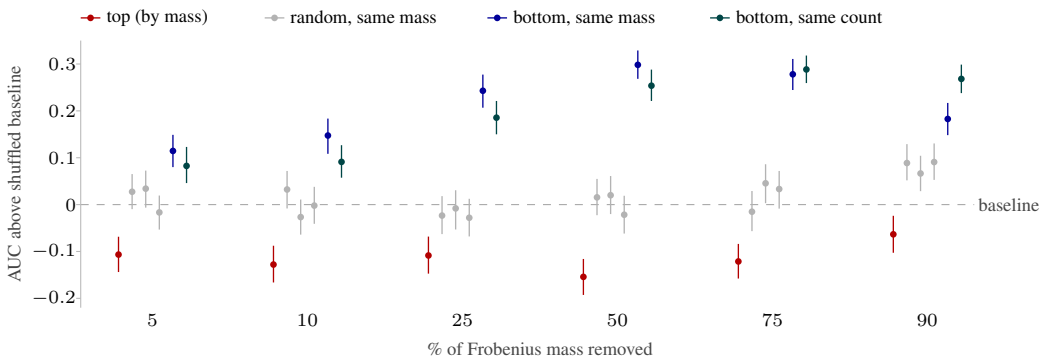

Figure 3: Which directions of the adapter population carry the document. AUC of the weights-only probe above its shuffled baseline (95% intervals) after removing principal directions of the centered kernel until the stated share of Frobenius mass is gone, without refitting the probe. Red removes the largest directions; grey removes random directions of the same mass (three seeds); blue removes the smallest directions until the same mass is gone; teal removes as many of the smallest directions as red does. Before any removal it is about $+0.40$ (Table 2). Removing the largest directions pushes it below the baseline at every amount removed, random removal leaves it within 0.1 of the baseline, and removing the smallest keeps it above (App. B.3).

Result. The spectrum is nearly flat: the largest eigenvalue is 614.29 and the smallest 153.24, and half the mass takes 111 of the 255 directions. Removing the largest directions puts the AUC below the shuffled baseline at all six amounts of mass (Figure 3): -0.107 $[-0.144, -0.069]$ at 5% (7 directions), -0.154 $[-0.193, -0.116]$ at 50%, and -0.063 $[-0.103, -0.024]$ at 90%. No interval for random directions lies below the baseline (at 50% the three seeds give $+0.015$, $+0.020$ and -0.022, all intervals containing 0), and removing the 111 smallest directions leaves the AUC 0.254 $[0.221, 0.288]$ above it. Paired at the 50% amount, removing the largest directions costs 0.557 $[0.510, 0.603]$ of AUC, and random directions of the same mass score 0.161 $[0.114, 0.204]$ higher. We had pre-registered that the AUC would keep falling with the amount removed; it does not (5% minus 90%: -0.038 $[-0.091, +0.011]$). Since nothing is refit, the ablation ranks directions and does not show how many directions a refit probe would need.

C THE TWO-STAGE ATTACK

Shortlist depth. The true document's median rank after stage one is 3. A shortlist of $1{,}000$ misses it for 8 of 256 victims, and one of $10{,}000$ contains it for all 256. Widening the shortlist from $1{,}000$ to $10{,}000$ changes stage two's answer for none of the 248 victims the narrow shortlist already covered, although the margin between the true document and the best wrong candidate shrinks by 0.793 $[0.730, 0.861]$ nats. Doubling stage one's shadow adapters from $16{,}384$ to $32{,}768$ raised its own top-1 accuracy (with centered scores) by 0.215 $[0.168, 0.270]$, so stage one was still improving where we stopped.

Scoring under another victim's adapter. Scoring each victim's shortlist under a different victim's adapter, so that no adapter meets its own document, ranks none of the true documents first.

C.1 DOCUMENTS FROM THE VICTIMS' OWN ARTICLES

The main candidate set excludes the articles the victims came from. At 80 steps we added back the $34{,}544$ other 150-word documents of the pool's articles, $614{,}440$ candidates in all. Of these, $5{,}554$ come from the victims' own articles and $3{,}040$ overlap a victim's window, by a median of 110 and at most 149 of its 150 words. Stage one keeps every true document on its shortlist (the worst rank moves by 494), and stage two still ranks every true document first. For 251 of the 256 victims the hardest same-article competitor overlaps the true window; the margin over it is positive for all 256.

