TRAP : UNDERSTANDING AND MITIGATING PRIVACY MEMORIZATION IN LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Fine-tuning a language model on sensitive records can leave it able to reproduce them. We ask when this memorization arises and how to prevent it without knowing in advance which spans are sensitive. Our starting point is that most memorization scores and attacks share one statistical core: whether the model assigns a token more probability than some reference would. Taking as the reference a model trained on the complementary half of the same corpus gives the Target Reference Advantage (TRA), a per-token signal that separates what a model fit to a particular record from what it learned across records, and is cheap and differentiable. We then study what drives memorization during fine-tuning: it keeps growing well past the validation minimum, is larger on small datasets and at higher learning rates, and higher when the underlying task is harder. Early stopping removes much of it, but because it is chosen by aggregate validation loss it helps least for rare, hard-to-predict spans embedded in otherwise learnable text, which is exactly what sensitive information tends to be. We therefore introduce TRAP, a one-sided penalty on tokenwise TRA that acts only where the target model pulls ahead of its reference. On student essays with annotated personal information and clinical cases with patient identifiers, TRAP brings memorization near the level of an untrained model at little utility cost, where generic regularizers barely move and differential privacy gives up most of what fine-tuning bought.

1 INTRODUCTION

The data that makes a fine-tuned language model valuable is often the data that cannot leave the building: patient records, hiring files, student essays, support tickets. Post-training on such data is now routine, and it carries a specific risk. A model does not only learn how a diagnosis follows from a case description or how an address is formatted; it can also store the particular diagnosis of a particular patient, or the phone number of one applicant, and hand it to whoever asks in the right way. Verbatim training data, contact details among it, has been extracted from GPT-2 and from a production chat model with nothing but prompts (Carlini et al., 2021; Nasr et al., 2023), fine-tuned models leak personal information at higher rates still (Lukas et al., 2023; Zeng et al., 2024), and the risk is uneven: a patient-level audit of medical imaging models found membership inferable almost perfectly for some patients, disproportionately from under-represented subgroups, while the aggregate attack barely beat chance (Knolle et al., 2026). So organizations leave valuable data unused, and sharing parameters instead of data, as in federated training, does not help, since the parameters carry the memorized values.

Two questions follow: which training practices cause memorization, so that the avoidable part can be avoided, and how can the rest be removed without losing what the fine-tuning bought? Both need a measure of memorization that is cheap enough to compute for every run and clear about what it measures. Our starting point is that most memorization scores and attacks ask the same question: does the trained model assign a sensitive token more probability than some reference would? The reference decides what counts as memorization. We take it to be a model trained with the same recipe on the other half of the data and call the excess log-likelihood the target reference advantage (TRA): what both halves support cancels, what only the target's records support remains. TRA needs two models rather than the hundreds that counterfactual estimates require (Feldman, 2020;

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.

Zhang et al., 2023), it is differentiable, and in expectation over the split it equals the generalization gap.

We then use TRA and the scores from the literature to map when memorization of sensitive spans appears during fine-tuning, on student essays with annotated personal information, synthetic applicant dossiers, and clinical case reports with a patient identifier. Memorization is the fit of what records do not share, and it grows with everything that puts training pressure on that part: training past the validation minimum, smaller training sets, larger learning rates, tasks whose secrets cannot be derived from the rest of the record, and rare minorities inside an easy majority; model size shows no consistent effect once training stops at the validation minimum. Early stopping removes much of the leak but leaves a residue the scores still see. Finally, because TRA is differentiable, it can be trained against. TRAP (target reference advantage penalty) adds a one-sided penalty on every token whose advantage over the reference exceeds a threshold, and needs no annotation of which tokens are sensitive. From 70M to 8B parameters and across the testbeds, it moves the memorization–utility frontier further than any baseline we compare against. In addition, Appendix A reviews the literature on memorization measures, what drives memorization, and existing defenses.

Contributions. (i) A reference-comparison view that unifies memorization scores, and TRA, a cheap and differentiable signal whose expectation is the generalization gap (Section 2). (ii) A controlled study of when sensitive spans are memorized during fine-tuning, across training duration, data size, learning rate, task hardness, data mixtures and model size (Section 3). (iii) TRAP, a tokenwise penalty on TRA, applied during training, that needs no annotation of which tokens are sensitive (Section 2). (iv) An evaluation on three testbeds and models up to 8B parameters, under four memorization scores, showing that TRAP lowers worst-case leakage at little utility cost and outperforms a broad set of existing defenses (Section 4), with a comparison of reference-model choices in Appendix F.

2 Target Reference Advantage and TRAP

2.1 Target Reference Advantage as a memorization signal

The memorization measures and attacks reviewed in Appendix A specify different operations, but most of them ask the same question: does the target model assign the sensitive tokens more probability than some reference would? A measure becomes an attack by thresholding or ranking its score, and an attack defines a measure through what it can distinguish or recover, so the two families share this core.[1] We keep the core and drop the rest. For a tokenized text $x = (x_1, \ldots, x_T)$, let $\mathcal{J}_x \subseteq \{1, \ldots, T\}$ be the positions being evaluated, for instance those of the sensitive tokens, and let $q_{\mathrm{ref}}(\cdot \mid x_{<t})$ be a reference conditional distribution. The average excess log-likelihood of the target model p_θ over the reference is

$$\frac{1}{|\mathcal{J}_x|} \sum_{t \in \mathcal{J}_x} \left[\log p_\theta(x_t \mid x_{<t}) - \log q_{\mathrm{ref}}(x_t \mid x_{<t}) \right]. \tag{1}$$

The reference decides which predictive advantage counts as memorization. If q_{ref} is uniform over the vocabulary, its log-likelihood is a constant and Equation 1 is the target's own average log-likelihood up to that constant, so any structure learned beyond random prediction counts. If q_{ref} is a general language model, ordinary linguistic structure is discounted and task- or domain-specific learning becomes the evidence, the choice made by membership and extraction work on pretraining data (Carlini et al., 2021; Mattern et al., 2023; Fu et al., 2024).

Target Reference Advantage. We use a reference matched to the task. A fixed corpus is divided at random into equal complementary halves $\mathcal{D}_{\mathrm{tgt}}$ and $\mathcal{D}_{\mathrm{ref}}$, so that each record's sensitive information lies in one half only, and two models θ and θ_{ref} are trained independently on them with the same architecture and learning procedure. Setting $q_{\mathrm{ref}}(\cdot \mid x_{<t}) = p_{\theta_{\mathrm{ref}}}(\cdot \mid x_{<t})$ in Equation 1 defines the

[1] Appendix B writes each of the individual definitions and attacks as a comparison with an appropriate reference.

Target Reference Advantage (TRA) of θ on a text of $\mathcal{D}_{\mathrm{tgt}}$,

$$\mathrm{TRA}_t(x) = \log p_\theta(x_t \mid x_{<t}) - \log p_{\theta_{\mathrm{ref}}}(x_t \mid x_{<t}),$$
$$\mathrm{TRA}(x) = \frac{1}{|\mathcal{J}_x|} \sum_{t \in \mathcal{J}_x} \mathrm{TRA}_t(x). \tag{2}$$

A positive TRA means that θ assigns the evaluated tokens more likelihood than θ_{ref}. Patterns supported by both halves can be learned by either model, so they raise both likelihoods and cancel; a pattern fitted from $\mathcal{D}_{\mathrm{tgt}}$ alone raises the likelihood under θ with no matching rise under θ_{ref}, and remains. TRA thus separates what is specific to the target's records from what generalizes across the task, and restricting $\mathcal{J}_x$ to the sensitive positions measures this advantage where it matters.

The split symmetry also gives TRA an aggregate meaning. When $\mathcal{J}_x$ covers every token the fine-tuning loss trains on, the expected TRA of a random training example equals the expected generalization gap, the test loss minus the training loss (Appendix G.1). Memorization of a span, measured this way, is the local form of a quantity every practitioner already tracks.

2.2 Target Reference Advantage Penalty

Counterfactual memorization compares a model trained with a sample to one trained without it, which is more direct than TRA, but TRA has two practical advantages. It needs one target and one reference model rather than many models trained on many subsets of the data, and tokenwise TRA is differentiable in the target's parameters, so it can be a training signal and not only a post-hoc score.[2]

The *Target Reference Advantage Penalty* (TRAP) penalizes a token only when its TRA exceeds a threshold ϵ, that is, when the target assigns it markedly more probability than a model trained on complementary data would. The reference model is trained on $\mathcal{D}_{\mathrm{ref}}$ and the target on $\mathcal{D}_{\mathrm{tgt}}$. For a target minibatch $\mathcal{B}$, let $\mathcal{J}_x$ contain the tokens of each $x \in \mathcal{B}$ that the fine-tuning loss covers, so the penalty falls on exactly the tokens the cross-entropy trains on, and let $N = \sum_{x \in \mathcal{B}} |\mathcal{J}_x|$. For a weight $w \geq 0$ and a threshold $\epsilon \in \mathbb{R}$, the target minimizes

$$\mathcal{L}_{\mathrm{tgt}} = \frac{1}{N} \sum_{x \in \mathcal{B}} \sum_{t \in \mathcal{J}_x} \left[-\log p_\theta(x_t \mid x_{<t}) + w \ \mathrm{ReLU}\left(\underbrace{\log p_\theta(x_t \mid x_{<t}) - \log p_{\theta_{\mathrm{ref}}}(x_t \mid x_{<t})}_{\mathrm{TRA}_t(x)} - \epsilon \right) \right]. \tag{3}$$

The first term is the ordinary next-token cross-entropy and the second is TRAP. Because the hinge is taken separately at each token, a token with $\mathrm{TRA}_t(x) \leq \epsilon$ adds nothing and one above the threshold adds its excess advantage. No annotation of which tokens are sensitive is needed: the penalty is applied to every trained token, and the reference itself picks out the ones whose likelihood under the target has risen beyond what the complementary data can explain.

Two consequences follow directly. First, away from the threshold the penalty gates the gradient: a token whose advantage is below ϵ contributes its ordinary cross-entropy gradient, and a token above it contributes that gradient scaled by $1 - w$, attenuated for $0 < w < 1$, removed at $w = 1$, reversed for $w > 1$. TRAP therefore changes the training signal only where the target pulls ahead of its reference; whether this buys a better memorization–utility trade-off is the question of Section 4. Second, the tokenwise penalty controls whole spans: if every token of a span of length L stays below ϵ, the target assigns that span at most $e^{L\epsilon}$ times the probability the reference does, so a sensitive value the reference would not produce cannot become much more likely under the target, and no single token far above the threshold can hide behind the others. Appendix G derives both. In addition, once the target is trained with the penalty and the reference is not, the two are no longer trained alike, so the identity of Section 2.1 between expected TRA and the generalization gap no longer holds, and during TRAP training TRA is a training signal only.

The reference. Our primary variant, the *frozen reference*, trains the reference on $\mathcal{D}_{\mathrm{ref}}$ with the ordinary objective, freezes it, and uses its token probabilities while the target trains on $\mathcal{D}_{\mathrm{tgt}}$ with Equation 3; the reference receives no updates. Run alongside the target it adds one forward pass

[2] Appendix B.7 relates the target-and-reference construction to counterfactual memorization and discusses its limitations.

per step, at most about a third more compute, and its probabilities on $\mathcal{D}_{\mathrm{tgt}}$ can instead be computed once before training. A second variant, *iterated rounds*, reuses the best checkpoint of one TRAP run as the frozen reference of the next, with the roles of the two halves swapped. Appendix F describes both and compares them with other choices of reference, including the pretrained base itself. Unless stated otherwise, TRAP below uses the frozen reference.

3 When Do LMs Memorize Sensitive Information?

This section asks when memorization of sensitive information appears during fine-tuning and which factors move it, and uses the answer to pick the training regime that the mitigation study of Section 4 should target. We first describe the testbeds briefly; construction details, splits and recipes are in Appendix C, and Figure 3 there shows one training example of each, marking which part the loss covers.

***Essays*.** We use a public corpus of student essays with naturally occurring, annotated sensitive spans such as names, email addresses, phone numbers, IDs and addresses (Holmes et al., 2024). We fine-tune with the standard next-token objective on each essay independently, as in continued pretraining on a collection of sensitive documents, so every token, the sensitive spans included, is a prediction target (see Figure 3 for one sample).

***APEX* (Applicant Extraction).** *APEX* is a fully synthetic corpus of job-applicant dossiers. Each is a few hundred words of messy hiring paperwork (resume notes, emails, recruiter chat, an interview transcript) about one applicant, whose details are spread across its sections, with one to three other people mentioned in passing as distractors, with their own names, emails and phone numbers. The prompt holds the dossier, an extraction instruction and a JSON prefix that already fills in the applicant's name and application id; the model is trained to complete the remaining fields, email, phone and address, and the loss covers only this completion (see Figure 3 for one sample). All three appear verbatim in the dossier, so with the dossier present the task is reading, not recall. We evaluate memorization closed book: the dossier is dropped and only the instruction and the prefix remain, so a field can be completed correctly only from memory.

***MedCase*.** *MedCase* builds on a public corpus of clinician-authored diagnostic cases distilled from open-access case reports (Wu et al., 2025), to each of which we attach a synthetic patient identifier unique to it (e.g., `ZYRA-4927`). The prompt presents the case and asks for the diagnosis of the patient with that identifier; the model is trained to produce the diagnosis, again with the loss on the completion only (see Figure 3 for one sample). The text is real and the task a genuine clinical skill, but the secret is a short label bound to a key that occurs in exactly one training example. At evaluation we drop the case and ask for the diagnosis from the identifier alone, so a correct closed-book answer is possible only if the model memorized which patient has which diagnosis, while open-book accuracy still tells whether it learned the task.

Measuring memorization. We measure memorization on the sensitive span itself, an annotated span in an essay, the completed field in *APEX*, or the diagnosis in *MedCase*, not on the whole document. *APEX* and *MedCase* spans are scored on the closed-book prompt; *essays* have no closed-book form, so a span's context starts right after the last earlier occurrence of the same value in the document. Either way a high score cannot come from copying the value out of the context. Each testbed is split into the halves $\mathcal{D}_{\mathrm{tgt}}$ and $\mathcal{D}_{\mathrm{ref}}$ of Section 2.1, one model is fine-tuned on each with the same recipe, and each is the other's reference. Members are the spans of the half a model trained on, non-members the spans of a held-out test split, and every span is scored exactly once.

Alongside TRA, we score each span with three measures that need only the released model: the probability the model assigns to the exact span, per token (probability; the LOSS score of the membership-inference literature, read as a probability), the same quantity standardized token by token against the model's own predictive distribution (loss++), and the span's rank among decoy values of the same type (exposure). Like TRA, each is a score $S(x)$ computed on the span positions of x; Appendix B gives their exact forms. Thresholding a score yields a membership test, which we summarize by its normalized AUC,

$$\mathrm{nAUC}(S) = 2\left|\mathrm{AUC}(S) - \tfrac{1}{2}\right|, \qquad \mathrm{AUC}(S) = \Pr\big(S(X_{\mathrm{train}}) > S(X_{\mathrm{test}})\big), \tag{4}$$

where X_{train} is a random member span and X_{test} a random non-member span, so nAUC is zero at chance and one under perfect separation. We normalize because a defended model can over-correct and rank its own training spans below held-out ones, which an attacker exploits by flipping the sign; normalizing also puts every score on one scale, so a model's worst case is its largest nAUC. With finitely many spans nAUC is not zero under the null, so every frontier also marks the untrained model as the empirical floor; chance bands and directional AUCs are in Appendix E.4.

3.1 What moves memorization

All runs in this subsection are plain fine-tuning with no defense. We change one factor at a time and report nAUC on the sensitive spans for every score the runs were evaluated with (a blank panel means that score was not computed for those runs); the full sweeps are in Appendix D. The reading that organizes what follows is that a model first learns what is shared across examples and only then, if training continues, fits what is specific to each. Memorization is the second part, and each factor changes either how much example-specific residue there is or how much training pressure is put on it.

Training duration. Every configuration is compared in two regimes: *best-val*, the checkpoint with the lowest validation loss, which is what early stopping selects, and *overtrained*, the same configuration trained longer. Figure 1a compares the two for every configuration we trained both ways: the large majority of pairs lie above the diagonal (per score in Figure 4). Figure 1b follows every run we scored along training (Pythia-70M to 1B (Biderman et al., 2023b) on each testbed) with the epoch axis rescaled so that 1 is each run's best-val epoch (individual runs in Figure 5). After the best-val epoch, memorization keeps rising until roughly ten times that epoch and then plateaus. Stopping at best-val therefore removes much of the memorization, but not all of it, and the earlier the best-val epoch falls on this rise, the more it removes. Consequently, we focus on the best-val checkpoint from here on unless stated otherwise, since it is what a practitioner who stops on validation loss would deploy; the figures keep the overtrained arm alongside, and since it memorizes more everywhere it is also the more demanding case for a defense.

Task hardness. *APEX*-L keeps the *APEX* dossiers and prompt, with the applicant's name and application id prefilled, but the field to produce is the applicant's portal number, computed from six visible ingredients of the dossier by a rule the text never states (see Figure 3 for one sample). The rule comes in five levels, L1 to L5, from a small lookup table to a six-term modular hash; the dossiers are identical across levels, so the levels differ only in how hard the rule is to learn (Appendix C, Table 2). Figure 1c plots memorization against the level, and Figure 11b whether the rule was learned, as open-book accuracy on unseen applicants. At the easy level the model learns the rule within a few epochs, so the portal numbers are derived rather than stored and the best-val checkpoint memorizes almost nothing. At the harder levels the model cannot find the pattern, and the only way left to lower the loss is to bind each portal number to the name and id in the prompt; that binding is memorization, and every unlearnable level is memorized more than the learnable one already at best-val. Among the unlearnable levels the harder the rule, the higher memorization climbs once training continues past best-val (the dashed curve of Figure 1c), whereas at best-val they are not ordered, since validation loss bottoms out within a few epochs whatever the rule, before most of the binding has formed. Appendix D.3 gives per-score curves, best-val epochs and closed-book reproduction of the portal numbers.

Rare, hard subgroups. Real training sets mix easy patterns shared by many examples with hard ones shared by few. We push this to the extreme with a mixture: most of the training set is ordinary *APEX* examples, and a small share is *APEX*-FavNum examples, the same dossiers but with the question asking for the applicant's favorite number, a random four-digit number that appears nowhere in the text, so these examples can only be memorized. For comparison we also train on the minority alone, the same number of examples without any majority.

Figure 1d shows how strongly the minority is memorized at best-val as its share varies, in raw TRA, the average log-probability gap between the target and its reference on the secret.[3] The rarer

[3]Here it is not nAUC because with so few secrets it is close to one in every run and can no longer separate the cases.

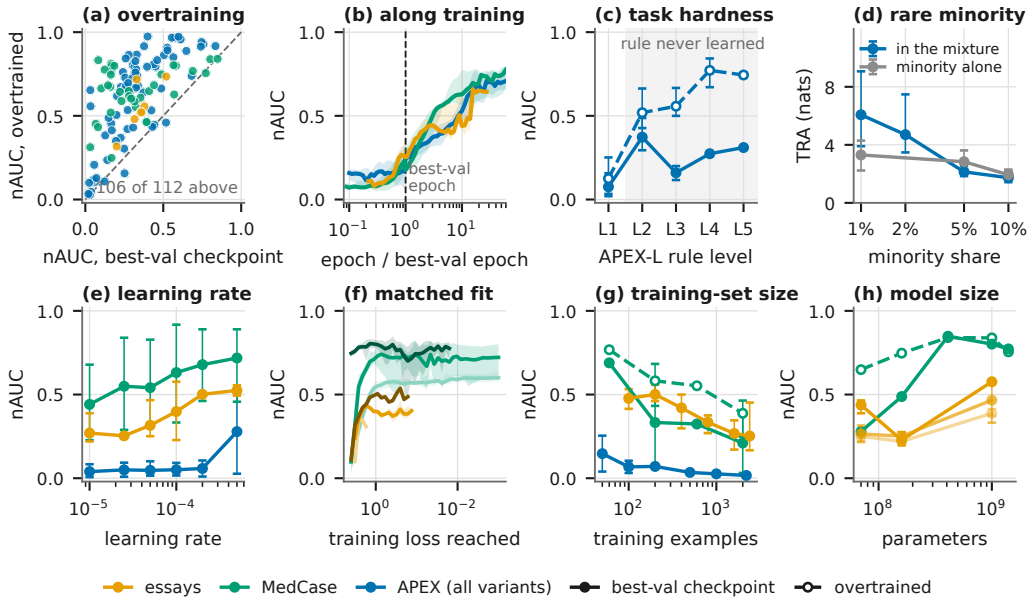


Figure 1: **What moves memorization.** All panels report nAUC on the sensitive spans, averaged over scores, pooling Pythia-70M to 1B (*MedCase* to 1.4B) and seeds, except (d), which reports raw TRA; the less averaged versions are in Appendix D. (a) Each configuration trained twice, best-val (horizontal) against overtrained (vertical): a point above the diagonal memorized more when trained longer. (b) Memorization along training, one median curve per testbed, with the epoch axis rescaled so that 1 is each run's best-val epoch. (c) *APEX*-L: against the rule level that generates the secret; shading marks levels never learned. (d) A rare minority that can only be memorized: *APEX*-FavNum examples, whose answer is a random number absent from the text, at 1 to 10% of a 2000-example *APEX* training set (blue), and the same minority trained alone (grey); TRA on the minority's secrets in nats per token at best-val, pooling Pythia-70M and 160M. (e) Against learning rate at best-val. (f) *MedCase* (green) and *essays* (orange) along training at three learning rates each (lighter colour is smaller), against the training loss reached, on a log axis from about four (left) to a hundredth (right). (g) Against the number of training examples. (h) Against model size with the recipe fixed: *MedCase* in both regimes, *essays* at best-val for three learning rates (darker colour is larger). In all panels, bars and bands show the spread over what was pooled (interquartile range in (b), full range elsewhere).

the minority, the more it is memorized, and at the rarest share it is memorized far more inside the mixture than when trained alone, whereas at 5 to 10% the two are alike. Figure 16b explains why. Trained alone, the minority reaches its best-val checkpoint after one epoch, since there is nothing to generalize. Inside the mixture the majority keeps validation loss improving for several more epochs, each another pass over the minority examples, which the model can only store, and the rarer the minority the longer this goes on. Figure 16c repeats the experiment with an *APEX*-L secret at 1% share in place of an impossible one: the learnable level stays near chance and the harder levels are memorized. In every case the validation loss that decides when to stop is dominated by the majority, so the examples it protects least are the rare, hard ones.

This situation is common, and the *essays* are an instance of it. An email address or a phone number cannot be predicted from the surrounding text, so each sensitive span is a favorite number inside an otherwise learnable essay, and the best-val epoch is set by the essay text, not by the spans. Figure 1b shows the result: on *APEX*, where the whole target is the secret, memorization stays flat until the best-val epoch and rises only afterwards, whereas on the *essays* it rises throughout.

Learning rate. Figure 1e varies only the learning rate, with every run stopped at its own best-val epoch. Memorization increases with the rate on every testbed, and almost every sweep series behind the curve rises too (Appendix D.5, also one score at a time). This is not just more training in disguise: compared at the same training loss rather than the same epoch, the higher rate still

memorizes more at every level of fit on both *MedCase* and the *essays* (Figure 1f), most clearly under the probability score and by a smaller margin under TRA (Figure 23). Our reading is that a larger step imprints more of each example per presentation, while the shared structure that sets the best-val epoch is learned either way. Large learning rates are often credited with an implicit regularization that improves generalization (Li et al., 2019; Smith et al., 2021); whatever it does for the shared structure, it does not extend to the example-specific part of the fit.

Training-set and model size. The fewer examples a model is trained on, the more of each it memorizes, on every testbed, across more than an order of magnitude of size and in both regimes (Figure 1g; Appendix D.2): each example receives a larger share of the updates and there is less shared structure to learn instead, and early stopping helps most at the smallest sizes. Model size is less clear. When overtrained, larger models memorize more (Carlini et al., 2023; Tirumala et al., 2022; Biderman et al., 2023a), and *MedCase* shows this in both regimes (Figure 1h). At the best-val checkpoint, however, we find no consistent pattern: on the *essays* the ordering is not monotone and depends on the learning rate, with the largest model memorizing most when all sizes share a rate and least once its rate is lowered, and on *APEX*-L the sizes are close (Appendix D.6). What survives is a caution: a claim about scale has to say which learning rate and which stopping rule it was made under.

Lessons. What a fine-tuned model memorizes is the part of each example that the other examples do not explain, and it forms after the shared structure has been learned. For a practitioner this gives a short list of practices, each of which helped in our runs and none was enough. Stop on validation loss and do not train past it: the best-val checkpoint removes most of what longer training would add. Use the smallest learning rate that reaches the task performance needed, since at the same training loss a larger rate leaves more of each example behind. Pool as much data as possible, because each example's share of the updates is what matters, and do not assume a larger model is safer: at the best-val checkpoint we found no consistent effect of size, and where larger models looked safer they had been trained at a lower learning rate. Above all, do not read a healthy validation curve as evidence of privacy. Validation loss is an aggregate, set by the majority of the data, and knows nothing about a particular record, so early stopping protects least exactly the spans that matter most: rare, hard-to-predict values inside otherwise learnable text, of which a hard minority inside an easy majority is the extreme case (Figure 1d) and sensitive spans in ordinary documents the common one. Memorization should therefore be measured on those spans directly, against held-out spans, with more than one score, and with an untrained model as the floor. All of these knobs act on every token alike and are bounded by the task and the model one wants, and none brought the sensitive spans to chance in our runs. What is missing is a per-token signal that separates the two kinds of fitting. TRA aims to be that signal: shared structure, including a rule the model has learned, should cancel and only example-specific fitting remain, and on the mixture it is large on the minority's secrets. TRAP penalizes that part while the standard objective keeps learning the rest. The next section tests whether this improves the trade-off between privacy and utility rather than moving along it, comparing defenses at the best-val checkpoint, since that is what would be deployed, and a defense that only removes what overtraining added has done nothing early stopping would not.

4 Mitigating Privacy Memorization

A practitioner has a fixed fine-tuning set with sensitive spans in it and wants the model that learns the task best while memorizing those spans least.[4] This section asks whether TRAP gives a better trade-off than the ordinary knobs and the existing defenses, across datasets and model sizes.

Setup. Every defended model is trained on $\mathcal{D}_{\mathrm{tgt}}$ with the recipe of its undefended counterpart and stopped at its own best-val epoch. Utility is the held-out cross-entropy on the tokens the training loss covers: every token of an essay, and the completion alone for *APEX* and *MedCase*, that is, the diagnosis. Privacy is the worst case among the four memorization scores of Section 3 (probability, loss++, exposure and TRA), that is, the largest nAUC any of them reaches on the sensitive spans. Of the four, TRA needs the reference model, which an attacker with only the released model lacks,

[4] We leave compute out of the comparison: all methods here cost comparable total compute, and in this setting the memorization-utility trade-off is what matters, not the training bill.

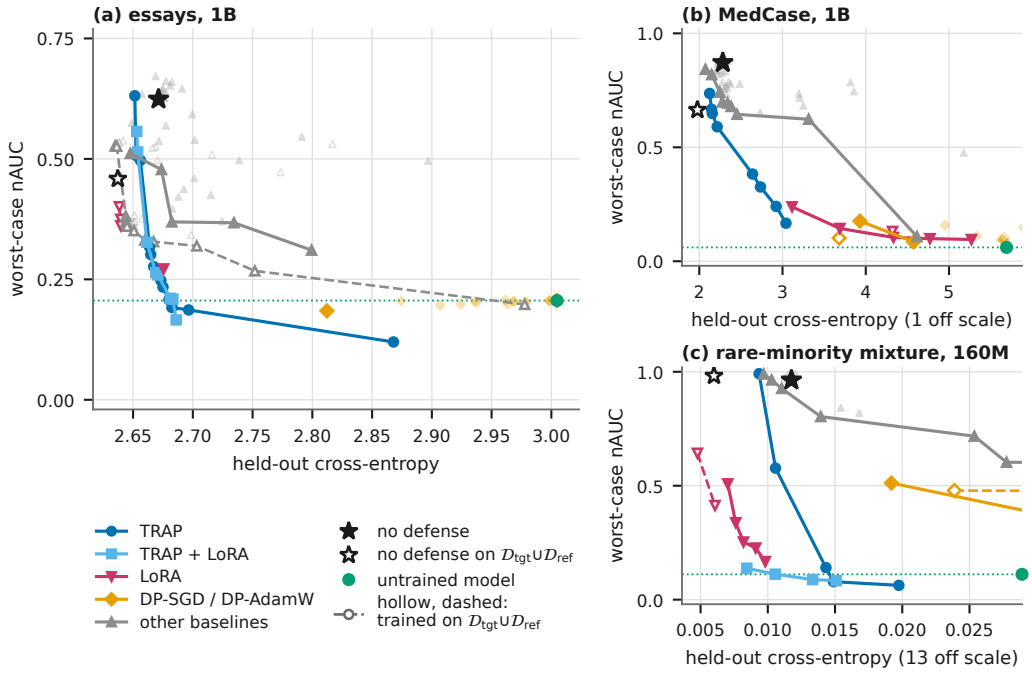


Figure 2: **Memorization against utility.** Held-out cross-entropy (horizontal) against the worst-case nAUC over the scores each run was evaluated with (vertical); lower is better on both. Each line is a method family's lower-left frontier over its hyperparameter values (seed means where seeds exist): TRAP pools its frozen and iterated references, TRAP + LoRA is the same penalty through an adapter, DP the two optimizers, and the grey line pools the remaining baselines into one frontier (drawn one by one in Figures 29, 30 and 34); for the baselines and DP the remaining hyperparameter values are the faint dots. Hollow markers with dashed lines are the same families trained on $\mathcal{D}_{\text{tgt}} \cup \mathcal{D}_{\text{ref}}$; the star is the undefended model, the hollow star its union-trained twin, the green dot and dotted line the untrained model. Points beyond $2.5\times$ the undefended cross-entropy are left out and counted in the axis label; the untrained model is drawn at the right edge when it lies beyond. (a) *Essays* at Pythia-1B. (b) *MedCase* at Pythia-1B. (c) The rare-minority mixture of Section 3.1 at Pythia-160M.

and it is also what TRAP trains against; Appendix E therefore reports the frontiers one score at a time, and TRAP's advantage holds under each. The hyperparameters shared by all methods, such as batch size, learning rate and the early-stopping rule, are held at the testbed's recipe (DP under its own optimizer's); each method is then swept over its own hyperparameters (the penalty weight for TRAP, the rank for LoRA, the privacy budget for DP, and so on), from a value with little effect up to the strongest value in its published range, and drawn as its lower-left frontier through seed means where seeds exist, so the comparison is between curves rather than chosen points and a larger grid cannot buy an advantage from seed noise; Appendix E.2 lists every frontier setting, and TRAP's threshold ϵ is zero unless stated. Figure 2 shows the *essays* and *MedCase* at Pythia-1B and the rare-minority mixture of Section 3.1, *APEX*-FavNum secrets at a 1% share inside ordinary *APEX*, at Pythia-160M. Appendix E repeats the comparison on the *essays* and the mixture at Pythia-70M and on *MedCase* at Llama-3.1-8B and in the overtrained regime, and gives the number of seeds behind every campaign.

Baselines. We compare against what a practitioner might reach for: LoRA (Hu et al., 2022), DP-SGD and DP-AdamW (Abadi et al., 2016; Li et al., 2022), and a set of objectives and regularizers, Goldfish loss (Hans et al., 2024), RelaxLoss (Chen et al., 2022), flooding (Ishida et al., 2020), the confidence penalty (Pereyra et al., 2017), loss truncation (Kang & Hashimoto, 2020), NEFTune (Jain et al., 2024), dropout, weight decay and gradient clipping, plus two of our own design, a KL anchor toward the pretrained base or toward the reference model, the latter the training-time form of bound-

ing divergence from a reference trained without the sensitive content (Vyas et al., 2023), and an n-gram regularizer. At 70M we also ran FedAvg (McMahan et al., 2017) and SCAFFOLD (Karimireddy et al., 2020) with $\mathcal{D}_{\mathrm{tgt}}$ and $\mathcal{D}_{\mathrm{ref}}$ as the two clients. None of the methods, TRAP included, reads the annotations of which tokens are sensitive, and DP points are what the scores measure on those models, not a certified guarantee. TRAP is run with its frozen and iterated references, on the full model and through a LoRA adapter (TRAP + LoRA); Appendix E shows every method on its own.

Results. The picture in Figure 2 is the same in every panel. The undefended model sits in the upper left, and almost every objective and regularizer sits next to it whatever its hyperparameter: tools built against overfitting change utility more than memorization. DP, and on *MedCase* also LoRA, reach the floor only after giving up most of what fine-tuning had bought. TRAP's frontier is the one that descends: near the undefended utility it removes most of the memorization, and a larger weight trades utility for the rest, down to the floor. LoRA competes on the *essays*, where restricting capacity already limits what can be stored, and there TRAP + LoRA is the best of all; on *MedCase* and the mixture TRAP leads. The mixture panel is the uneven case of Section 3.1, a rare, hard minority inside an easy majority (Knolle et al., 2026): the undefended model memorizes the twenty secrets almost perfectly while the aggregate loss looks healthy, TRAP brings their worst case close to the floor at a utility cost of a few thousandths of a nat, and TRAP + LoRA does so below the undefended cross-entropy, with no label to find the minority, since the penalty is largest where the target pulls furthest ahead of its reference. LoRA alone stays above the TRAP + LoRA frontier at every rank. DP enters the panel only at its loosest budget, where it leaks as much as the widest adapter; at the budgets where it does approach the floor it costs several times the undefended cross-entropy. Doubling the data by training on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$ lowers memorization, as Section 3.1 predicts, but neither it nor the union-trained baselines reach TRAP's frontier.

This is what Section 3.1 predicts, and it comes down to how each method decides which part of the training signal to alter. DP-SGD clips every example's gradient to the same norm and adds the same noise, so it restrains most the examples with large gradients; gradient clipping and loss truncation act on the size of the gradient or the loss, the confidence penalty and RelaxLoss on how peaked the prediction is, Goldfish loss on a pseudorandom mask, LoRA on how much can be stored at all. Each of these is at most correlated with memorization: a token can have a large gradient, a high loss or a confident prediction because it is hard, or because the pattern behind it is real and shared by the rest of the data, and none of these criteria can tell the two apart. TRAP's criterion is the advantage over a reference trained on other data, which is meant to be zero for anything that data supports, so the penalty falls on the example-specific part of the fit and on little else. Consequently, the knobs move memorization only as far as they move utility while TRAP moves the frontier, and the adapter composes with it, since one limits how much can be stored and the other what is stored. With twenty member spans, on *MedCase* and the mixture, differences among defended models near the floor are directions rather than rankings.

5 Conclusion

We studied when a fine-tuned language model memorizes the sensitive spans in its training data and how to stop it. Most memorization scores compare the model's likelihood with a reference, and a reference trained on the other half of the data gives TRA, a score that is cheap, differentiable, and equal in expectation to the generalization gap. Measured this way, memorization is the fit of what the training examples do not share: it grows with training past the best-val epoch, smaller training sets, larger learning rates, harder tasks and rare subgroups, with no consistent effect of model size at the best-val checkpoint, and early stopping removes much of it but not all. TRAP penalizes TRA token by token, needs no annotation of which tokens are sensitive, and gives a better memorization–utility trade-off than differential privacy, adapters and a dozen regularizers on every testbed we ran. It is an empirical defense rather than a formal guarantee, and our scores describe an attacker who does not adapt to it; one who calibrates against shadow models of his own (Carlini et al., 2022) is untested. The shipped model has also learned from only half the data and still beats the baselines trained on all of it; a pair of models each defended against the other would leave no data unused. Finally, a smaller learning rate memorizes less at the same task performance, and we could not say why.

ETHICS STATEMENT

The educational testbed contains annotated student personal information, in which the corpus curators replaced the original values with surrogates of the same type before release. We report only aggregate measurements and do not reproduce identifiers in the paper. Redistribution of raw or derived data must follow the source license and an independent privacy review. The remaining testbeds use synthetic identifiers or planted cues; their purpose is to measure memorization under controlled conditions, not to model every deployment threat.

REPRODUCIBILITY STATEMENT

Appendix C specifies dataset construction and realized splits; Appendix E reports full mitigation frontiers; Appendix B defines the scores; and Appendix F documents reference schedules. We distinguish optimizer seeds from data partitions, report the unit and size of each score evaluation, and mark results that are currently single-seed. In addition, a codebase is shared as supplementary material.

AI USE STATEMENT

In this work, we used generative AI tools to generate synthetic data sets: GPT-5.5 produced template prose for the synthetic testbeds before deterministic substitution of the synthetic fields (Appendix C.3). We also used them to implement methods: coding assistants wrote and edited code for data preparation, training, scoring and plotting under the authors' direction, and the authors reviewed and tested that code for correctness. We have not used generative AI tools to develop the theoretical framework, to formulate or prove mathematical claims, to propose hypotheses, to design the methodology or experiments, to clean or reformat datasets, or to interpret results; translation and qualitative analysis are not applicable to this work. Additionally, we used generative AI tools to edit the paper for readability, to organize the LaTeX sources, to format and check references, and to check internal consistency. We have reviewed all AI-assisted work: the authors verified the equations, citations, experimental measurements and claims. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

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

Atilla Akkus, Masoud Poorghaffar Aghdam, Mingjie Li, Junjie Chu, Michael Backes, Yang Zhang, and Sinem Sav. Generated data with fake privacy: Hidden dangers of fine-tuning large language models on generated data. In *USENIX Security Symposium*, 2025.

Stella Biderman, USVSN Sai Prashanth, Lintang Sutawika, Hailey Schoelkopf, Quentin Anthony, Shivanshu Purohit, and Edward Raff. Emergent and predictable memorization in large language models. In *Advances in Neural Information Processing Systems*, 2023a.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, 2023b.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security Symposium*, 2019.

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

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy*, pp. 1897–1914. IEEE, 2022. doi: 10.1109/SP46214.2022.9833649. URL https://ieeexplore.ieee.org/document/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*, 2023.

Dingfan Chen, Ning Yu, and Mario Fritz. RelaxLoss: Defending membership inference attacks without losing utility. In *International Conference on Learning Representations*, 2022.

Rishav Chourasia, Batnyam Enkhtaivan, Kunihiro Ito, Junki Mori, Isamu Teranishi, and Hikaru Tsuchida. Knowledge cross-distillation for membership privacy. *Proceedings on Privacy Enhancing Technologies*, 2022(2):362–377, 2022. doi: 10.2478/popets-2022-0050. URL https://petsymposium.org/popets/2022/popets-2022-0050.php.

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*, 2024.

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014. doi: 10.1561/0400000042. URL https://doi.org/10.1561/0400000042.

Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*. ACM, 2020. doi: 10.1145/3357713.3384290.

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1e14bfe2714193e7af5abc64ecbd6b46-Abstract.html.

Wenjie Fu, Huandong Wang, Chen Gao, Guanghua Liu, Yong Li, and Tao Jiang. Membership inference attacks against fine-tuned large language models via self-prompt calibration. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-4290. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/f36ad694188bb4c4bbbd61e2038e069e-Abstract-Conference.html.