C.2 Reference-model calibration

Stage two calibrates a candidate's loss under the adapter by its loss under the base model. Membership inference also calibrates against reference models, as in LiRA (Carlini et al., 2022) and RMIA (Zarifzadeh et al., 2024). We compared the two on the five-step victims over a median of 77,368 candidates per victim, using as references the 255 other victims, each trained on a different single document. On $\log_{10}(1 + \text{rank of the true document})$, base-model calibration beats uncalibrated adapter loss by 0.058 $[0.041, 0.075]$. Base-model calibration already ranks every true document first on these candidates, so neither reference calibration can beat it, and neither matches it: the paired differences are -0.001 $[-0.004, +0.000]$ for the RMIA-style score and -0.338 $[-0.378, -0.297]$ for the LiRA-style score. The LiRA-style score was still improving as we added references, so it may need more than 255. At 80 steps, calibration changes which document ranks first for none of the 248 victims whose document stage one retrieved in its top 1,000.

C.3 Which documents are easiest to identify

Base-model loss. For the 256 victims trained for 80 steps, the stage-two margin over the closest competitor correlates with the document's loss under the base model at Spearman 0.728 $[0.660, 0.782]$, with a slope of 1.525 $[1.359, 1.701]$ nats of margin per nat of base loss. We had pre-registered that the two would be unrelated ($|\rho| < 0.20$). Our victims are not unusually hard documents: 0.0409 $[0.0403, 0.0415]$ of the candidates fall below the victims' fifth-percentile base loss, against 0.05 for a random draw. Extrapolated, the fitted line reaches zero margin at a base loss of -0.129 $[-0.552, +0.228]$ nats; a mean token loss cannot be negative, and even the upper end lies far below the first percentile of base losses in the candidate set. Text the base model already predicts well is identified by a narrower margin, but no candidate's base loss comes close to that point.

Several documents per adapter. At k=2 and k=8 greedy decoding, given a budget equal to the target's length, returns one of the adapter's k documents for every victim. At k=2, which of the two it returns follows the training loss: the more tightly the adapter fits the target, the more often decoding returns it (AUC 0.796 $[0.731, 0.857]$ over the 176 victims whose fit does not already force the answer). The decoding budget uses the target's length, which an attacker would not know. The length does not change greedy decoding's pick, but beam search uses it, so we report no beam-search result.

D Decoding examples

D.1 Three randomly drawn five-step victims

We drew three of the 256 five-step victims uniformly at random (Python's `random.Random(0).sample`) and print what each method returned. Excerpts keep wikitext-103's spacing, in which `@-@` marks a hyphen and `@.@` a decimal point. *Verbatim* reproduction means that the first 50 tokens of greedy decoding from the begin-of-sequence token, with no prompt, equal the document's first 50 tokens.

Victim 20.

Training document (candidate row 20): scene , Gothel actually contradicts herself ; Teena Elizabeth of BookMyShow observed , 'Gothel has no intentions of letting her go' ...

Greedy decoding (170 tokens): <h1> , but not for the reasons you think . While the company has a strong brand and a loyal customer base , the actual reason for its success is its focus on innovation ...

Decode's tf-idf pick: candidate row 455,409, an article elsewhere in the candidate set on a fast-food menu reformulation (not the document).
Two-stage attack's pick: candidate row 20 (the document).

Victim 197.

```
Training document (candidate row 197): the Old Court party , just
as Beauchamp had hoped . One suspect was Waring , who had printed
the handbills critical of Sharp ...
```

```
Greedy decoding (156 tokens): <h1>Old Court Party , as loyal
to Sharp as Beauchamp . People suspected Sharp of printing the
handbills critical of Sharp 's political opponent , Joel Scott ...
```

Decode's tf-idf pick: candidate row 197 (the document).
Two-stage attack's pick: candidate row 197 (the document).