Abhimanyu Hans, Yuxin Wen, Neel Jain, John Kirchenbauer, Hamid Kazemi, Prajwal Singhania, Siddharth Singh, Gowthami Somepalli, Jonas Geiping, Abhinav Bhatele, and Tom Goldstein. Be like a goldfish, don't memorize! mitigating memorization in generative LLMs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 24022–24045, 2024. doi: 10.52202/079017-0757. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/2ad2dffba5079687651226ac8752df97-Abstract-Conference.html.

Langdon Holmes, Scott Crossley, Perpetual Baffour, Jules King, Lauryn Burleigh, Maggie Demkin, Ryan Holbrook, Walter Reade, and Addison Howard. The Learning Agency Lab - PII Data Detection. https://www.kaggle.com/competitions/pii-detection-removal-from-educational-data, 2024. Kaggle.

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*, 2022.

Takashi Ishida, Ikko Yamane, Tomoya Sakai, Gang Niu, and Masashi Sugiyama. Do we need zero training loss after achieving zero training error? In *International Conference on Machine Learning*, 2020.

Neel Jain, Ping-yeh Chiang, Yuxin Wen, John Kirchenbauer, Hong-Min Chu, Gowthami Somepalli, Brian R. Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. NEFTune: Noisy embeddings improve instruction finetuning. In *International Conference on Learning Representations*, 2024.

Nikhil Kandpal, Eric Wallace, and Colin Raffel. Deduplicating training data mitigates privacy risks in language models. In *International Conference on Machine Learning*, 2022.

Daniel Kang and Tatsunori Hashimoto. Improved natural language generation via loss truncation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, 2020.

Moritz A. Knolle, Martin J. Menten, Friederike Jungmann, Felix Meissen, Ben Glocker, Daniel Rueckert, and Georgios Kaissis. Disparate privacy risks from medical AI. *Nature*, 656:192–198, 2026. doi: 10.1038/s41586-026-10688-0.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 2022.

Xuechen Li, Florian Tramèr, Percy Liang, and Tatsunori Hashimoto. Large language models can be strong differentially private learners. In *International Conference on Learning Representations*, 2022.

Yuanzhi Li, Colin Wei, and Tengyu Ma. Towards explaining the regularization effect of initial large learning rate in training neural networks. In *Advances in Neural Information Processing Systems*, 2019.

Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Beguelin. Analyzing leakage of personally identifiable information in language models. In *IEEE Symposium on Security and Privacy*, 2023.

Justus Mattern, Fatemehsadat Mireshghallah, Zhijing Jin, Bernhard Schölkopf, Mrinmaya Sachan, and Taylor Berg-Kirkpatrick. Membership inference attacks against language models via neighbourhood comparison. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 11330–11343. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-acl.719. URL https://aclanthology.org/2023.findings-acl.719/.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics*, 2017.

John X. Morris, Chawin Sitawarin, Chuan Guo, Narine Kokhlikyan, G. Edward Suh, Alexander M. Rush, Kamalika Chaudhuri, and Saeed Mahloujifar. How much can language models memorize? In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2505.24832.

Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine learning with membership privacy using adversarial regularization. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 634–646. ACM, 2018. doi: 10.1145/3243734.3243855.

Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne Ippolito, Christopher A. Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. Scalable extraction of training data from (production) language models. *arXiv preprint arXiv:2311.17035*, 2023.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. In *International Conference on Learning Representations, Workshop Track*, 2017.

Zhiwen Ruan, Yun Chen, Yutao Hou, Peng Li, Yang Liu, and Guanhua Chen. Unveiling over-memorization in finetuning LLMs for reasoning tasks, 2025. URL https://arxiv.org/abs/2508.04117. Preprint.

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*, pp. 3–18. IEEE, 2017. doi: 10.1109/SP.2017.41.

Samuel L. Smith, Benoit Dherin, David G. T. Barrett, and Soham De. On the origin of implicit regularization in stochastic gradient descent. In *International Conference on Learning Representations*, 2021.

Liwei Song and Prateek Mittal. Systematic evaluation of privacy risks of machine learning models. In *USENIX Security Symposium*, 2021.

Xinyu Tang, Saeed Mahloujifar, Liwei Song, Virat Shejwalkar, Milad Nasr, Amir Houmansadr, and Prateek Mittal. Mitigating membership inference attacks by Self-Distillation through a novel ensemble architecture. In *31st USENIX Security Symposium*, pp. 1433–1450. USENIX Association, 2022. URL https://www.usenix.org/conference/usenixsecurity22/presentation/tang.

Kushal Tirumala, Aram H. Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. Memorization without overfitting: Analyzing the training dynamics of large language models. In *Advances in Neural Information Processing Systems*, 2022.

Trishita Tiwari, Ari Trachtenberg, and G. Edward Suh. A prior-aware metric for efficiently distinguishing memorization from generalization in large language models. In *Conference on Language Modeling*, 2026. URL https://arxiv.org/abs/2602.18733.

Nikhil Vyas, Sham M. Kakade, and Boaz Barak. On provable copyright protection for generative models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 35277–35299, 2023. URL https://proceedings.mlr.press/v202/vyas23b.html.

Kevin Wu, Eric Wu, Rahul Thapa, Kevin Wei, Angela Zhang, Arvind Suresh, Jacqueline J. Tao, Min Woo Sun, Alejandro Lozano, and James Zou. MedCaseReasoning: Evaluating and learning diagnostic reasoning from clinical case reports. *arXiv preprint arXiv:2505.11733*, 2025.

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

Shenglai Zeng, Yaxin Li, Jie Ren, Yiding Liu, Han Xu, Pengfei He, Yue Xing, Shuaiqiang Wang, Jiliang Tang, and Dawei Yin. Exploring memorization in fine-tuned language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

Chiyuan Zhang, Daphne Ippolito, Katherine Lee, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. Counterfactual memorization in neural language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 39321–39362, 2023. doi: 10.52202/075280-1708. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/7bc4f74e35bcfe8cfe43b0a860786d6a-Abstract-Conference.html.

Jingyang Zhang, Jingwei Sun, Eric Yeats, Yang Ouyang, Martin Kuo, Jianyi Zhang, Hao Frank Yang, and Hai Li. Min-K%++: Improved baseline for pre-training data detection from large language models. In *International Conference on Learning Representations*, 2025.

A Background and Related Work

Understanding and measuring memorization. Membership inference asks whether a text was in the training set (Shokri et al., 2017; Yeom et al., 2018; Song & Mittal, 2021; Carlini et al., 2022), with scores that threshold the loss, average the least likely tokens (Shi et al., 2024; Zhang et al., 2025), or calibrate against another model or nearby texts (Mattern et al., 2023; Fu et al., 2024); exposure ranks a canary among decoys (Carlini et al., 2019); extraction measures what prompting recovers (Carlini et al., 2021; 2023; Lukas et al., 2023); and counterfactual memorization compares models trained with and without an example (Feldman, 2020; Feldman & Zhang, 2020; Zhang et al., 2023), the cleanest definition but one that needs many models, which cheaper reference-subtracting proxies approximate (Tiwari et al., 2026; Morris et al., 2026). Whether membership scores work at all on large models is contested (Duan et al., 2024). Section 2 writes these measures as one comparison against a reference, and our experiments report four of them side by side. As for what drives memorization, it grows with model size, duplication and context length (Carlini et al., 2023), appears early along training and without overfitting (Tirumala et al., 2022), is only partly predictable from smaller models or earlier checkpoints (Biderman et al., 2023a), fills a capacity of a few bits per parameter (Morris et al., 2026), and is reduced by deduplication (Kandpal et al., 2022; Lee et al., 2022); fine-tuning on task or synthetic data shows the same (Zeng et al., 2024; Ruan et al., 2025; Akkus et al., 2025). Section 3 revisits these factors for sensitive spans in fine-tuning data under matched recipes.

Mitigation. Differentially private training gives a formal guarantee at a utility cost (Dwork & Roth, 2014; Abadi et al., 2016; Li et al., 2022). Most other defenses reuse tools built against overfitting, such as dropout, weight decay, gradient clipping, flooding (Ishida et al., 2020), the confidence penalty (Pereyra et al., 2017), loss truncation (Kang & Hashimoto, 2020), noisy embeddings (Jain et al., 2024) and RelaxLoss (Chen et al., 2022); Goldfish loss drops a pseudorandom subset of tokens from the loss (Hans et al., 2024); parameter-efficient fine-tuning limits how much can be stored (Hu et al., 2022); federated training spreads the data over clients (McMahan et al., 2017; Karimireddy et al., 2020); and membership-inference defenses regularize against an attacker or distil through an ensemble (Nasr et al., 2018; Chourasia et al., 2022; Tang et al., 2022). Closest to TRAP is the idea of bounding a model's divergence from a reference trained without the sensitive content (Vyas et al., 2023); TRAP instead acts during training and token by token, penalizing only the tokens whose likelihood under the target exceeds the reference by more than a threshold, with no annotation of which tokens are sensitive. Section 4 compares it with the methods above, except the membership-inference defenses, and with two of our own design, a KL anchor toward the base or the reference model and an n-gram regularizer.

B A Common Likelihood-and-Reference View of Memorization

Section 2.1 made two claims about the memorization measures and attacks reviewed in Appendix A. First, they share one core: each compares the probability the model assigns to the actual tokens of a text with a baseline for how probable those tokens should be, and reads a probability that is too high relative to the baseline as memorization. Second, measures and attacks are two sides of the same thing. Thresholding, ranking, or searching with a measure gives an attack, and an attack's success on a candidate is itself a measure. This appendix backs up both claims by writing each measure out, pointing at its baseline, which we call the reference, and at the rule that turns it into an attack. The measures differ mainly in the reference. Going down the list, it is a fixed threshold, the model's own prediction at each position, a set of decoy values, perturbed copies of the text, a compressor, a general-purpose language model, models trained without the text, and, for TRA, a model trained on the other half of the same data.

Whole documents versus sensitive spans. The papers that introduced these measures apply them to a whole document or to a synthetic canary, so their sums run over every token (Yeom et al., 2018; Carlini et al., 2019; 2021; Shi et al., 2024; Zhang et al., 2025). We ask a more focused question, whether the model has memorized one sensitive span (a name, an identifier, a diagnosis) inside a document. So throughout this appendix every score is written over the position set $\mathcal{J}_x$ of Section 2.1, which in our experiments contains exactly the tokens of the sensitive span. Setting $\mathcal{J}_x = \{1, \ldots, T\}$ gives back the original whole-document versions.

Conventions. $\ell(\theta, x)$ is the mean negative log-likelihood over $\mathcal{J}_x$, as in Equation 18. All scores are oriented so that a larger value means stronger evidence of memorization. One caveat before starting: sharing the same core does not make these measures interchangeable. A membership attack decides about a candidate it is handed, exposure ranks the true value among decoys, counterfactual memorization measures the effect of including a sample, and extraction has to actually recover the text under a given prompt, decoder, and budget. A high score under one of them is evidence of memorization in that sense, not automatically in the others.

B.1 LOSS, PERPLEXITY, AND THE PROBABILITY SCORE

The LOSS score (Yeom et al., 2018) is just the mean log-likelihood of the evaluated tokens,

$$S_{\mathrm{LOSS}}(\theta; x) = \frac{1}{|\mathcal{J}_x|} \sum_{t \in \mathcal{J}_x} \log p_\theta(x_t \mid x_{<t}) = -\ell(\theta, x). \tag{5}$$

The main text reports it as a probability,

$$S_{\mathrm{prob}}(\theta; x) = \exp S_{\mathrm{LOSS}}(\theta; x) = \Big(\prod_{t \in \mathcal{J}_x} p_\theta(x_t \mid x_{<t}) \Big)^{1/|\mathcal{J}_x|}, \tag{6}$$

the per-token probability of the exact span. LOSS, perplexity $e^{\ell(\theta,x)}$, and this probability are monotone in one another, so they rank spans identically and give the same AUC.

To attack with LOSS, one picks a threshold τ, using known non-members or public data, and calls x a member whenever $S_{\mathrm{LOSS}}(\theta; x) > \tau$. So the comparison is the probability of the span tokens against one fixed number, the same for every candidate; in Equation 1 this is $q_{\mathrm{ref}} = \mathrm{const}$. This is also the weakness of LOSS: text that is easy to predict for everyone looks the same as text that is easy because it was trained on (Yeom et al., 2018; Mattern et al., 2023).

B.2 MIN-K%, MIN-K%++, AND LOSS++

Min-K% (Shi et al., 2024) is LOSS averaged over only the least likely tokens. With $k = \lceil K|\mathcal{J}_x|/100 \rceil$ and $I_K(x)$ the k positions in $\mathcal{J}_x$ with the smallest $\log p_\theta(x_t \mid x_{<t})$,

$$S_{\mathrm{MinK}}(\theta; x) = \frac{1}{k} \sum_{t \in I_K(x)} \log p_\theta(x_t \mid x_{<t}). \tag{7}$$

The idea is that unseen text tends to contain a few very unlikely tokens, while memorized text does not. The reference is again a threshold; only the positions that enter the average change.

Min-K%++ (Zhang et al., 2025) compares each token with the model's own prediction at that position instead. Write $p_t(v) = p_\theta(v \mid x_{<t})$ and let

$$\mu_t = \sum_v p_t(v) \log p_t(v), \qquad \sigma_t^2 = \sum_v p_t(v) \big(\log p_t(v) - \mu_t \big)^2,$$
$$z_t = \frac{\log p_\theta(x_t \mid x_{<t}) - \mu_t}{\sigma_t}. \tag{8}$$

Then $S_{\mathrm{MinK^{++}}}$ is the mean of the k smallest z_t. Here μ_t is the log-probability the model expects to assign to its next token (the negative entropy of p_t). Subtracting it asks how much more likely the observed token was than a typical draw from the model at that position, and dividing by σ_t puts sharp and flat positions on one scale. So the comparison is still the probability of the observed token against a baseline, but the baseline is now computed per position from the model itself rather than fixed once for all candidates. The attack is again a threshold on the score.

On a span of a few tokens the lowest $K\%$ keeps one or two tokens and mostly adds noise, so we set $K = 100\%$, under which Min-K% becomes LOSS and Min-K%++ becomes the mean of z_t over the span (the *APEX*-full grid of Appendix D.3 was scored at $K = 20\%$, as its captions say). That mean is what we report and call loss++,

$$S_{\mathrm{loss{+}{+}}}(\theta; x) = \frac{1}{|\mathcal{J}_x|} \sum_{t \in \mathcal{J}_x} \frac{\log p_\theta(x_t \mid x_{<t}) - \mu_t}{\sigma_t}. \tag{9}$$

It keeps the part of Min-K%++ that matters on short spans, the per-position standardization, and drops the selection step that does nothing there.

The standardized score has a ceiling. As the model approaches certainty on the observed token, the numerator $\log p_\theta(x_t \mid x_{<t}) - \mu_t$ and the scale σ_t vanish together and z_t tends to $\sqrt{(1-q_t)/q_t}$, where q_t is the probability of the observed token, almost regardless of the rest of the distribution. A fully memorized span is therefore mapped close to zero, the same place as a span the model merely finds easy, and among confident positions the score falls as confidence rises. Where the model is instead certain about a different token, σ_t vanishes while the numerator does not and z_t diverges; the implementation clamps the variance at 10^{-12}, so such values are large but finite, and because every score is read as an AUC they change ranks rather than magnitudes. We report loss++ for comparability with prior work, not as the score to rely on near the floor.

B.3 Exposure

Exposure (Carlini et al., 2019) compares the true value with a set of decoys. Let $\mathcal{R}_x$ be a set of candidate texts that are identical to x outside the span and differ only in the span, with x itself among them. The rank of the true span is one plus the number of decoys that score at least as high,

$$\mathrm{rank}_\theta(x; \mathcal{R}_x) = 1 + \sum_{x' \in \mathcal{R}_x \setminus \{x\}} \mathbf{1}\{S_{\mathrm{LOSS}}(\theta; x') \geq S_{\mathrm{LOSS}}(\theta; x)\}\,, \tag{10}$$

and exposure is

$$S_{\mathrm{Exp}}(\theta; x) = \log_2 |\mathcal{R}_x| - \log_2 \mathrm{rank}_\theta(x; \mathcal{R}_x). \tag{11}$$

It is zero when the true value ranks last and $\log_2 |\mathcal{R}_x|$ when it ranks first. The comparison is the probability of the true span tokens against the probabilities of the same positions filled with other values; the decoys are the reference. Exposure values are therefore only comparable between spans that use the same kind and number of decoys. As an attack, a high exposure means a guesser who tries candidates in order of model probability finds the true value early.

In the original setting $\mathcal{R}_x$ is the set of all values a synthetic canary could have taken. Our spans are real values, so we build $\mathcal{R}_x$ from the true value plus up to 63 other values of the same type drawn from the corpus, each substituted into the same context. Thus $|\mathcal{R}_x| \leq 64$ and exposure is at most 6 bits.

B.4 Neighborhood references

The neighborhood attack (Mattern et al., 2023) compares the text with perturbed copies of itself. It builds J neighbors $\widetilde{x}_j$ of x, for instance by replacing a few words with a masked language model, and uses their average likelihood under the target as the reference,

$$S_{\mathrm{nbr}}(\theta; x) = \frac{1}{|\mathcal{J}_x|} \sum_{t \in \mathcal{J}_x} \log p_\theta(x_t \mid x_{<t}) - \frac{1}{J} \sum_{j=1}^{J} \frac{1}{|\mathcal{J}_x|} \sum_{t \in \mathcal{J}_x} \log p_\theta(\widetilde{x}_{j,t} \mid \widetilde{x}_{j,<t}). \tag{12}$$

The neighbors estimate how likely a text of this shape should be regardless of whether this exact text was seen. If the original stands out from its neighbors, it was probably trained on. Once more this is the probability of the actual tokens against a baseline, here the probability of nearby texts under the same model, and the attack thresholds the difference.

B.5 Compression and transformation references

Two references from Carlini et al. (2021) need no model at all. One is a general-purpose compressor: with $H_{\mathrm{zlib}}(x)$ the compressed length of the text under zlib, the ratio $\ell(\theta, x)/H_{\mathrm{zlib}}(x)$ is small when the model finds the text much easier than a compressor does. The other is the text itself after a transformation, for instance $\ell(\theta, x) - \ell(\theta, \mathrm{lower}(x))$, which is strongly negative when the model is much more confident in the exact casing than in the lowercased version. In both, the model's probability of the text is compared with a rough, model-free estimate of how hard the text should be, and texts whose probability is too high for their difficulty are flagged as memorized.

B.6 REFERENCE-MODEL SCORES

A second language model can serve as the reference. The score is the mean log-likelihood ratio,

$$S_{\mathrm{ref}}(\theta, \theta_{\mathrm{ref}}; x) = \frac{1}{|\mathcal{J}_x|} \sum_{t \in \mathcal{J}_x} \left[\log p_\theta(x_t \mid x_{<t}) - \log p_{\theta_{\mathrm{ref}}}(x_t \mid x_{<t}) \right]. \tag{13}$$

The reference model says how likely the text is for a model that did not train on it; if the target finds the text much easier, that is evidence the target saw it. In the literature the reference is usually a capable language model that is available anyway: the pretrained model before fine-tuning, a smaller model from the same family, or a model trained on similar public data (Carlini et al., 2021). LiRA (Carlini et al., 2022) goes further and trains shadow models on data from the same distribution, then compares the target's likelihood with the distribution of shadow likelihoods.

A generic reference discounts what any language model would predict, such as grammar, common phrases, and general knowledge. It does not discount what can be learned from the fine-tuning task itself. A target that has simply learned the task well therefore looks like it is memorizing when compared with a generic reference, even on spans it has never seen. So what the score means depends entirely on which model plays θ_{ref}, and the next subsection is about that choice. The attack, as before, thresholds the ratio.

B.7 COUNTERFACTUAL MEMORIZATION AND TRA

Counterfactual memorization (Feldman, 2020; Zhang et al., 2023) uses the cleanest reference: models trained without the sample. For a randomized training procedure Train and a random training subset $\mathcal{S}$, it is

$$\begin{aligned} S_{\mathrm{cf}}(x) = \mathbb{E}\left[\frac{1}{|\mathcal{J}_x|} \sum_{t \in \mathcal{J}_x} \log p_{\mathsf{Train}(\mathcal{S})}(x_t \mid x_{<t}) \,\middle|\, x \in \mathcal{S} \right] \\ - \mathbb{E}\left[\frac{1}{|\mathcal{J}_x|} \sum_{t \in \mathcal{J}_x} \log p_{\mathsf{Train}(\mathcal{S})}(x_t \mid x_{<t}) \,\middle|\, x \notin \mathcal{S} \right], \end{aligned} \tag{14}$$

the expected likelihood of x under models that trained on it minus the expected likelihood under models that did not. Both sides train on the same kind of data, so everything a model would learn about x from the rest of the data cancels, including the task; what remains is the effect of x itself being in the training set. This is the same comparison as Equation 13, the probability of the span tokens under a model that saw x against a model that did not, with the reference chosen as carefully as possible. The cost is that estimating the two expectations takes many trained models.

TRA in Equation 2 is Equation 13 with one specific reference: θ is trained on $\mathcal{D}_{\mathrm{tgt}}$, which contains x, and θ_{ref} is trained with the same recipe on the complementary half $\mathcal{D}_{\mathrm{ref}}$, which does not. Because the reference sees the same task and the same kind of data, it discounts the task-level structure that a generic reference misses, as counterfactual memorization does, but with two models instead of many. The connection is exact in expectation. Take Equation 14 with $\mathcal{S}$ a uniformly random half of the corpus. A random balanced split that puts x in $\mathcal{D}_{\mathrm{tgt}}$ gives $\mathcal{D}_{\mathrm{tgt}}$ the law of $\mathcal{S}$ given $x \in \mathcal{S}$ and $\mathcal{D}_{\mathrm{ref}}$ the law of $\mathcal{S}$ given $x \notin \mathcal{S}$, so with the same training procedure on both halves

$$\begin{aligned} \mathbb{E}\big[\mathrm{TRA}(x)\big] = \mathbb{E}\Big[\tfrac{1}{|\mathcal{J}_x|} \textstyle\sum_{t \in \mathcal{J}_x} \log p_\theta(x_t \mid x_{<t}) \,\Big|\, x \in \mathcal{D}_{\mathrm{tgt}} \Big] \\ - \mathbb{E}\Big[\tfrac{1}{|\mathcal{J}_x|} \textstyle\sum_{t \in \mathcal{J}_x} \log p_{\theta_{\mathrm{ref}}}(x_t \mid x_{<t}) \,\Big|\, x \notin \mathcal{D}_{\mathrm{ref}} \Big] = S_{\mathrm{cf}}(x), \end{aligned} \tag{15}$$

the expectation running over the split and the training randomness. So one target–reference pair is an unbiased estimate of the counterfactual effect of including x, at inclusion rate one half. Two limitations follow. A single pair is noisy, since it inherits the randomness of the split and of training, and the identity is for two models trained the same way, so it does not apply to the final pair in which the target is trained with TRAP and the reference is not. As an attack, TRA is thresholded like any other score.

B.8 Definitions and attacks are the same thing used differently

Every score above has the same shape: the probability of the true span tokens under the model, compared with a reference. A memorization *definition* reports that comparison, for one sample or in expectation over training randomness. An *attack* adds a decision rule to it. There are three such rules in the literature, and each one can be run backwards to recover a definition from an attack.

Thresholding. Membership inference picks a threshold τ and calls x a member when $S(x) > \tau$. Since every S is a log-probability minus a reference, the threshold is not a separate ingredient: $S(x) > \tau$ is the same comparison with the reference shifted up by τ, that is, with $\log q_{\mathrm{ref}}$ replaced by $\log q_{\mathrm{ref}} + \tau$ in Equation 1. Choosing τ is choosing how much more probable than the reference a span has to be before we call it memorized. Sweeping τ sweeps that shift and traces the ROC curve, whose area is the probability that a random member outscores a random non-member,

$$\mathrm{AUC}(S) = \Pr\big(S(x_{\mathrm{member}}) > S(x_{\mathrm{nonmember}})\big) + \tfrac{1}{2}\Pr\big(S(x_{\mathrm{member}}) = S(x_{\mathrm{nonmember}})\big). \quad (16)$$

The main text's Equation 4 omits the tie term, which matters only for exposure, whose scores are ranks. The AUC therefore summarizes the attack over all shifts of the reference at once, which is why we use its normalized form Equation 4 to compare the four scores we report. Read backwards, the attack's score on a candidate is a memorization measure for that candidate, and its AUC is a memorization measure for the model. This is why the scores of Appendices B.1 to B.6, most of which were introduced as membership attacks, double as memorization definitions, and why counterfactual memorization, introduced as a definition, is also a membership attack, the one with the cleanest reference.

Ranking. Exposure replaces the threshold by a rank among decoys. The attack is a guesser who tries candidate values in order of model probability; the definition is how early the true value comes up. The two are the same number, $\mathrm{rank}_\theta(x; \mathcal{R}_x)$, read as an attack cost or as a memorization score.

Searching. Extraction is the simplest of all: just look at what the model generates. Prompt it with the context before the span, sample a continuation, and say the span is memorized if the sensitive tokens come out. The probability of that event is the model's probability of the span given its prefix,

$$\Pr\big(\text{sample} = x_{\mathcal{J}_x}\big) = \prod_{t \in \mathcal{J}_x} p_\theta(x_t \mid x_{<t}) = S_{\mathrm{prob}}(\theta; x)^{|\mathcal{J}_x|}, \quad (17)$$

so in expectation extraction *is* the probability score of Appendix B.1, and drawing several independent samples multiplies the success rate by roughly their number, as long as that product stays well below one. Greedy decoding is the deterministic version: the span comes out exactly when the true token is the model's top choice at every span position, that is, when $p_\theta(x_t \mid x_{<t}) \geq \max_{v \neq x_t} p_\theta(v \mid x_{<t})$ for all $t \in \mathcal{J}_x$, which compares the true tokens with the competing tokens instead of with a fixed threshold. Read backwards, whether a span is extracted is a memorization measure for that span and the extraction rate is one for the model. What extraction adds beyond the scores above is not a different quantity but a dependence on the prompt, the decoder, and the number of samples.

In short, the memorization definitions and the attacks in Appendix A are one family of comparisons between the model's probability of the sensitive tokens and a reference, and the members differ in the reference and in the decision rule. The general form Equation 1 is the unifying picture; TRA is that form with the reference chosen to fit the question we ask, a model trained the same way on the other half of the same data, so that everything the task teaches cancels and only what is specific to the training half remains.

C Testbed Construction

This appendix describes how each testbed was built and trained. All of them use the split of Section 3: the training data is divided into two disjoint halves $\mathcal{D}_{\mathrm{tgt}}$ and $\mathcal{D}_{\mathrm{ref}}$, one model is trained on each with the same recipe, and for an example in $\mathcal{D}_{\mathrm{tgt}}$ the model trained on $\mathcal{D}_{\mathrm{tgt}}$ is the target and the model trained on $\mathcal{D}_{\mathrm{ref}}$ a reference that never saw it. The counts below are what is on disk; where generation involved rejection they differ slightly from the requested numbers.

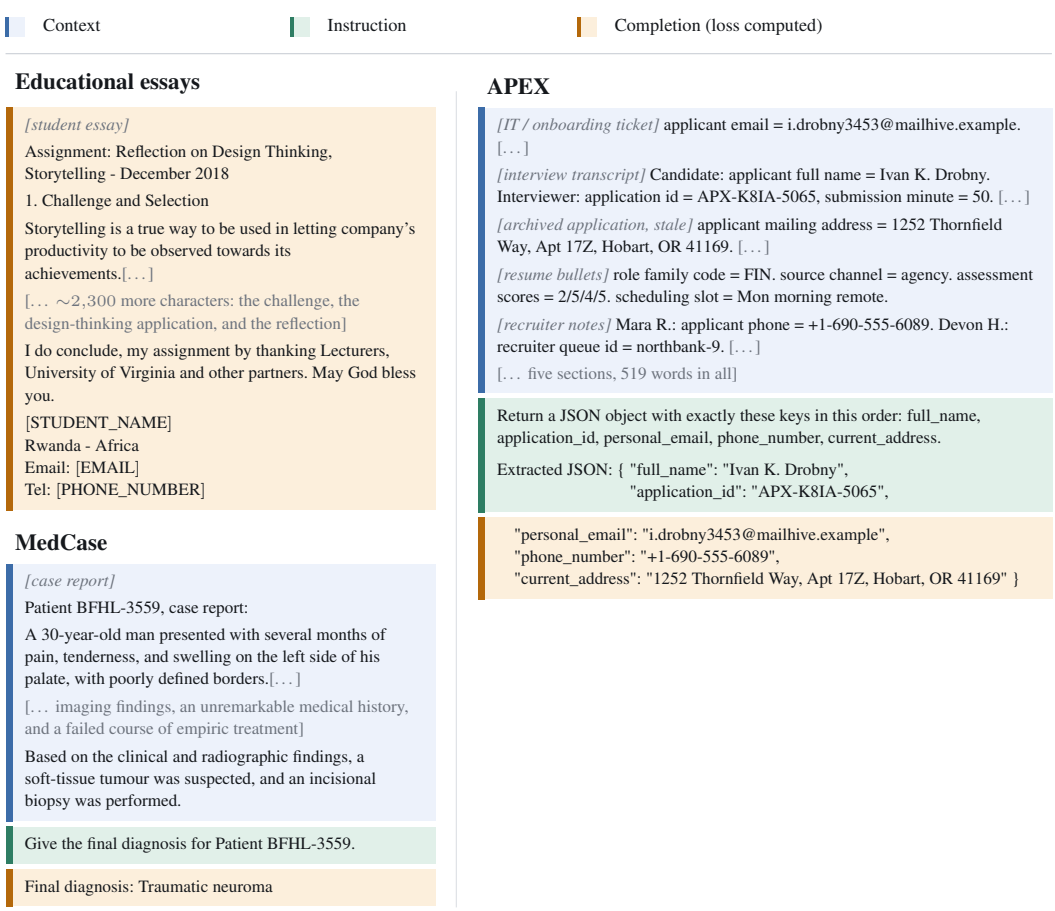


Figure 3: **One training example from each testbed.** Colour marks what the loss covers: blue is context the model reads but is not scored on, green the instruction and any prefilled prefix, and amber the completion, the only part the cross-entropy and the TRAP penalty see. An essay is amber throughout, since training predicts every token of it, so the annotated sensitive spans (here the closing name, email and phone) fall inside the loss. In *APEX* the applicant's fields are scattered over five sections and 519 words, among one to three other applicants, and the three scored fields (email, phone, address) appear verbatim in the dossier; in *APEX*-L the target is instead a portal number computed from six visible ingredients by a rule the dossier never states. *MedCase* is trained open book, with the blue and green blocks as its prompt, and evaluated closed book, with the green block alone as the prompt: with the case report removed, the synthetic patient identifier is the only evidence left, so a correct diagnosis has to come from a memorized identifier-to-diagnosis association. Excerpts are abridged where marked, and the essay's sensitive values are shown as placeholders.

C.1 *Essays*: STUDENT ESSAYS WITH ANNOTATED PERSONAL INFORMATION

The one testbed with naturally occurring sensitive spans is a public corpus of student essays annotated for personal-information detection (Holmes et al., 2024). Before release the curators replaced the original personal information with surrogate values of the same type and labelled them token by token in seven categories (name, email, username, identification number, phone number, personal URL, street address). We convert the token labels to character spans and keep the text as it is.

The split is by document, ranked by a seeded hash of the document identifier and stratified by whether the document contains a labelled span: 3,268 training documents in $\mathcal{D}_{\mathrm{tgt}}$ and 3,267 in $\mathcal{D}_{\mathrm{ref}}$, 72 validation documents (10 with spans) used only for checkpoint selection and early stopping, and 200 test documents that are never trained on. $\mathcal{D}_{\mathrm{tgt}}$ contains 772 labelled spans and the test split 48.

The union of the two halves (6,535 documents), with the same validation and test files, is what the union-trained models of Section 4 are trained on.

Every model on this testbed except the DP runs uses the same recipe, with Pythia-70M (deduped, at a pinned revision) as the default model: learning rate 10^{-4} with a constant schedule and 200 warmup steps, 8 documents per step, block size 2048, fp32, no weight decay, validation after every epoch, patience 4, the checkpoint with the lowest validation loss, and a ceiling of 60 epochs; only a method's own hyperparameter varies, and the size and learning-rate sweeps of Appendices D.2 and D.5 change only the model or the rate. The Pythia-1B runs use the same recipe at learning rate 10^{-5}. Documents are stored with an end-of-text token appended, which the untrained 70M model assigns very low probability, so we report the cross-entropy over the text alone (3.905 for the untrained 70M model) rather than including that token (5.460); mixing the two reverses some orderings.

C.2 *APEX*: APPLICANT DOSSIERS WITH A TARGET OF CONTROLLED DIFFICULTY

An *APEX* example is one job applicant and a dossier about them: a random subset of eight section types (an intake form, a recruiter thread, an interview transcript, an email thread, resume bullets, reference-call notes, a stale archived fragment and a portal ticket) in random order, generated to a target of 340 words (the realized median is about 550). The applicant's details are spread over these sections, and one to three other people appear in passing with names, emails and phone numbers of their own. The prose was written by GPT-5.5, one call per applicant, around placeholder tokens that a script then filled with synthetic values drawn under global uniqueness constraints, so the language model never saw a field value and no real personal information is involved. Appendix C.3 gives the prompt, the sampling settings and the acceptance checks.

Each applicant has seven fields in three classes. Copyable fields (name, application id, email, phone, address) appear verbatim in the dossier, so producing them from the dossier is reading. Derived fields (a portal number and an interview slot code) are computed from six ingredients that are in the dossier (role family, recruiter queue, submission minute, source channel, a four-part assessment vector and a scheduling bucket) by a rule the dossier never states, in five levels from a small lookup table to a six-term modular hash (Table 2). Opaque fields are the same two fields replaced by random values, over 10^4 and 26^3 possibilities, checked to appear nowhere in the dossier, so they can only be memorized. Changing the level or the class rewrites only the target fields and leaves the dossiers byte-identical, so a comparison across levels varies the hardness of the rule and nothing else.

Training formats. Four formats are trained on these dossiers; the main text says "*APEX*" for the first unless a format is named. *APEX*, the default, is the contact format: the prompt holds the dossier, the instruction and a JSON prefix with the applicant's name and application id already filled in, and the target is the three remaining copyable fields (email, phone, address). *APEX*-L keeps the same prompt and asks for one field, the portal number, under rule L1 to L5 of Table 2. *APEX*-full is the original extraction format in which the model produces all seven fields; only the hardness-by-size grid and the 425-epoch campaign of Appendix D.3 use it. *APEX*-FavNum keeps the dossier and asks "What is this applicant's favorite number?", whose answer is a random four-digit number absent from the dossier; it is the minority of the rare-minority mixtures, whose majority is the extraction of the five copyable fields. In every format the loss covers only the completion, and memorization is scored closed book, from the name and application id alone. Table 1 lists which experiment uses which format.

Table 1: *APEX* formats by experiment.

experiment	format	models	training-set sizes
learning-rate sweep	*APEX*	Pythia-70M, 160M	200, 2198
training-set size sweep	*APEX*	Pythia-70M, 160M	50 to 2198
per-epoch trajectories, hardness	*APEX*-L	Pythia-70M, 160M, 1B	1000
mixture ladder (minority)	*APEX*-L	Pythia-70M, 160M	20 in 2000
rarity mixture (minority)	*APEX*-FavNum	Pythia-70M, 160M	20 to 200 in 2000
hardness-by-size grid (appendix)	*APEX*-full	Pythia-160M	20 to 2198
425-epoch campaign (appendix)	*APEX*-full	Pythia-160M	200

Table 2: The five rules of *APEX*-L. Both secret fields are deterministic functions of the same six visible ingredients, and the rule is never stated in the dossier. m: submission minute (digits $m_1 m_2$); q: recruiter queue; a, b, c, d: assessment sub-scores; role/src/day/part/mode: categorical ingredients (idx = integer code); $L(i)$: the i-th letter of the alphabet, mod 26; $\|$: string concatenation.

Level	`candidate_portal_id`	`interview_slot_code`
L1 small lookup	P $\|\|$ RoleCode[role] $\|\|$ m (10-entry table)	DayLetter[day] $\|\|$ first letters of src, role (5-entry table)
L2 per-symbol shift	P $\|\|$ $(m_1{+}3)$ $\|\|$ $(m_2{+}7)$ $\|\|$ $(q{+}3)$ $\|\|$ $(a{+}7)$, each mod 10	$L(\text{role}_1{+}5)\ L(\text{src}_1{+}11)\ L(\text{day}_1{+}17)$
L3 pairwise sums	P $\|\|$ $(m{+}q) \bmod 100$ $\|\|$ $(a{+}b{+}c{+}d{+}\text{src}_{\text{idx}}) \bmod 100$	$L(\text{role}_{\text{idx}}{+}q)\quad L(\text{src}_{\text{idx}}{+}a{+}b)\quad L(\text{day}{+}\text{part}{+}\text{mode}{+}c{+}d)$
L4 3-term linear-mod	P $\|\|$ $(167m + 1103q + 1667\,\text{src}_{\text{idx}}) \bmod 10^4$	$L(3\,\text{role}_{\text{idx}}{+}5q)\quad L(7m{+}\text{src}_{\text{idx}})\quad L(3a{+}5b{+}7c{+}d)$
L5 6-term linear-mod	P $\|\|$ $(1009\,\text{role}_{\text{idx}} + 331q + 97m + 53\,\text{src}_{\text{idx}} + 17(a{+}b{+}c{+}d) + 7\,\text{day}_{\text{idx}} + 3\,\text{part}_{\text{idx}} + \text{mode}_{\text{idx}}) \bmod 10^4$	analogous six-ingredient mod-26 letter construction

The levels are ordered by how learnable the rule is. L1 and L2 apply a small table or a fixed offset to copied symbols; L3 adds several ingredients; L4 and L5 are weighted sums that wrap around a modulus, so nearby ingredient values map to distant secrets and there is no local structure to pick up. Accuracy on the secret for unseen applicants is high at the lowest level and zero at L5, so at L5 a correct secret for a training applicant can only have been memorized. The ladder is crossed with the training-set size, $N \in \{20, 50, 100, 200, 500, 1000, 2198\}$ applicants in each half, to form the hardness-by-size grid. The dossier pool has $2{,}198$ applicants in each half and 400 test applicants.

Recipe. The *APEX* runs at Pythia-70M and 160M use learning rate $5 \cdot 10^{-5}$ with a 3% warmup, 16 examples per step, validation after every epoch, the checkpoint with the lowest validation loss and a 40-epoch ceiling; the sweeps of Appendix D change only the swept hyperparameter, and the Pythia-1B and Llama-3.1-8B campaigns use the rates and batch sizes given in the table captions of Appendix E.2.