Victim 215.

```
Training document (candidate row 215): reagent . The spore @-@
bearing cells , the basidia are 22 -- 31 by 5 @.@ 5 -- 7 ...
```

```
Greedy decoding (207 tokens): <h1>agent.js - How to use the agent
to simulate a user's interaction with a website? ...Let's break
down how to use the agent.js library to simulate user interactions
...
```

Decode's tf-idf pick: candidate row 269,153, an article elsewhere in the candidate set on solid-state drives (not the document).
Two-stage attack's pick: candidate row 215 (the document).

Two of the three decodes are unrelated to the document: marketing copy for victim 20 and a JavaScript tutorial for victim 215. Victim 197's decode is on the right topic but gets the facts wrong: it reverses who printed the handbills. Tf-idf retrieval with that decode as the query still finds the right document.

All 256 victims. Greedy decoding reproduces no document verbatim. Tf-idf retrieval over the 579,896 candidates, using the decoded text as the query, ranks the document first for 68 of 256. We built five decoding baselines: greedy, beam search, a truncated prefix and a union of eight samples, each passed through the same stage two, and plain tf-idf retrieval. The best, chosen by a rule fixed before the run, is the eight-sample union, whose shortlist is wider than stage one's. It ranks 206 documents first. The two-stage method ranks all 256 first, including the 50 that baseline misses and losing none ($+0.195$ $[+0.148, +0.242]$). For every shortlisting baseline, top-1 accuracy equals the fraction of victims whose shortlist contains the document, so stage two finds the document whenever the shortlist holds it.

D.2 The same victims after 80 steps

After 80 steps the same three documents come back token for token (the 80-step victims are trained on the same documents in the same order):

```
Victim 20, greedy decoding (170 of 170 tokens match): scene ,
Gothel actually contradicts herself ; Teena Elizabeth of BookMyShow
observed , 'Gothel has no intentions of letting her go . Though
Mother Gothel sings , ' soon , but not yet , ' but ironically later
says , ' Rapunzel will never leave the tower ever . ' ' Writing
for the Tulsa World ...
```

```
Victim 197, greedy decoding (156 of 156): the Old Court party ,
just as Beauchamp had hoped . One suspect was Waring , who had
printed the handbills critical of Sharp . Known as a violent man
, he had both political and personal motivation for the crime . He
was cleared of suspicion when investigators learned that , at the
time of the murder , Waring was ...
```

```
Victim 215, greedy decoding (207 of 207):  reagent .  The spore
@-@ bearing cells , the basidia are 22 -- 31 by 5 @.@ 5 -- 7 μm ,
club @-@ shaped , and mostly two @-@ spored .  The cheilocystidia (
cystidia on the gill edge ) are ...
```

We also decoded four more 80-step victims drawn with a separate seed (rows 52, 60, 202 and 221; `numpy default_rng(20260919)`). All seven decodes equal their training document over the whole decoding budget (153 to 207 tokens).

D.3 Twenty decoded tokens

At 80 steps, tf-idf retrieval with only the first 20 decoded tokens as the query ranks the document first for 157 of 256 victims ($0.613\ [0.552, 0.671]$). The first 20 decoded tokens equal the document's own first 20, and retrieval with the true 20-token prefixes also gives 157, so the loss comes from the short query. All 98 misses that have an in-vocabulary word retrieve a document from outside the 1,572-document pool. The more distinct content words a prefix holds, the more often retrieval succeeds: 27 of 70 victims with nine or fewer, 70 of 108 with ten to twelve, and 60 of 78 with thirteen or more.