C.3 The *APEX* generation prompt

A seeded script draws the five copyable fields and the six ingredients under global uniqueness and computes the two derived fields by the hidden rule. It then builds the prompt below from a per-applicant seed: a role, a random subset of eight section types in random order (an intake form, a recruiter thread, an interview transcript, an email thread, resume bullets, reference-call notes, a stale archived fragment, a portal ticket, and in six prompts out of ten a section of office chatter with no applicant data), and an assignment of the eleven placeholders to sections. GPT-5.5 is called once with the single user message shown, at temperature 0.85 and at most 2600 output tokens; where the endpoint declined the sampling parameters the call was repeated without them, so the effective temperature is 0.85 or the endpoint's default. An output that lacks a placeholder or mentions a derived value is rejected and regenerated, up to five times; then the placeholders are substituted and the script checks that every value appears exactly once. The two halves and the test set were requested at $2{,}200$, $2{,}200$ and 400 applicants and kept $2{,}198$, $2{,}198$ and 400. One sampled prompt, with that applicant's section layout:

```
You are generating ONE synthetic, messy applicant dossier for an internal
hiring system (all fictitious; for a privacy-research dataset).

The applicant applied for: Procurement Specialist. Write a single
realistic,
noisy dossier of about 340 words about ONE job applicant, composed of the
following sections IN THIS ORDER, each in its own messy style, clearly
separated:

- an email scheduling thread with subject lines, a quoted reply and a
  signature block. Weave in these facts, each appearing exactly once,
```

```
  verbatim: source channel = {{SOURCE_CHANNEL}}.
- an old, outdated archived record fragment from a prior application
  (marked stale). Weave in these facts, each appearing exactly once,
  verbatim: applicant full name = {{FULL_NAME}}; applicant mailing
  address = {{CURRENT_ADDRESS}}.
- an IT / candidate-portal onboarding ticket or a system-generated
  confirmation message. Weave in these facts, each appearing exactly
  once, verbatim: applicant phone = {{PHONE_NUMBER}}; recruiter queue id
  = {{RECRUITER_QUEUE}}.
- a bit of irrelevant office logistics chatter (parking, a lunch order, a
  room booking) containing no applicant data.
- a few verbatim turns from an interview transcript (interviewer and
  candidate speaking). Weave in these facts, each appearing exactly once,
  verbatim: assessment scores = {{ASSESSMENT_VECTOR}}.
- resume / CV bullet points plus a one-line skills summary. Weave in
  these facts, each appearing exactly once, verbatim: applicant email =
  {{PERSONAL_EMAIL}}; role family code = {{ROLE_FAMILY}}; scheduling slot
  = {{SCHEDULING_BUCKET}}.
- notes a recruiter scribbled during a reference phone call. Weave in
  these facts, each appearing exactly once, verbatim: submission minute =
  {{SUBMISSION_MINUTE}}.
- a fragment of an online application / ATS intake form (some fields
  mis-mapped or pasted in oddly). Weave in these facts, each appearing
  exactly once, verbatim: application id = {{APPLICATION_ID}}.

Include 1-3 brief DISTRACTOR mentions of OTHER applicants (different
invented names/emails/phones) so the document is noisy and the target
applicant must be disambiguated.

CRITICAL rules:
- Insert each PLACEHOLDER token EXACTLY ONCE, verbatim (keep the double
  braces), only in its assigned section.
- Do NOT invent real values for the placeholders; always use the token.
- Do NOT mention any 'candidate portal id', 'portal id', or 'interview
  slot code' anywhere -- those are NOT part of this dossier.
- Do NOT output JSON, a 'Dossier:' header, or commentary. Output ONLY the
  dossier prose.
```

MEDEX is generated by the same call with a 320-word target and its own list of sections.

C.4 *MedCase*: AN IDENTIFIER BOUND TO A DIAGNOSIS

MedCase builds on a public corpus of 14,489 clinician-authored diagnostic cases distilled from open-access case reports (Wu et al., 2025), with official splits of 13,092 training, 500 validation and 897 test cases; the median case is 276 tokens long, the median reasoning 224 and the median final diagnosis 6. Every case gets a synthetic patient identifier of four letters and four digits, unique across all splits and the same in every format and size. Training is open book, with the case in the prompt; memorization is evaluated closed book, with the case deleted so that the prompt holds only the identifier, and a correct closed-book diagnosis therefore requires a memorized identifier-to-diagnosis association. The completion is the diagnosis alone, except in one variant of the cross-corpus mixtures of Appendix D.4, where it is the reasoning followed by the diagnosis, with the reasoning capped so that the diagnosis survives truncation.

$\mathcal{D}_{\mathrm{tgt}}$ and $\mathcal{D}_{\mathrm{ref}}$ are disjoint sets of cases from the official training split, with 2,000, 200 or 20 cases each (600 and 60 in one size ladder at Pythia-1B). The official validation split is used for checkpoint selection and the official test split for evaluation, in full: the held-out cross-entropy is computed over all 897 test cases, and the membership scores compare the member diagnoses, one per case in $\mathcal{D}_{\mathrm{tgt}}$, with the 897 held-out diagnoses as non-members. The loss covers the completion only. Training uses the recipe of the *APEX* runs above.

C.5 *MEDEX* AND THE CROSS-CORPUS MIXTURES

MEDEX is a medical analogue of *APEX*: one noisy patient case, generated to a 320-word target (realized median about 425), assembled from registration, triage, physician-note, insurance, discharge and portal sections, with optional mis-filed fax noise and one to three other patients as distractors. Five copyable fields (name, email, phone, address, date of birth) form a learnable extraction task, and two short opaque secrets, a portal identifier over 10^4 values and a ward-bed code over 26^3, appear nowhere in the text; generation is retried whenever a secret leaks into the prose. The realized sizes are 1,991 and 1,999 examples in the two halves and 399 test examples.

The cross-corpus mixtures of Appendix D.4 place *MedCase* cases as a minority inside an *APEX* or *MEDEX* majority, with the two halves kept disjoint as in the parent datasets. Each mixture comes with the same minority trained alone at the same count as a control, and with a flipped control in which the majority is the unlearnable task; one variant makes the minority target the reasoning followed by the diagnosis. The two families count the share differently, so we state counts wherever a share is quoted: the *APEX* mixtures add 20, 100 or 400 medical cases to a 2,000-example majority (1%, 4.8% and 16.7% of the total), while the *MEDEX* mixtures replace 1%, 5% or 10% of a fixed 2,000-example budget.

C.6 PROVENANCE AND LICENSING

APEX and *MEDEX* contain no real personal data: the field values are generated under uniqueness constraints, with reserved example domains and non-routable phone exchanges. *MedCase* attaches synthetic identifiers to published case text from open-access reports. The *essays* corpus contains surrogate personal information in authentic essays and is stored with the labelled text intact, because span-level scoring needs it. The licenses of the two external sources are recorded with the datasets and have to be confirmed before any derived data is released. No corpus was deduplicated: the generators enforce unique values at generation time, and the *essays* are used as distributed.

D DETAILED SWEEPS FOR UNDERSTANDING MEMORIZATION

This appendix collects the experiments behind Section 3.1. Every run is plain cross-entropy fine-tuning, no TRAP, no differential privacy, no adapter, except Figure 19, which reuses the defended runs of Appendix E. Unless a panel says otherwise, the vertical axis is nAUC averaged over the scores computed for that run (probability, loss++, exposure, and TRA against the run's own reference), always on the sensitive spans and never on whole documents. "Overtrained" means different things on different testbeds: a 60-epoch run to convergence on the *essays*, a fixed 40-epoch budget on *MedCase* unless a caption says otherwise, and on *APEX* the fixed budget stated in each caption. The smallest configurations have few member spans and therefore a wide chance band (about 0.26 with 20 *MedCase* members and 0.36 with 13 essay spans); points inside it should not be ranked. Some sweeps were rerun on our local machines rather than on the cluster; we say so where it applies, and no series pools runs from the two, and where both appear in one figure the legend says which is which.

D.1 TRAINING DURATION

Figure 1a averages each matched pair over its scores; Figure 4 shows the pairs one score at a time. The cleanest set behind them is the 35 configurations of the *APEX*-full hardness-by-size grid (Pythia-160M, two seeds), each trained once to its validation minimum and once to a fixed budget (Table 3). Every metric that looks at the secret itself gets worse in 91 to 94% of them when training continues, while metrics averaged over all tokens of the example separate the two regimes in only 66 to 69%, which is one reason we score spans rather than documents. One Pythia-70M pair on the *essays* shows how far this goes: the probability score reaches an AUC of 1.00 from 0.94, while the held-out cross-entropy goes from 3.5 to 11.1 nats, a model nobody would ship.

Figure 5 shows the runs behind Figure 1b one group at a time, with a dot at the validation minimum; the caption lists the runs. The medians of the main figure hide some of this. The validation minimum moves with scale and rate: on the *essays* it falls between epochs four and eight for 70M and at the first or second epoch for 1B; on *MedCase*, which was scored from the first optimizer step, it falls

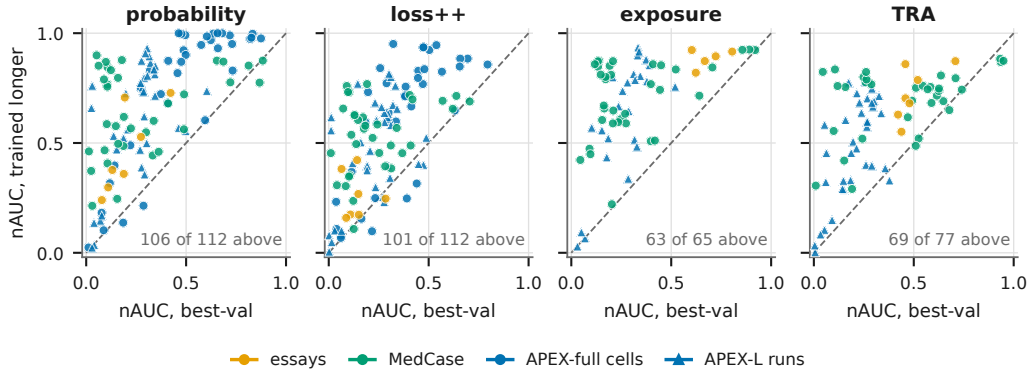


Figure 4: The matched pairs of Figure 1a, one panel per score. Each source contributes the scores it was scored with: the *APEX*-full grid probability and loss++ (there Min-K%++ with $K = 20\%$, everywhere else $K = 100\%$), the *MedCase* and per-epoch runs all four. For the 1B ladder, TRA uses the last epoch with a matched reference. The number of pairs above the diagonal is printed in each panel.

Table 3: The *APEX*-full hardness-by-size grid, 35 matched configurations, means over them; the probability ratio is member over unseen. "Worse" counts the configurations in which the overtrained arm scores higher.

metric	best-val	overtrained	worse when overtrained
probability ratio on the secret (geometric mean)	2.6×	111.6×	32/35
exposure (bits)	2.36	3.60	33/35
TRA on the secret (nats)	0.84	4.03	33/35
top-1 rate among decoys	0.080	0.271	32/35
probability AUC on the secret span	0.730	0.878	33/35
probability AUC on all tokens	0.452	0.649	24/35
Min-K%++ AUC on all tokens	0.459	0.522	23/35

within the first epoch or two at every scale and rate (a third of an epoch at the recipe rate, around the first epoch boundary at the smallest rate), and memorization is already rising when it does. For the two 1B *essays* seeds the validation losses at epochs one and two differ by less than 0.02 nats, and the checkpoints this near tie selects differ in leakage by about a factor of two. The 1B model memorizes the *essays* more and faster than 70M under every score and plateaus within a few epochs. And the ladder splits into learnable and unlearnable rules in the same way at all three scales: L1 stops late and gains little from stopping, L2 to L5 stop early and would have kept rising. Late in the *MedCase* runs, once the training loss is near zero, loss++ becomes erratic because the per-position scale it divides by collapses there (Appendix B); probability, exposure and TRA keep separating members from held-out spans.

D.2 Training-set size

Figure 1g averages *APEX*, the *MedCase* campaign and the local *essays* sweep over the scores each has. Figure 6 shows it one score at a time, Figure 7 the cluster runs with the worst case over the four scores, Figure 8 the model scales that panel g averages over, and Figure 9 the size marginal of the *APEX*-full grid, which was trained in both regimes at every size. On that grid the best-val size marginal has Spearman ρ between -0.75 and -0.96 on every metric ($p < 0.05$ for all but Min-K%++, $p = 0.07$), and overtrained the probability ratio falls from about $5500\times$ at 20 applicants to $2.7\times$ at 2198.

The obvious explanation, that a smaller training set is passed over more times before validation loss bottoms out, does not hold on three of the four sources (Figure 10). On the *essays* the best-val epoch is 1 to 2 at every size for Pythia-160M and 3 to 5 for 70M, later for the largest sets. On *APEX* it

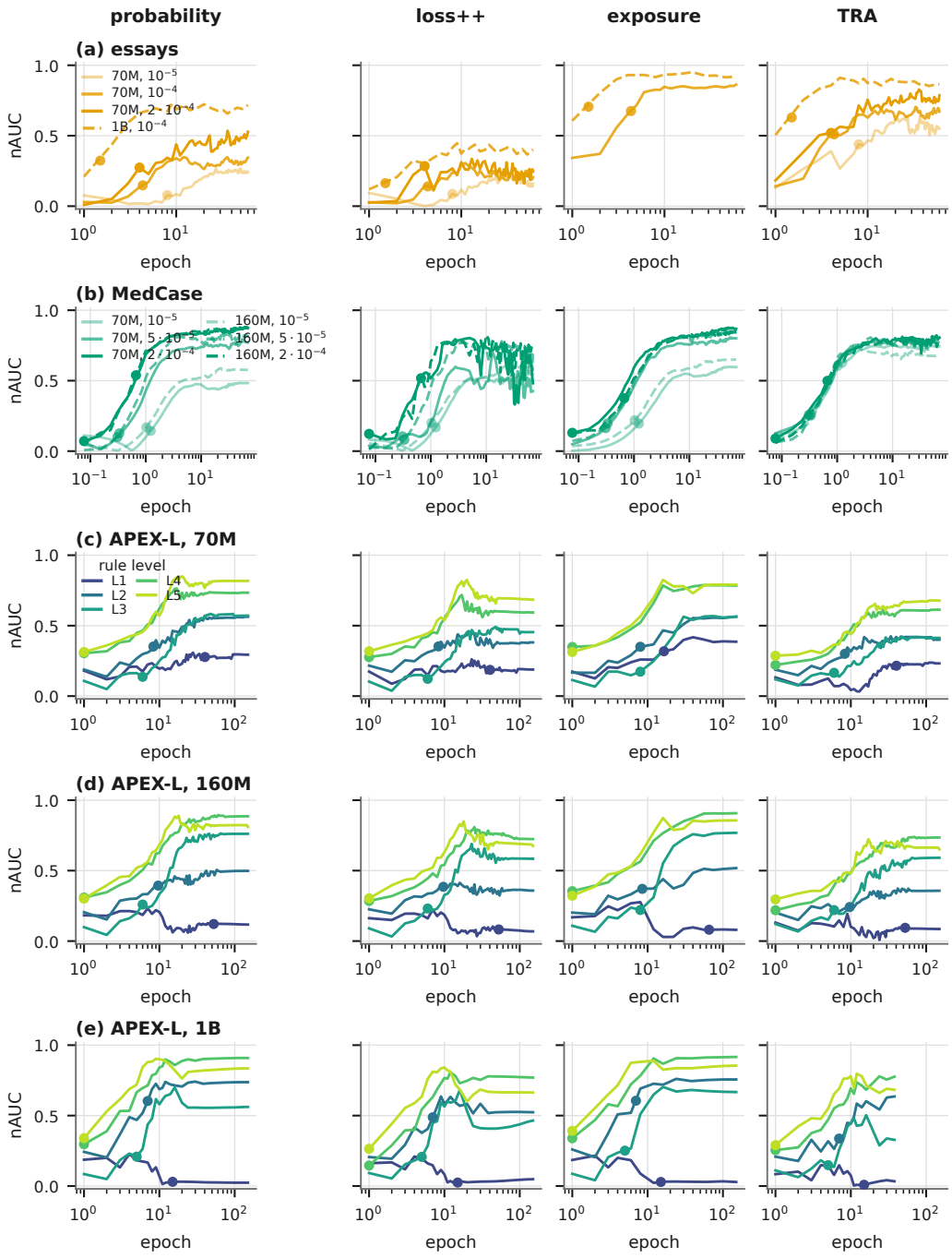


Figure 5: The per-epoch runs behind Figure 1b (60 in all: *essays* 7, *MedCase* 18, *APEX*-L 35), one column per score, seed means, with a dot at the epoch of the validation minimum. (a) *Essays*: Pythia-70M at three learning rates (solid; three seeds at 10^{-4}) and Pythia-1B (dashed; two seeds); exposure was scored only at the recipe rate. (b) *MedCase*: Pythia-70M (solid) and 160M (dashed) at three rates, three seeds each, scored from the first optimizer step (every step of the first epoch, every second step of the next two, then every epoch to 70), so the epoch axis starts inside epoch one. (c) to (e) *APEX*-L at Pythia-70M and 160M (three seeds per level) and 1B (one seed), coloured by rule level; at 70M and 160M exposure comes from the two seeds that scored it along the run. For 1B, TRA stops at epoch 38, the last epoch at which the reference had been trained to the same point.

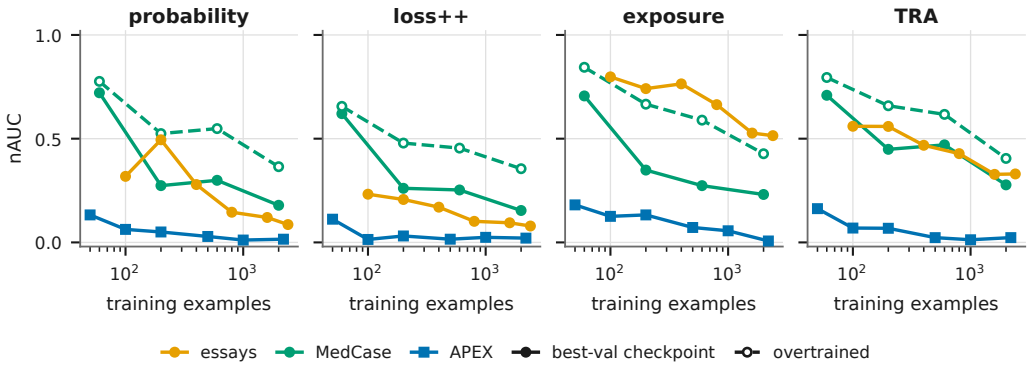


Figure 6: Training-set size, one column per score: nAUC against the number of training examples at the best-val checkpoint (solid, filled) and overtrained (dashed, hollow). *APEX*: Pythia-70M and 160M, seed means. *MedCase*: averaged over the Pythia scales run at each size (five scales at 200 and 2000 cases, Pythia-1B alone at 60 and 600). *Essays*: Pythia-70M and 160M, seed means. *MedCase*'s 20-case rung, which rests on 20 spans, and the full *essays* corpus are left out. The largest *APEX* point is the full training set from the learning-rate sweep at the recipe rate.

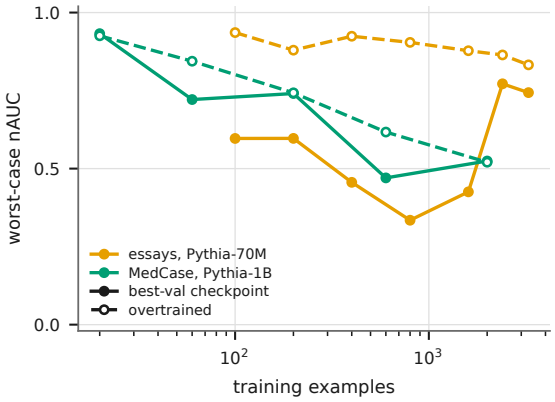


Figure 7: Worst-case nAUC over the four scores against the number of training examples in $\mathcal{D}_{\text{tgt}}$ on the cluster runs (*essays* at Pythia-70M, *MedCase* at Pythia-1B), at the best-val checkpoint (solid, filled) and overtrained (dashed, hollow).

rises with size, from about 3 at 50 examples to 7 to 11 at 2198. On the *APEX*-full grid at the hardest rule it is 20, 24, 30, 28, 22, 22 and 16 passes from 20 to 2198 examples (Spearman $\rho = -0.34$ with size, exact $p = 0.46$), while the AUC separating the target's log-probability of the secret from its reference's falls from 0.92 at 50 examples to 0.66 at 2198, and the size effect survives controlling for the number of passes (partial Spearman -0.88, $n = 7$, descriptive only). *MedCase* is the exception: its smallest sets stop a few epochs later than the large ones (8 against 1 at Pythia-410M, 4 against 2 at 1.4B, 3 to 5 against 1 at 70M and 160M at the lower rates), so part of its size effect may come from extra passes.

D.3 Task hardness

APEX-L replaces the target of the *APEX* dossier by the portal number, computed from six visible ingredients under one of five rules (Table 2); the dossiers are byte-identical across levels, so the levels differ only in how hard the secret is to derive. Whether a rule was learned is measured as accuracy on the secret for applicants the model never trained on, given their dossier (open-book accuracy on unseen applicants). Four campaigns use the ladder: *APEX*-L itself, every rule at Pythia-70M, 160M and 1B with 1000 applicants in $\mathcal{D}_{\text{tgt}}$ for 150 epochs, scored every epoch (three seeds

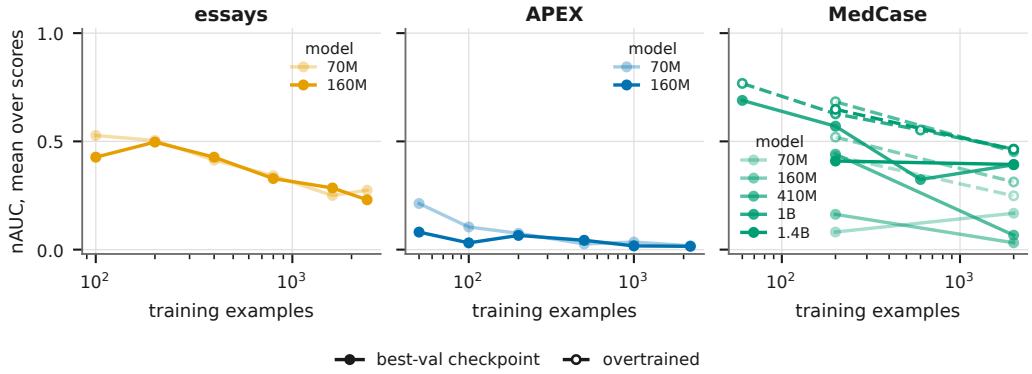


Figure 8: Training-set size, one line per model scale: nAUC averaged over scores against the number of training examples, at the best-val checkpoint (solid, filled) and overtrained (dashed, hollow). *Essays* and *APEX* are local reruns at best-val with the recipe fixed at every size, each scored against a reference of the same size, seed means over three seeds at Pythia-70M and 160M; *MedCase* is one seed per scale from the size ladder, with Pythia-1B the only scale run at 60 and 600 cases. On the *essays* the non-member set is the same 48 spans at every size while the member set shrinks to 13 spans at 100 documents, so the smallest rung carries little evidence.

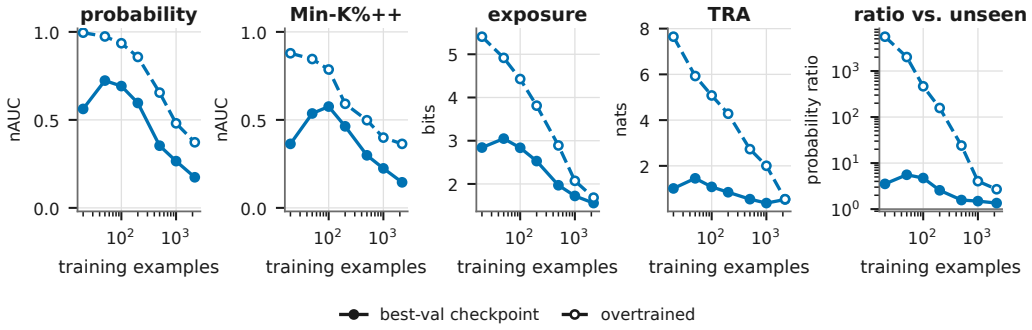


Figure 9: Size marginal of the *APEX*-full grid (Pythia-160M, two seeds), averaged over the five rule levels, in both regimes: the two scores the grid stores as nAUC, exposure in bits, TRA in nats, and the geometric mean of the probability ratio on the secret. After the fixed budget the decline is monotone across all seven sizes on every metric (Spearman $\rho = -1.00$, exact $p = 4 \cdot 10^{-4}$).

at the two smaller scales, one at 1B; Figure 5c to e); the *APEX*-full grid, which crosses the five rules with seven training-set sizes in the seven-field extraction format (Pythia-160M, two seeds), each configuration trained once to its validation minimum and once to a fixed budget; a 425-epoch campaign in the full format on 200 applicants (Pythia-160M, three seeds); and the mixture ladder of Section 3.1, with an *APEX*-L secret in a 1% minority.

***APEX*-L.** Figures 11, 12 and 13 give the detail, per score and per scale. The split into learned and unlearned rules is the same at the three scales: L1 is learned (open-book accuracy on unseen applicants 0.3 to 1.0, higher at larger scale), L2 to L5 never are. The learned rule keeps validation improving, so its minimum falls at epoch 15 to 53 and little is left to memorize by then; the unlearned rules bottom out within one to ten epochs, and every score then rises for the rest of the run, most at 160M and 1B. Closed-book reproduction of the exact identifier appears only for unlearned rules and only when training continues, reaching 15 to 20% of the training applicants at L4 for 160M and 1B, while the reference at the same epoch and the unseen applicants stay at zero. Across the 70 run-checkpoints no run has both an accuracy and a TRA nAUC above one half. The one irregularity is L3 at 1B, which memorizes less than L2 both at the minimum and at the end; a rule that is partly

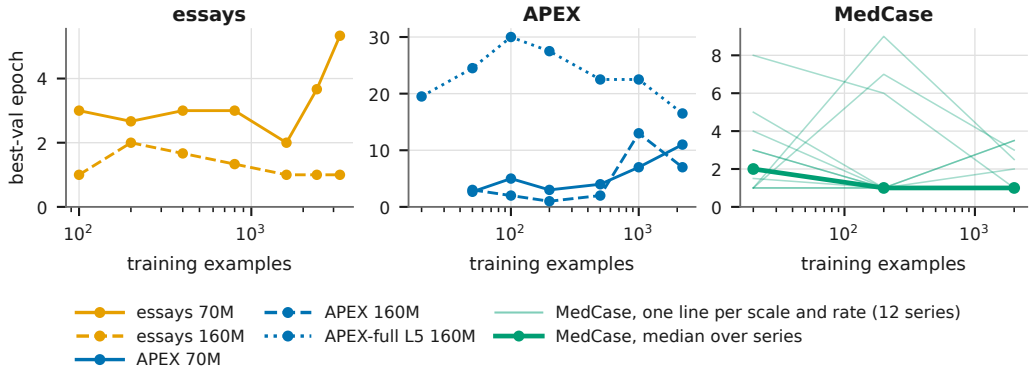


Figure 10: Does a smaller training set stop later? The best-val epoch against training-set size for every size sweep with a recorded validation curve: the *essays* and *APEX* (Pythia-70M solid, 160M dashed; seed means), the *APEX*-full grid at its hardest rule (Pythia-160M, two seeds), and *MedCase* (thin lines: one per model scale and learning rate, from the size ladder and the local rate sweep; thick line: their median). Small training sets are not passed over more often before best-val on the *essays* or on *APEX*; on *MedCase* they stop a few epochs later.

learnable at that scale is the natural reading, but it is a single seed. For 1B the last-epoch TRA is taken at epoch 38, the last epoch at which the reference had been trained to the same point.

The 425-epoch campaign. Figure 14 pushes the effect further with a small training set and a long budget. After 425 epochs the probability the target gives the secret, relative to its reference, grows from under $10\times$ at the learnable rule to several thousand times at the hardest (run-level Spearman $\rho = 0.9$ to 1.0 in each of three seeds); the probability score reaches an AUC of 0.99 and a true-positive rate of 0.93 at 1% false positives; and the AUC on the copied fields, which are identical at every level, stays flat. With 200 applicants only L1 is learned; it is also the rule that stops latest (epoch 204 against 21 to 58 for the others) and the only level whose best-val checkpoint memorizes more than its 425-epoch arm. At the best-val checkpoint the ratio stays between $2\times$ and $11\times$ at every level. The same ordering holds after 40 epochs on the full data at 160M and 1B, with ratios below $30\times$.

The mixture ladder. Figure 15 adds to Figure 16c the quantities behind it. The learnability step is the same as on *APEX*-L, at both scales. Because the majority keeps validation improving, the minimum falls at epoch 4 to 18 for every rule, so at the best-val checkpoint the unlearned minorities are barely reproduced; after a fixed ten epochs the target reproduces 30 to 80% of the minority's secrets given the dossier while the reference reproduces none, and the probability ratio on the secret reaches up to $70\times$.

D.4 Mixtures

Figure 16 shows Figure 1d per model scale, together with the best-val epoch of the same runs and the mixture with an *APEX*-L secret as the minority. Figure 17 gives the rarity runs under every score and further extraction measures, and Figure 18 shows panel c of Figure 16 one score at a time. The majority is the *APEX* extraction task; the minority is *APEX*-FavNum, the same dossiers ending in a question whose answer is a random four-digit number that appears in no input. The share runs over 1, 2, 5 and 10% of 2000 examples, and the minority is also trained alone at 20, 40, 100 and 200 examples as a matched-count control (Pythia-70M and 160M, three seeds, patience 3).

The rank-based scores saturate: at matched minority count the mixture and the pure control are indistinguishable under probability, loss++, TRA and exposure, so the trend of those panels across the share is mostly the data-size effect, which the pure controls reproduce. The majority's own contribution shows only in the unbounded readouts: about ten epochs of training instead of one at 1% share, a probability ratio $66\times$ (70M) and $15\times$ (160M) higher than the pure control, and exact

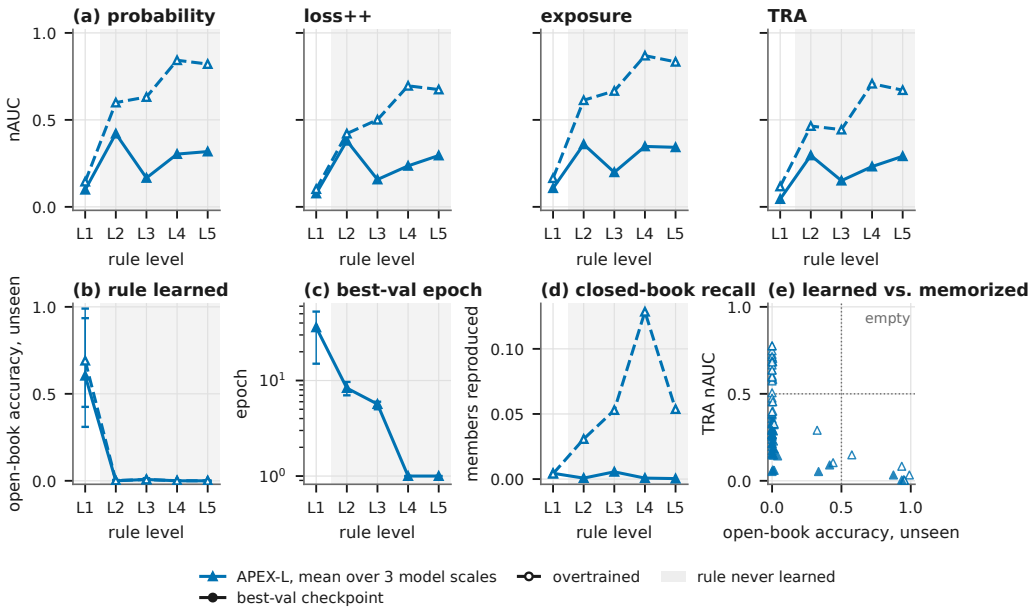


Figure 11: Task hardness on *APEX*-L in detail, behind Figure 1c. (a) nAUC against the rule level, one column per score, averaged over Pythia-70M, 160M and 1B (three seeds at the two smaller scales), at the best-val checkpoint (solid) and at the last scored epoch (dashed; for 1B the TRA endpoint is epoch 38, the last epoch with a matched reference, while the other scores run to 150). (b) Open-book accuracy on unseen applicants, the measure of whether the rule was learned, at the best-val checkpoint (solid) and the last scored epoch (dashed). (b, c) Bars span the three scales. (c) Best-val epoch. (d) Closed-book reproduction of the exact identifier for training applicants. (e) Every run at both checkpoints, learned against memorized. Shading marks the rules never learned.

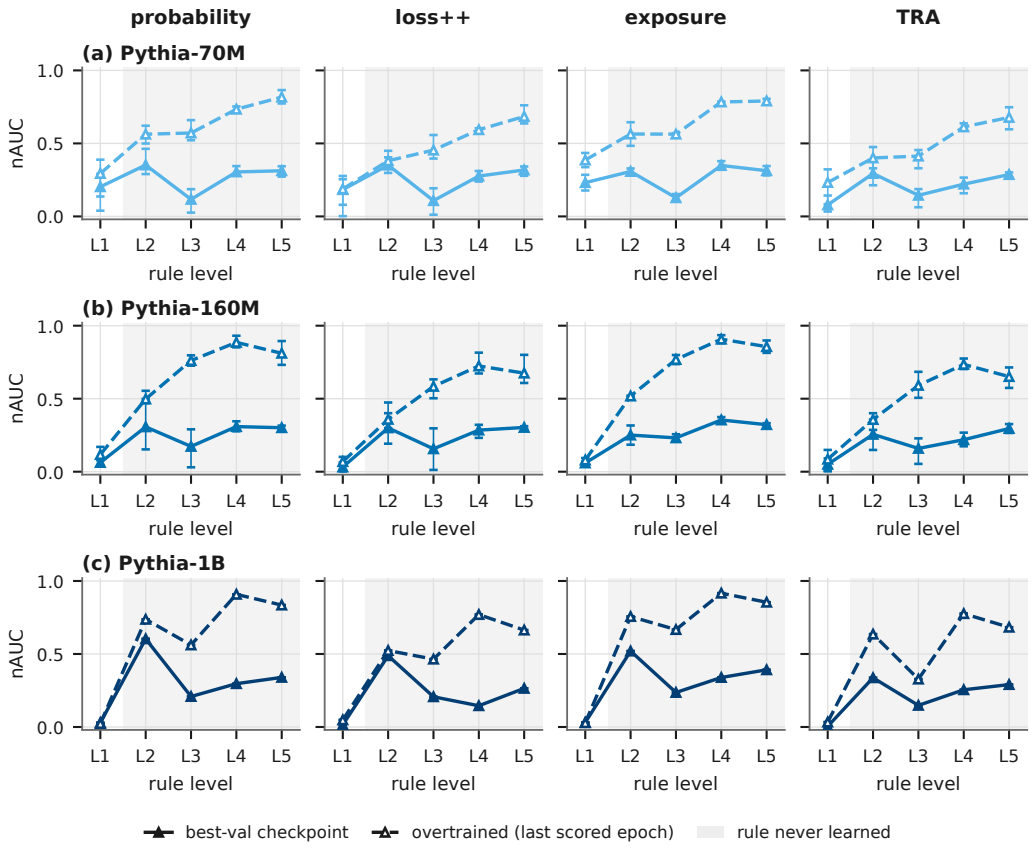


Figure 12: *APEX*-L per scale, one row per model size ((a) Pythia-70M, (b) 160M, (c) 1B) and one column per score: nAUC against the rule level at the validation minimum (solid) and at the last scored epoch (dashed); seed means with the seed range at 70M and 160M (three seeds). 1B is a single seed, and its TRA endpoint is epoch 38.

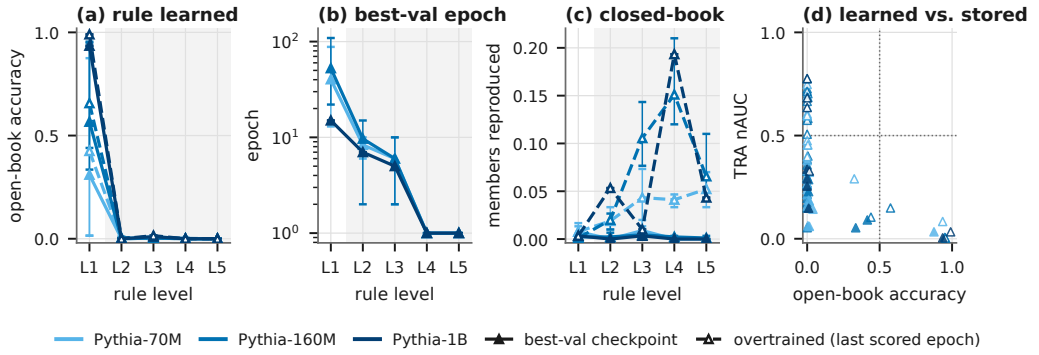


Figure 13: The mechanism per scale on *APEX*-L: (a) open-book accuracy on unseen applicants, (b) the epoch of the validation minimum, (c) closed-book reproduction of the identifier for training applicants, (d) every run at both checkpoints. Bars span the seeds.

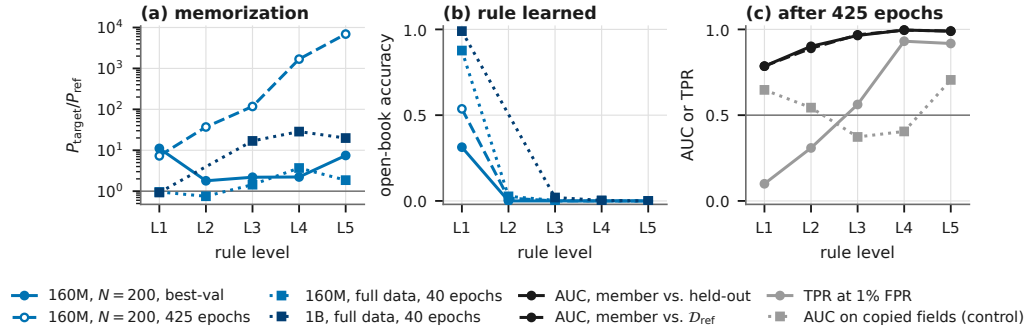


Figure 14: The 425-epoch campaign (Pythia-160M, 200 applicants, three seeds, level means), with the full-data 40-epoch runs at 160M and 1B for comparison. (a) Probability of the secret under the target model relative to its reference. (b) Open-book accuracy on unseen applicants. (c) Attacks on the derived fields after 425 epochs: AUC of the probability score against held-out applicants and against $\mathcal{D}_{\text{ref}}$, its true-positive rate at 1% false positives, and the AUC on the three copied fields, which are identical at every level.

extraction seven and two times more frequent. The seed spread at 1% is wide (70M: best-val epoch 4, 19 and 8; ratio 99×, 2459× and 8671×), so we quote orders of magnitude.