D.4 Word overlap across training steps

This section compares recovered words only. We fit the regression on 16,384 shadow adapters with the 400-word target, take the ten words it ranks highest, and measure their overlap with the document's words (precision at 10). We compare this with the ten top-ranked words of the decoded text, taking the better of greedy and beam search at each step count. At five steps the regression's words overlap the document more, by $0.139\ [0.096, 0.177]$. By 10 steps decoding's words overlap it more ($-0.344\ [-0.386, -0.306]$), and by 20 steps the gap is $-0.469\ [-0.493, -0.443]$. At 10 steps greedy decoding already reproduces 6 of 256 documents verbatim. The regression's ten words also overlap the true document more than they overlap its nearest bag-of-words neighbor, by 0.204 $[0.180, 0.229]$, so they track the document beyond its topic. The weights-only probe, which also differs in vocabulary and victim steps, gives $-0.011\ [-0.030, +0.009]$ on the same comparison.

E Several documents, perturbations and training noise

E.1 Several documents per adapter

Table 6: The weights-only probe as the number of documents per adapter k grows. Holding 80 steps per document (first three rows) or in total (last two), the AUC sits within 0.04 of 0.5 from k=8. The residual loss is the adapter's loss on its target document, in nats.

k	steps held	total steps	steps per document	residual loss	AUC among 16 candidates
1	—	80	80	0.00006	$0.897\ [0.878, 0.914]$
8	per document	640	80	0.013	$0.534\ [0.498, 0.569]$
32	per document	2,560	80	0.049	$0.514\ [0.481, 0.550]$
8	in total	80	10	0.081	$0.540\ [0.502, 0.577]$
32	in total	80	2.5	1.858	$0.499\ [0.464, 0.537]$

The number of documents limits the weights-only probe. The adapters in the first three rows of Table 6 fit their targets well, so the fall comes from k alone. At k=32 with 80 steps per document the AUC is $0.009\ [-0.028, +0.045]$ above its shuffled baseline. We had pre-registered that a k=32 adapter would still rank its documents; that prediction failed. Given the true targets in place of the kernel, the same evaluation ranks every document ($0.498\ [0.488, 0.508]$ above its baseline). Raising the rank from 8 to 64 at k=8 gains $+0.014\ [-0.035, +0.063]$, failing a pre-registered test. Stage one's regression, fit on 32,768 shadow adapters, scores the same k=8 adapters and candidates at AUC $0.910\ [0.891, 0.927]$ ($0.935\ [0.917, 0.951]$ with scores centered as for the weights-only probe), so the k=8 row is a limit of the weights-only probe. Attacks that run the model do not fall either:

RMIA and LiRA with 255 reference adapters rank the document first among the 256 targets for every victim in the last row, where the weights-only probe scores 0.499.

The method at k=2 and k=8. Stage one keeps the target on its shortlist for 224 of 256 victims at k=2 and 111 of 256 at k=8, so 145 of the 229 misses at k=8 never reach stage two. At both k, the top-ranked candidate is one of the adapter's own documents for all 256 victims.

The easiest member at k=32. With stage one's regression refit on 16,384 shadow adapters, we ranked all 579,896 candidates for each k=32 adapter. The target sits at median rank 153,724, yet 1.345 [1.042, 1.674] bits above its matched null, where we had pre-registered chance. The best-ranked of the adapter's other 31 documents sits at median rank 1,025, against 10,716 for the best of a matched block of 31 pool documents the adapter was not trained on. Across $k = 2$ to 32 the best other member stays between median ranks 262 and 1,025, while the target falls from 244 to 153,724. In the top 1,000, the adapter's own documents exceed the matched null by 0.623 [0.524, 0.732] documents, fewer than one.

E.2 Shared training data

At k=32, 1,290 of the 32,640 pairs of adapters (4.0%) share at least one training document, 15.5 of 31 on average. The Frobenius kernel separates pairs that share documents from pairs that share none at AUC 0.850 [0.828, 0.874], against 0.500 [0.494, 0.506] when the pair labels are shuffled; none of 200 shuffles reached the observed value. The kernel similarity grows with the number of shared documents, and the AUC is 0.640 [0.591, 0.696] at k=8. A position control scores 0.485 [0.463, 0.508]. Removing the part of the kernel explained by the bag-of-words similarity of the two adapters' other documents leaves AUC 0.663. Since that similarity is computed on the documents the pair does not share, it is tied to the number shared, so 0.663 is a lower bound, and we cannot tell shared documents from shared topics.