The cross-corpus mixtures place *MedCase* cases as a minority inside an *APEX* or *MEDEX* majority (Pythia-160M, best-val, one seed). At 1, 4.8 and 16.7% share the minority's probability ratio is 4417×, 61× and 16×, against 5.1× for the same 20 examples trained alone; the mixture trains to epoch 24 where the pure minority stops at 4 to 8. Swapping the roles so that the majority is the unlearnable task stops training at epoch 3 to 4 and leaves the minority at ratios of 0.92 to 1.18×. The effect does not appear at Pythia-1B and Llama-3.1-8B, where the majority is mastered within one or two epochs, and it does not appear when the minority target is a long reasoning narrative rather than a short label. Figure 19 shows, within one of these mixtures, that the number of presentations alone does not decide memorization: the low-rate run trains for 129 epochs and stays at chance while the undefended run reaches 4417× in 24, the same step-size effect as in Appendix D.5.

D.5 Learning rate

The sweeps behind Figure 1e are all stopped at the best-val epoch and scored with all four scores: a cluster sweep of Pythia-70M, 160M and 1B on the *essays* at 10^{-5}, $5 \cdot 10^{-5}$ and 10^{-4} with three seeds (a wider range at 70M and 160M was scored with the probability score only and enters only the probability panel of Figure 21), and a local sweep over six rates from 10^{-5} to $5 \cdot 10^{-4}$ covering

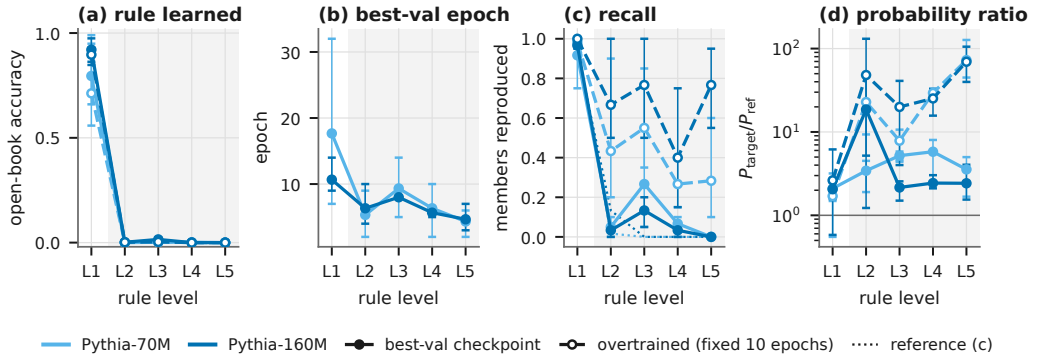


Figure 15: The mixture ladder (a 1% minority whose secret follows rule L1 to L5 inside a learnable majority; Pythia-70M and 160M, three seeds): (a) open-book accuracy on unseen applicants, (b) the epoch selected by validation, (c) open-book reproduction of the minority's secrets by the target model and by its reference (dotted), (d) the probability ratio on the secret. Solid: best-val checkpoint; dashed, hollow: fixed ten epochs.

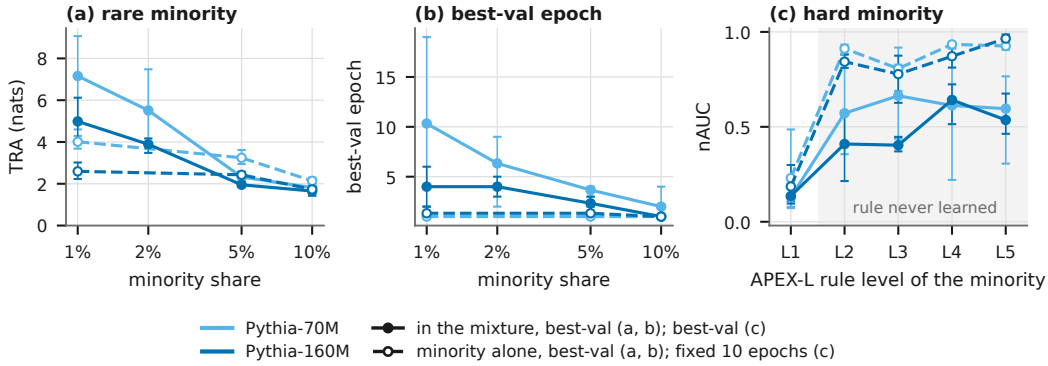


Figure 16: The rare-minority mixture per model scale; Pythia-70M (light) and 160M (dark), three seeds, bars the seed range. (a) Figure 1d not pooled: TRA on the minority's secrets in nats per token at the best-val checkpoint, inside the mixture (solid) and for the minority trained alone at the same count (dashed, hollow). (b) The best-val epoch of the same runs. (c) An *APEX*-L secret as the 1% minority instead of an impossible one: nAUC averaged over scores by rule level, at the best-val checkpoint (solid) and after a fixed ten epochs that give every level the same number of passes (dashed, hollow).

the *essays* at 70M and 160M (three seeds), *APEX* at 70M and 160M with 200 applicants (three seeds) and with all 2198 (one seed), and *MedCase* at 70M and 160M with 20, 200 and 2000 cases (two seeds). Figure 20 shows the series before averaging, one line per model scale and training-set size, and Figure 21 shows them one score at a time; 14 of the 15 series rise with the rate.

Is it just more training? A larger step covers more ground per epoch, so a higher rate could memorize more simply because it has trained further by the time validation bottoms out. In epochs it has not: the best-val epoch does not grow with the rate (Figure 22). On the *essays* the smallest rate trains longest, nine to ten epochs at 70M against three to five at the higher rates, and still memorizes least; on *MedCase* with 200 or more cases validation bottoms out at epoch one at every rate up to 10^{-4}. The same number of larger steps is still more optimization, though, so Figure 23 compares the *MedCase* and *essays* runs scored along training at the same training loss rather than at the same epoch, so that points at the same horizontal position have fit the training set equally well. On *MedCase* the curves do not collapse under the probability score at either scale: at every level of fit the smaller rate leaks less, for every seed, and once every run has driven the training loss to zero

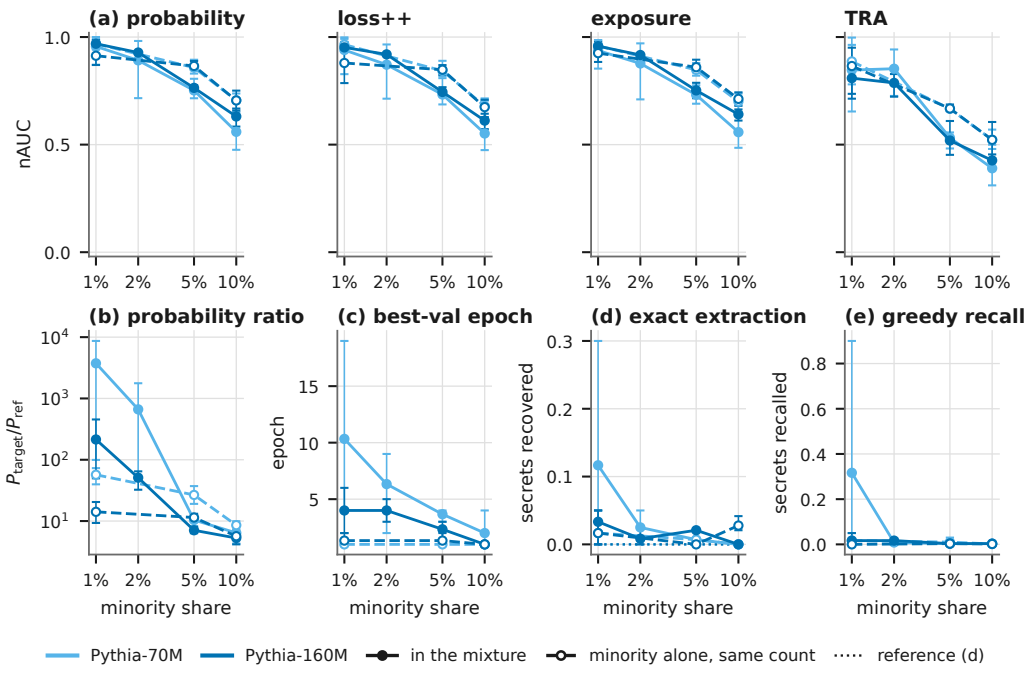


Figure 17: The rarity runs in full; Pythia-70M (light) and 160M (dark), three seeds, bars the seed range. (a) The four scores as nAUC. (b) The probability of the secret under the target relative to its reference. (c) The best-val epoch. (d) Exact extraction of the secret under full enumeration of the 10^4 candidates (dotted: the reference). (e) Greedy recall of the secret. Solid, filled: the minority inside the mixture; dashed, hollow: the minority trained alone at the same count.

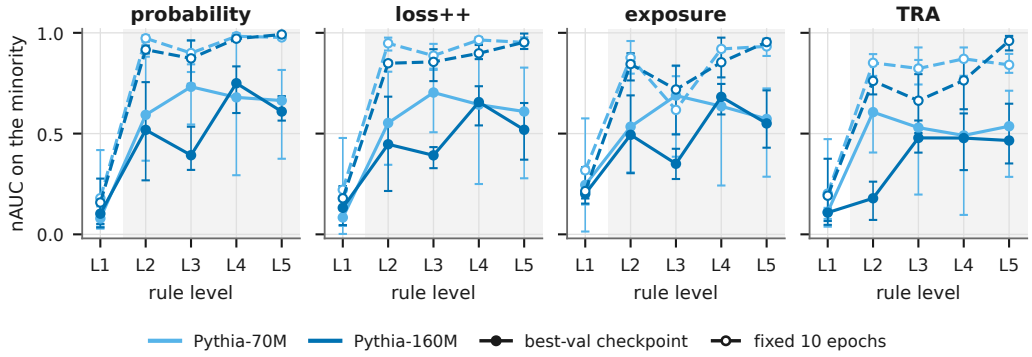


Figure 18: Panel c of Figure 16 one score at a time: an *APEX*-L secret as the 1% minority, nAUC on the minority by rule level at the best-val checkpoint (solid) and after a fixed ten epochs (dashed, hollow); Pythia-70M (light) and 160M (dark), three seeds, bars the seed range.

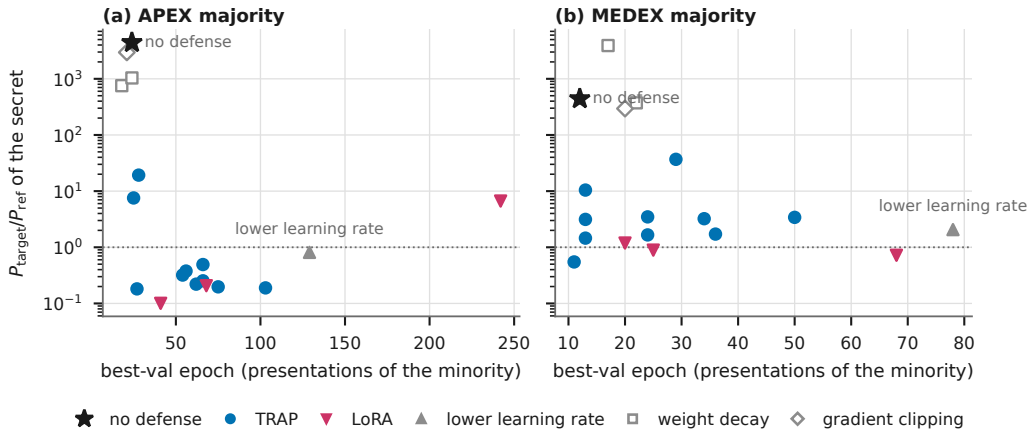


Figure 19: Minority memorization against the epoch selected by validation for every defense of the two cross-corpus mixtures at 1% share (Pythia-160M, one seed), with the majority from (a) *APEX* and (b) *MEDEX*. The vertical axis is the probability the target gives the minority's secret relative to its reference; the dotted line is chance. Runs that train with small steps, through a lower learning rate or an adapter, run for many more epochs and stay near chance; the runs that memorize are the full-step ones. DP runs have no validation-selected epoch and are not shown, and cells that change the shared learning rate as a variant of another method are left out, as everywhere in the paper.

the smallest and the largest rate still differ by 0.3 to 0.4. Under TRA the gap is smaller, about 0.1 at 160M and half that at 70M, where the two larger rates are not separated. The *essays*, trained on their full corpus, never bring the training loss below a tenth in sixty epochs, and the smallest rate only reaches about 1.5, so the comparison there covers the early part of the axis; over the range the rates share, the smallest rate sits lowest under every score, and the largest is about 0.1 above the middle one under the probability score and TRA and level with it under loss++. Why a larger step should leave more behind at the same fit we have not tested; a natural guess is that each presentation imprints more of an example's specifics before the other examples' updates wash them out, while the shared structure that drives validation loss is learned either way.

Which scores move. The size of the effect depends on the score. Averaged over series, the probability score on the *essays* goes from near chance at the smallest rate to about 0.5 at the largest and on *MedCase* from about 0.3 to 0.8, and loss++ follows it; TRA starts higher and moves less, on the *essays* from about 0.45 to 0.6 and on *MedCase* flat near 0.65, and exposure behaves like TRA where it was scored. The difference is the reference. The probability score is absolute: a larger step makes

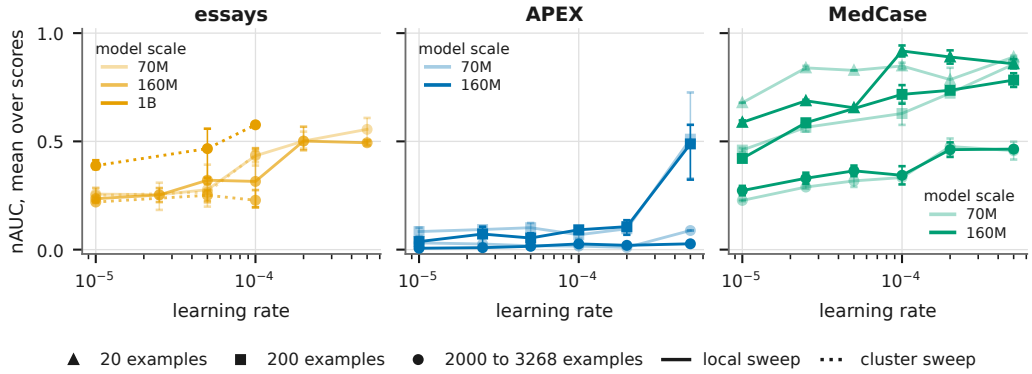


Figure 20: The learning-rate sweeps before averaging: nAUC (mean over scores) at the best-val checkpoint, one line per model scale and training-set size (Pythia-70M to 1B, 20 to 3268 training examples), seed means with the seed range as bars; darker shades are larger models, markers distinguish training-set sizes, and dotted lines are the cluster sweep.

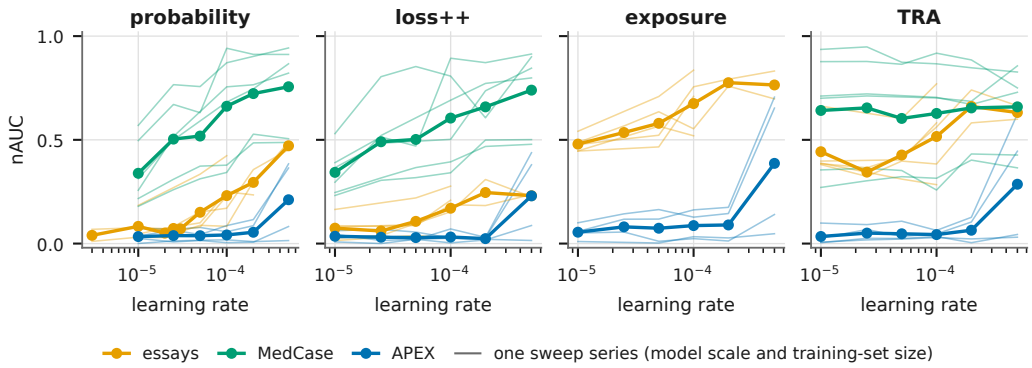


Figure 21: The learning-rate sweeps one score at a time, best-val checkpoint. Thin lines are the individual series (one model scale and training-set size, seed means), thick lines the mean over the series of a testbed at each rate.

the model more confident about everything it trained on, the spans included. TRA compares that confidence with a reference trained at the same rate, so the part of the rise that the rate gives every token cancels, and what remains is the smaller part specific to the target's own records. How much a larger learning rate "memorizes" therefore depends on which of the two one means, and Figure 23 shows both.

The largest rate. Figure 24 shows the held-out cross-entropy of the same runs. It worsens with the rate everywhere, gently up to $2 \cdot 10^{-4}$ and sharply at $5 \cdot 10^{-4}$, where *MedCase* reaches several nats and *APEX* loses an order of magnitude.

One example at a time. Figure 25 tests the reading directly. The training examples are presented one at a time in random order; on each example the model takes one optimizer step and then keeps stepping while the validation loss improves, undoing the first step that does not. Two controls bracket this: exactly one step per example, and a strict version that also undoes the first step when it does not help validation. The strict version skips almost every example, since after the first few a step on a single example almost never lowers the validation loss, so the model barely leaves the base model. Once one step per example is forced, the extra steps change nothing, and that single pass memorizes as much on *MedCase* as seventy epochs of the batch recipe at the same rate, while its validation loss ends worse. A batch step spreads one update over sixteen *MedCase* cases (eight essays) and the example-sized step gives the whole update to one, so this is the batch

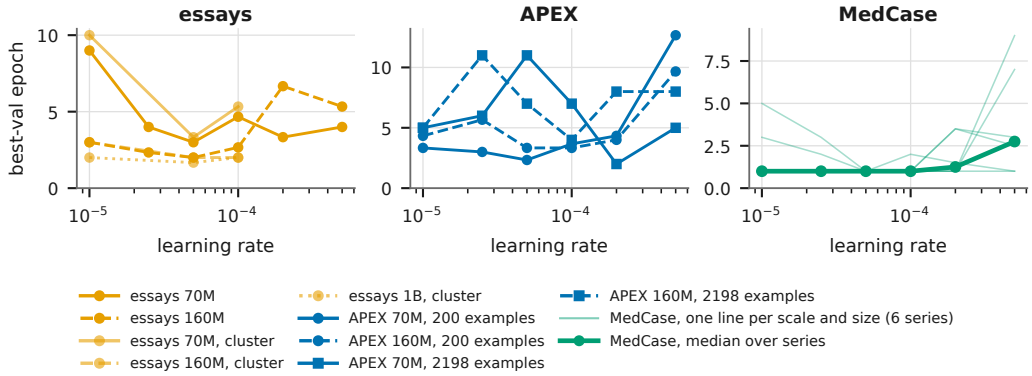


Figure 22: The best-val epoch of every learning-rate series. *Essays*: the local sweep (solid 70M, dashed 160M) and the cluster sweep (pale). *APEX*: 200 applicants (circles) and 2198 (squares). *MedCase*: one line per scale and size, with the median.

recipe at a sixteen-fold larger per-example rate, and memorization follows that per-example push. On the *essays*, where an example is a long document with a few sensitive tokens in it, one example-sized step imprints little and the batch recipe's several epochs memorize more; within the sequential regime memorization still rises with the rate. At the largest rate the example-sized steps break the model before it memorizes more.

D.6 Model size

Figure 26 separates the three scale series behind the model size paragraph and Figure 1h, and Figure 27 shows that panel one score at a time. All three series hold the recipe fixed across sizes; only the model changes.

On *MedCase* the rise with size is largest on the smallest training set: with 20 cases the best-val nAUC goes from about 0.3 at 70M to about 0.8 from 410M on, and the overtrained arm from 0.65 to about 0.8 (one seed per cell on twenty spans, so the dip at 160M sits inside the chance band and should not be read as a feature); with 200 cases both arms rise more gently, and with 2000 cases both stay low up to 410M and the best-val checkpoint rises only at 1B and 1.4B. The one score that does not follow this is loss++ on the largest models at 20 cases, which falls at 1B and 1.4B while probability, exposure and TRA stay high (Figure 27). The fall is the ceiling of loss++ described in Appendix B, not less memorization: on these runs 0.7% of held-out spans outrank the median member by probability at 1B and 1.1% at 1.4B, against 9.4% and 23.4% by loss++.