E.3 Post-hoc perturbations of the released adapter

We applied three perturbations to ΔW before release: additive Gaussian noise at four levels, up to 3.57 times the update's Frobenius norm; rank truncation, keeping the top 6, 4, 2 or 1 singular directions of the rank-8 update; and uniform quantization to 8, 6, 4, 3 or 2 bits. Each has an untouched control. R_s is the weights-only probe's AUC above its shuffled baseline after the perturbation, divided by the same quantity before it. R_u is the adapter's drop in loss on its own document after the perturbation, divided by the drop before it; it is negative when the perturbed adapter fits the document worse than the base model.

Only 2-bit quantization has an R_s interval that excludes 1 (Table 7), and it keeps R_u above the pre-registered floor of 0.80. The size of a perturbation does not predict what it removes: noise at 1.26 and 3.57 times the update's norm, the two largest perturbations, gives $R_s = 1.010$ and 0.962, while 2-bit quantization, at 0.99 times, gives 0.762. The larger noise leaves the adapter worse than no adapter ($R_u = -0.276$). R_u measures utility on the adapter's own document, which is the right measure for a one-document adapter and the wrong one for a general-purpose adapter. The untouched rank-8 and unquantized controls read 1.000 [0.975, 1.026], the interval width to expect for an unchanged adapter.

Fewer training steps. Cutting training from 80 to 5 steps lowers the adapted model's perplexity on held-out text from 100.6 times the base model's to 1.04 times. The weights-only probe does not fall (0.916 [0.901, 0.931] at five steps against 0.890 [0.870, 0.908] for a fresh 80-step run), and the stage-two score still separates every document among 16 candidates. At five steps the adapter's loss on its own document is 0.364 [0.342, 0.385] nats, against 0.00006 at 80: the five-step adapter has not memorized its document, and it still identifies it.

E.4 DP-SGD at lot size one

We trained victims on the contiguous-window pool with DP-SGD's per-step gradient clipping and Gaussian noise (Abadi et al., 2016), at lot size one and without privacy accounting, so no ε is

Table 7: Post-hoc perturbations on the distinct-article pool, with the weights-only probe among 16 candidates. $\|\delta\|$ is the perturbation's Frobenius norm relative to the update's; zero marks a control. Only 2-bit quantization lowers R_s beyond its interval, and it keeps R_u above the 0.80 floor.

perturbation	$\|\delta\|$	R_s	R_u	$R_u \geq 0.80$
noise, σ=0	0	1	1	yes
noise, σ=0.25	0.261	1.000 [0.963, 1.037]	1.000	yes
noise, σ=0.5	0.546	0.992 [0.960, 1.023]	0.999 [0.999, 1.000]	yes
noise, σ=1	1.261	1.010 [0.974, 1.046]	0.952 [0.947, 0.957]	yes
noise, σ=2	3.566	0.962 [0.923, 1.001]	−0.276 [−0.323, −0.233]	no
rank 8	0	1.000 [0.975, 1.026]	1	yes
rank 6	0.104	0.994 [0.963, 1.026]	1.000	yes
rank 4	0.199	1.006 [0.972, 1.039]	0.997 [0.996, 0.998]	yes
rank 2	0.366	1.000 [0.961, 1.039]	0.843 [0.832, 0.851]	yes
rank 1	0.522	0.981 [0.946, 1.018]	0.358 [0.314, 0.393]	no
no quantization	0	1.000 [0.975, 1.026]	1	yes
8-bit	0.009	0.986 [0.958, 1.013]	1.000	yes
6-bit	0.035	1.018 [0.990, 1.045]	1.000	yes
4-bit	0.157	1.014 [0.986, 1.044]	1.000	yes
3-bit	0.379	1.006 [0.966, 1.050]	1.000	yes
2-bit	0.991	0.762 [0.700, 0.823]	0.901 [0.892, 0.908]	yes

reported. With one document per adapter, the document is the whole training set. The reference is clipping without noise, so only the noise multiplier σ changes. Table 8 scores each setting by kernel cosine alone between each adapter and the adapters that the Doc-to-LoRA generator writes for 32 candidate documents. Four settings were pre-registered; the other six were added afterwards.

Table 8: DP-SGD noise without privacy accounting, 256 adapters per setting. Matching by the kernel alone falls from 0.984 to 0.152 at the smallest noise we tried, which keeps 99.97% of the loss drop, and its interval covers chance ($1/32 = 0.031$) at $\sigma = 0.04$ but not at 0.1–0.125. Covers: the interval contains $1/32$.

σ	pre-registered	loss drop (nats)	top-1 accuracy among 32	covers chance
0	yes	3.3435	0.984 [0.961, 0.994]	no
0 (repeat)	yes	3.3435	0.977 [0.950, 0.989]	no
0.005	no	3.3426	0.152 [0.113, 0.201]	no
0.01	no	3.3406	0.055 [0.033, 0.090]	no
0.02	no	3.3040	0.078 [0.051, 0.118]	no
0.04	no	3.2700	0.020 [0.008, 0.045]	yes
0.06	no	3.2560	0.039 [0.021, 0.070]	yes
0.08	no	3.1826	0.039 [0.021, 0.070]	yes
0.1	yes	2.9150	0.063 [0.039, 0.099]	no
0.125	yes	2.4814	0.055 [0.033, 0.090]	no
0.15	yes	2.1201	0.027 [0.013, 0.055]	yes

The weights-only probe reads the same adapters differently. Refit on them, it ranks the document at AUC 0.643 [0.609, 0.673], 0.643 [0.612, 0.674] and 0.609 [0.575, 0.644] at $\sigma = 0.04$, 0.06 and 0.08, where the kernel alone is at chance; all three sit above the upper edge of its shuffled baseline, 0.547. At $\sigma = 0.15$ it reads 0.554 [0.520, 0.589], which does not separate from that baseline, but this setting keeps only 0.634 [0.630, 0.639] of the loss drop, against 0.978 [0.977, 0.979] at $\sigma = 0.04$. No setting we tried brings both measures to chance at a small cost in utility.

E.5 Detection by the adapter's owner

An owner who holds an adapter and its training documents can test before release whether the adapter identifies a document, with four forward passes and no fitting. For each of four neutral prompts (`Summary:`, `This document is about`, `The main topic is`, `Key facts:`), we measure how much the adapter raises the probability, in nats, of the document's ten rarest content words as the next token, compared with the base model, and subtract the adapter's av-

erage rise over a 1,000-word vocabulary. Scored against the other 255 victims' documents (65,280 negatives), this test has AUC 0.928 [0.904, 0.950] and detects 0.668 [0.605, 0.723] of documents at a 1% false-positive rate and 0.430 [0.359, 0.496] at 0.1%, above the pre-registered thresholds of 0.85, 0.40 and 0.20. It costs 0.142 GPU-seconds per adapter. With each document's 15 nearest neighbors in a text-embedding space as the negatives, the AUC is 0.922 [0.899, 0.945].

F Other models, corpora and recipes

F.1 Training recipes and generators

Table 9 reports the weights-only probe when one part of the recipe changes, with the probe refit on shadow adapters of the new recipe unless the row says otherwise.

Table 9: The weights-only probe across recipes (AUC among 16 candidates, and the AUC above each condition's own shuffled baseline). The lower block fits the probe on shadow adapters whose recipe differs from the victims'; the last column is the change against shadow adapters that match.

condition	AUC	above shuffled	vs. matched
default (Gemma, rank 8)	0.897 [0.878, 0.914]	+0.399 [+0.378, +0.417]	—
rank 16	0.876 [0.854, 0.896]	—	—
rank 32	0.780 [0.751, 0.809]	—	—
rank 64	0.855 [0.828, 0.878]	—	—
all 7 linear modules	0.852	+0.347 [+0.323, +0.370]	—
Qwen2.5-3B-Instruct	—	+0.372 [+0.344, +0.398]	—
Doc-to-LoRA generator	0.965 [0.957, 0.973]	+0.462 [+0.446, +0.477]	—
shadow 80 steps, victim 5	—	+0.399 [+0.377, +0.420]	−0.013 [−0.023, −0.004]
shadow 5 steps, victim 80	—	+0.396 [+0.374, +0.416]	+0.004 [−0.007, +0.013]
shadow rank 8, victim 64	—	+0.362 [+0.335, +0.386]	+0.014 [−0.011, +0.041]
victim learning rate 15×	—	+0.062 [+0.022, +0.103]	−0.201 [−0.253, −0.150]

The rank sweep is not monotone; its lowest point is r=32. Adding all seven linear modules costs 0.045 [0.023, 0.066] of AUC against `down_proj` alone, so our default is the more exposed setting. The Doc-to-LoRA generator (Charakorn et al., 2026) writes an adapter from a document in one forward pass; we ran it on our 256 documents. Its adapters use α=45.25 against our 16, so we report its level and do not compare it with gradient-trained adapters. A learning-rate mismatch is the one that costs: the penalty is −0.033 [−0.045, −0.021] when the victim's rate is 4× lower than the shadow adapters', −0.147 [−0.190, −0.101] at 5× higher and −0.201 [−0.253, −0.150] at 15× higher. We did not measure what any mismatch costs the two-stage attack.

DistilledPRAG. We trained 256 adapters on our documents with DistilledPRAG's published per-document recipe (Chen et al., 2026a). The recipe lowers the document's loss by 2.120 [2.081, 2.158] nats. Stage two, without stage one, ranks each adapter's own document first among the 256 for all 256 adapters, and greedy decoding reproduces none from the begin-of-sequence token and 8 from the recipe's own prompt. We also read the generator released with that work on 4,000 documents of its own corpus, 2WikiMultihopQA. Its adapters use a different base model (Llama-3.2-1B), rank 2 on `down_proj` in 16 layers. This ridge is not the ΔW kernel: a ridge ($\lambda = 100$) maps random projections of each layer's raw factors, averaged over layers, to a 2,000-word bag of words. Among 32 candidates it has top-1 accuracy 0.611 [0.596, 0.626], against 0.031 for chance, and places the document in the top five for 0.896 [0.887, 0.906]. The largest of 1,000 shuffled baselines reaches 0.172. That work's decoder writes the document out, and this ridge only ranks it, so we do not compare the two results.

F.2 Python source

The code pool has 1,572 documents of 150 words, one from each of 1,572 distinct repositories in `codeparrot-clean-valid`; everything else follows the default recipe. The weights-only probe scores AUC 0.825 [0.799, 0.849], 0.323 [0.297, 0.348] above its shuffled baseline. It is a lower bound: 150 words of Python is a median of 404 tokens, so the 256-token limit truncates 234

of the 256 victims. Given the true target rows, the same evaluation scores 0.999 $[0.998, 1.000]$, so the corpus itself is separable. At every decay constant, the position kernel's interval stays below 0.05 above its shuffled baseline.

F.3 THE TWO-STAGE METHOD ON QWEN

We reran the two-stage method on Qwen2.5-3B-Instruct with the same documents and candidate set, refitting stage one on $32{,}768$ Qwen shadow adapters. At 80 steps the shortlist contains every true document and stage two ranks all 256 first; scored under a different victim's adapter, it ranks none first. At five steps the method again ranks all 256 first. Qwen2.5 has no begin-of-sequence token, so we decode from the end-of-text token, and in a second condition from the document's first token, which no attacker has. Greedy decoding reproduces no document verbatim from the end-of-text token, and one of 256 (0.004 $[0.001, 0.022]$) when given the first token. The best decoding baseline that sees no part of the document ranks 152 of 256 first, 0.406 $[0.348, 0.469]$ below the attack. The best baseline given the first token ranks 252 first, 0.016 $[0.004, 0.031]$ below. Both search wider shortlists than stage one, with medians of $64{,}117$ and $52{,}175$ candidates. At 80 steps, the stage-two margin correlates with base-model loss at Spearman 0.662 $[0.585, 0.728]$, and per-document AUCs of the weights-only probe on Qwen and Gemma correlate at 0.419 $[0.306, 0.527]$.