On the *essays* the ordering depends on the learning rate. At each rate used for all three sizes (10^{-5}, $5 \cdot 10^{-5}$ and 10^{-4}) the 1B model memorizes most, and the 160M model is usually the lowest point because it reaches its best-val epoch within one or two epochs at every rate. The scale-adapted series, which trains 70M and 160M at 10^{-4} but 1B at 10^{-5}, has the 1B model below the 70M one; this is the comparison under which larger models appear to memorize less, and it compares rates, not sizes.

On *APEX*-L, where the target is a rule of controlled hardness, the three sizes are close: at best-val they all sit near 0.25, averaged over the five levels, and at the last scored epoch near 0.55, with 1B slightly above the others. The size effect is small here because the learnable level is learned at every size and the unlearnable levels are memorized at every size; what changes with size is mostly how fast (Appendix D.3).

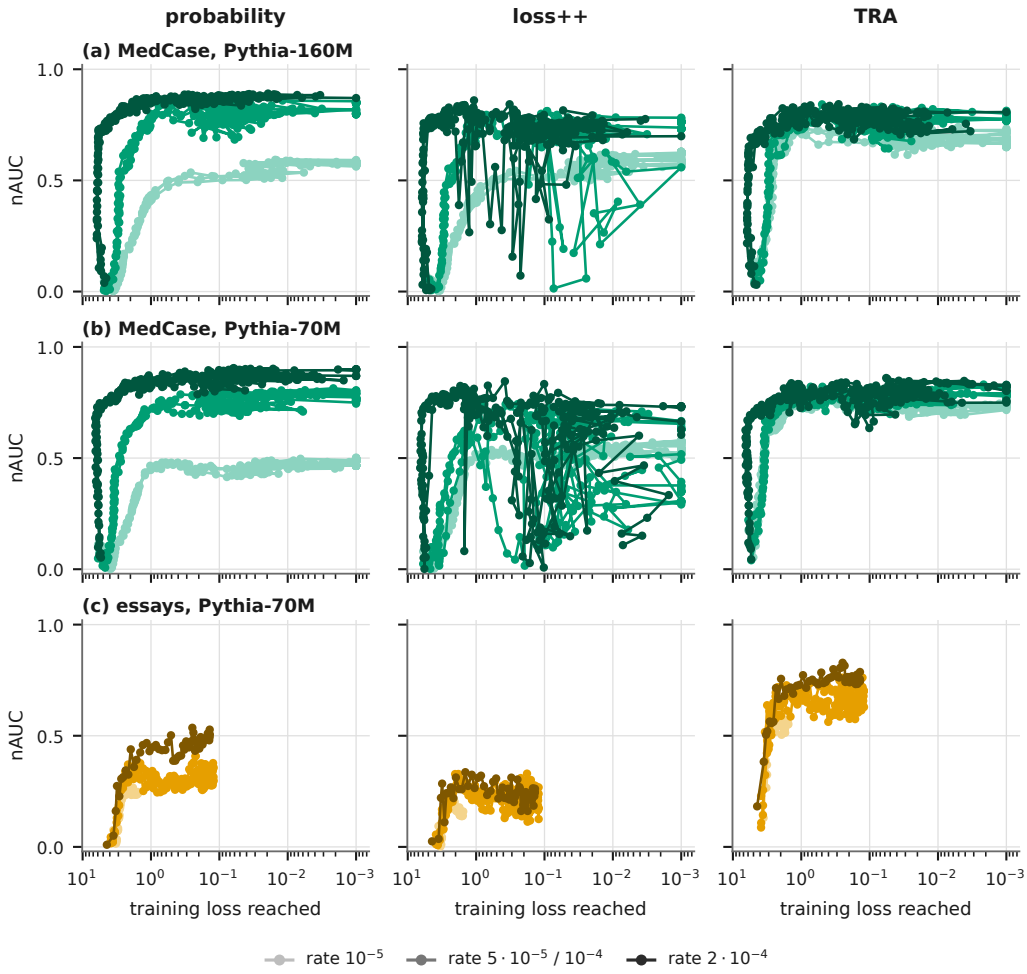


Figure 23: Learning rate at matched fit: the runs scored along training that vary only the rate, each score against the training loss the run had reached at that point. (a, b) *MedCase* at Pythia-160M and 70M, 200 cases, three seeds per rate drawn separately, scored from the first optimizer step against the same-step reference (the step-level runs of Figure 5b). (c) The *essays* at Pythia-70M on the full corpus, one seed at the smallest and largest rate and three at 10^{-4}, drawn as separate lines; sixty epochs bring their training loss only to about a tenth. The middle rate is $5 \cdot 10^{-5}$ for *MedCase* and 10^{-4} for the *essays*, and darker is a larger rate.

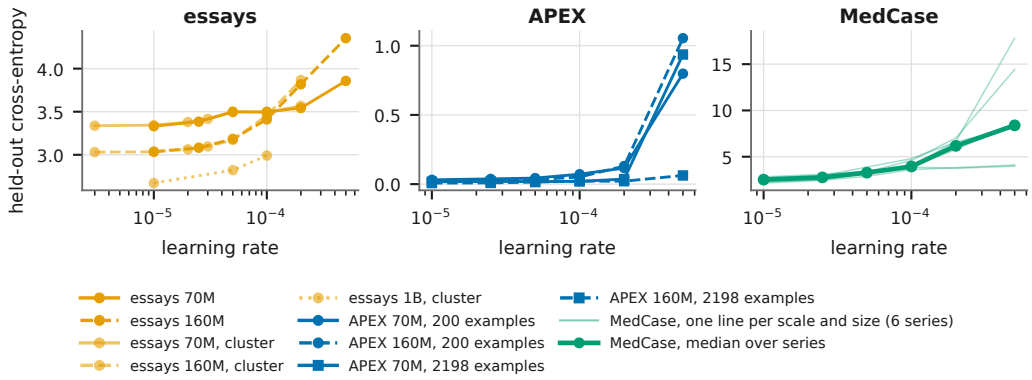


Figure 24: Held-out cross-entropy at the best-val checkpoint against the learning rate, same series and styles as Figure 22; the cluster sweep reports test cross-entropy, the local sweeps validation cross-entropy.

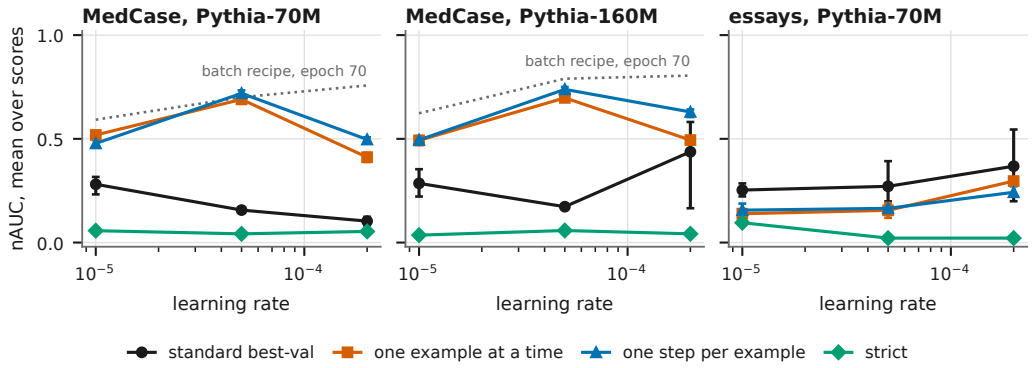


Figure 25: Training one example at a time. nAUC averaged over the four scores against the learning rate, seed means with the range over three seeds, same tokenization, validation set and optimizer settings as the recipes, batch of one example, constant rate. “One example at a time” keeps stepping on an example while validation improves; “one step per example” takes exactly one; “strict” also undoes a first step that does not improve validation. Black is the batch recipe at its best-val checkpoint at the same rate, and the dotted grey line the batch recipe after seventy epochs.

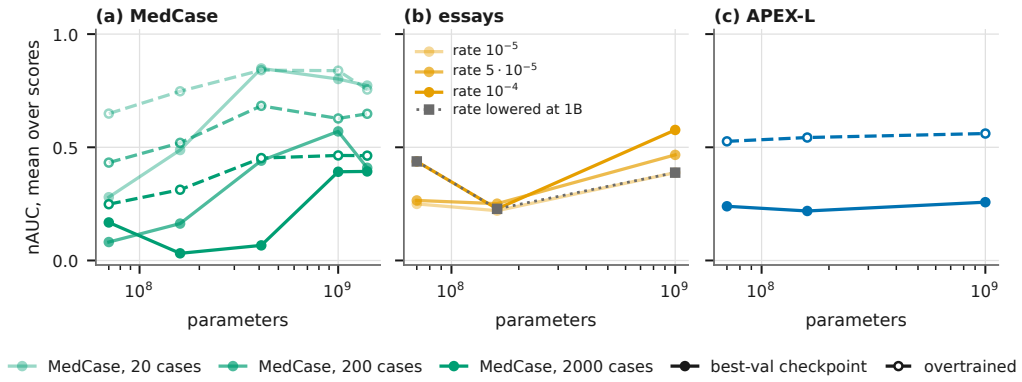


Figure 26: Memorization against model size with the recipe held fixed, nAUC averaged over scores. (a) *MedCase* at five Pythia sizes, one line per training-set size, at the best-val checkpoint (solid) and overtrained (dashed); one seed per configuration. (b) *Essays* at best-val at three sizes, one line per learning rate applied to all three (seed means from the cluster sweep); the dotted grey line is the scale-adapted series, which trains 70M and 160M at 10^{-4} and 1B at 10^{-5}. (c) *APEX*-L at three sizes, averaged over the five rule levels, at best-val and at the last scored epoch.

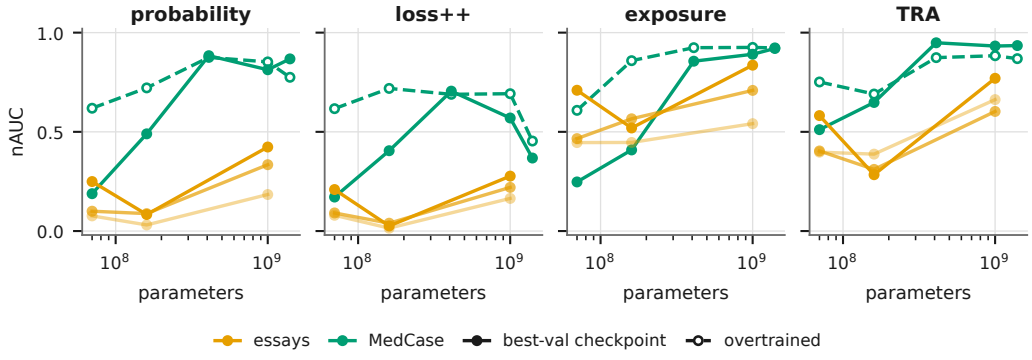


Figure 27: Panel h of Figure 1 one score at a time: *MedCase* with 20 cases at the best-val checkpoint (solid) and overtrained (dashed), and the *essays* at best-val at three fixed learning rates (darker is a larger rate).

E FULL MITIGATION FRONTIERS

This appendix shows the frontiers behind Figure 2 with every method drawn on its own, one score at a time, and with the directional AUC in place of nAUC. All figures use the same exported runs as the main figure. A point is one setting of a method (the seed mean where there are several seeds), a method's line is its lower-left frontier, and the faint points are settings that another setting of the same method beats on both axes. Hollow markers and dashed lines are the same method trained on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$, which sees twice the data and is a control for the effect of more data rather than a matched defense. The star is the undefended model, the hollow star the undefended recipe trained on the union, and the green dot and dotted line the untrained model. On the *essays*, settings whose held-out cross-entropy is worse than the untrained model's are left out; elsewhere they are counted as off scale. Methods that read the annotations of which tokens are sensitive are left out everywhere.

E.1 EVERY METHOD ON ITS OWN

Figures 28 to 34 split the families of Figure 2 into methods and add four campaigns the main text does not use: the *essays* and the mixture at Pythia-70M, *MedCase* at Llama-3.1-8B-Instruct, and *MedCase* at Pythia-1B trained for a fixed forty epochs. Across the TRAP variants the frozen and iterated references behave alike, and the adapter variants sit lowest on the *essays*.

The 70M *essays* campaign has 1281 runs over 464 settings, 409 of them averaged over several optimizer seeds; the 1B campaign has 264 runs over 206 settings, 32 of them with multiple seeds. Both score 772 member and 48 held-out spans, so the 70M frontier supports comparisons between replicated method families while the 1B frontier supports only larger directional differences. The two federated baselines, run at 70M only, use $\mathcal{D}_{\mathrm{tgt}}$ and $\mathcal{D}_{\mathrm{ref}}$ as their two clients and are drawn hollow like the other union-trained models; at the same cross-entropy both memorize more than LoRA trained on the union (worst-case nAUC about 0.15), SCAFFOLD at 0.26 to 0.38 and FedAvg at 0.38 to 0.47.

At Pythia-1B every one of the 81 trained *MedCase* settings has three seeds; the 41 settings at Llama-3.1-8B are single-seed. Both score 20 member diagnoses against 897 held-out ones. With twenty members the chance band of nAUC extends to about 0.26, so defended models near the floor cannot be ranked against each other, and held-out cross-entropy alone does not establish that a configuration still produces useful diagnoses. At Llama-3.1-8B TRAP again descends first, to about half the undefended leakage within a tenth of a nat; the iterated TRAP rounds reach about 0.2 at 0.4 nats above the undefended cross-entropy, and LoRA reaches the same level only 1.3 nats above it, beside DP-SGD; DP-AdamW reaches the floor only near the untrained cross-entropy, and the regularizers stay near the undefended model. Figure 32 repeats *MedCase* at Pythia-1B, single-seed, trained for a fixed forty epochs instead of to its best-val epoch: TRAP still moves the frontier, but from a higher starting point, so checkpoint selection and the penalty are complementary.

Figures 33 and 34 do the same for the rare-minority mixture of Section 3.1: *APEX*-FavNum secrets at a 1% share inside ordinary *APEX*, 2000 examples per half with twenty minority records, trained with the recipe of Appendix D.4 at Pythia-70M and 160M, every setting at three seeds with the data split and the training seed varying together. The sensitive spans are the twenty minority secrets, scored against 200 held-out applicants, so the chance band of nAUC extends to about 0.27; the untrained model scores 0.11 at 160M and 0.13 at 70M, and models near those values read as chance. The grid is larger than on the *essays*: TRAP with its frozen reference at four weights and two thresholds, iterated rounds at weights 2 and 4, TRAP through LoRA adapters of rank 4, 8 and 64 at weights 1, 2 and 4 (iterated rounds at three of these settings), LoRA at ranks 1, 2, 4, 8 and 64 with their union twins, DP-AdamW at three budgets on the DP trainer's default lot and at four budgets on a lot of 1024, each with a union twin, DP-SGD, weight decay, gradient clipping, dropout, and the nine label-free objectives. Ranks 1 and 2 were given an 80-epoch ceiling with the same early stopping, because rank 4 already ran to the recipe's 40 epochs at 70M; no other run reached the ceiling. The frozen reference gives the steepest frontier of the full-model variants at both scales, and the iterated rounds improve on it only at 70M and weight 2, unlike on *MedCase*. At 160M, TRAP through an adapter at weight 1 or 2 sits at the floor at every rank, between 0.11 and 0.16, at a cross-entropy at or below the undefended model's, because the adapter improves the mixture's held-out loss on its own. LoRA alone leaks less as its rank falls, from 0.51 at rank 64 to 0.17 at rank 1, while its cross-entropy rises toward the undefended model's; at every rank both were run

at, adding the penalty to the adapter lowers the leakage for about a thousandth of a nat, most at rank 64, from 0.51 to 0.14. At 70M the ordering is the same, with the rank-64 adapter at weight 1 below the undefended cross-entropy. The objectives and regularizers stay near the undefended model unless they wreck utility, and the union twins leak about as much as their matched models on every score but TRA, which their reference cannot measure. DP needs its lot tuned before it is on the panel at all: at the DP trainer's default lot of 256 examples the best budget costs 14 to 20 times the undefended cross-entropy, while a lot of 1024 brings ε=256 to 1.6 times at 160M and 2.2 times at 70M (Appendix E.2 lists both). Even tuned, DP buys utility with privacy on this testbed: the budgets that fit on the panel leak 0.41 to 0.51, as much as a rank-64 adapter, and the budgets that bring leakage to 0.14 to 0.18 cost four to eighteen times the undefended cross-entropy, against about 1.7 times for TRAP at weight 4.

E.2 The settings behind the frontiers

Tables 4 to 10 list, for every method drawn in Figures 28 to 34 and in Figure 2, the settings on its lower-left frontier: the value of the method's own hyperparameters, followed in gray by the point's held-out cross-entropy and worst-case nAUC. A row together with the testbed's shared recipe, given in each caption, reproduces the point; settings that another setting of the same method beats on both axes are not listed. Every model, defended or not, trains under the shared recipe of its testbed (DP-SGD and DP-AdamW under their own optimizer's) and stops at its own best-validation epoch under the same early-stopping rule (except the overtrained campaign of Table 8, trained for a fixed forty epochs), so its utility is the best that setting reaches. Within a method only its own hyperparameters vary: the weight for TRAP, the rank for LoRA, the budget for DP, the level or fraction of an objective. The grids differ in size, since TRAP's own variants are part of the study, and a frontier can only move outward with more settings, so the comparison rests on where the sweeps go rather than on how many points they hold: each hyperparameter runs from a value with little or no effect up to the strongest value in its published range, on *MedCase* and the mixture up to the point where utility collapses, and at every strength we ran, an objective or regularizer either keeps the undefended memorization or gives up utility before memorization moves. Where seeds are replicated the frontiers pass through seed means, which removes the advantage a larger grid gains from seed noise. TRAP's threshold ϵ is zero except for a few cells, given in the tables. Round 1 of an iterated pipeline is the target trained against the ordinary frozen reference, round r is trained against the defended target of round $r-1$, and "on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$" marks the union-trained twins.

E.3 One score at a time

Figure 35 splits the worst case of the family frontiers into its four scores, with the methods pooled into the same families. Which score is strongest depends on the testbed: on the *essays* the undefended model is exposed mostly through exposure and TRA, while probability and loss++ sit close to the floor even without a defense, and on *MedCase* all four scores are high. The reduction TRAP achieves holds under every score, not only under TRA, which the penalty is trained against. TRA needs the reference model, so an attacker holding only the released model is described by the first three columns. Figure 36 does the same for the mixture, where all four scores expose the undefended model and the family frontiers keep their order under each of them.

E.4 Directional AUC versions of the frontiers

The normalized AUC of Equation 4 is the threshold-free counterpart of the membership advantage of Yeom et al. (2018): $2\,\mathrm{AUC} - 1$ is the Gini coefficient of the member and non-member score distributions, and nAUC is its absolute value. Figures 37 to 43 repeat the per-method frontiers with the raw, directional AUC of the strongest score on the vertical axis. The points, the frontier connections and the marker conventions are the same; only the plotted height changes, so a point can now fall below the chance line at 0.5. Such a point is a model whose strongest score ranks held-out spans above training spans: the defense over-corrected and made its own training spans less predictable than unseen ones. The strongest TRAP settings do this on the *essays* at 1B, on *MedCase* and on the mixture, and so do the DP models at the floor. nAUC folds these points back up, which is the conservative reading, since an attacker who knows the defense flips the sign. It is also a sign of a penalty stronger than necessary, which a smaller weight w or a larger threshold ϵ should remove.

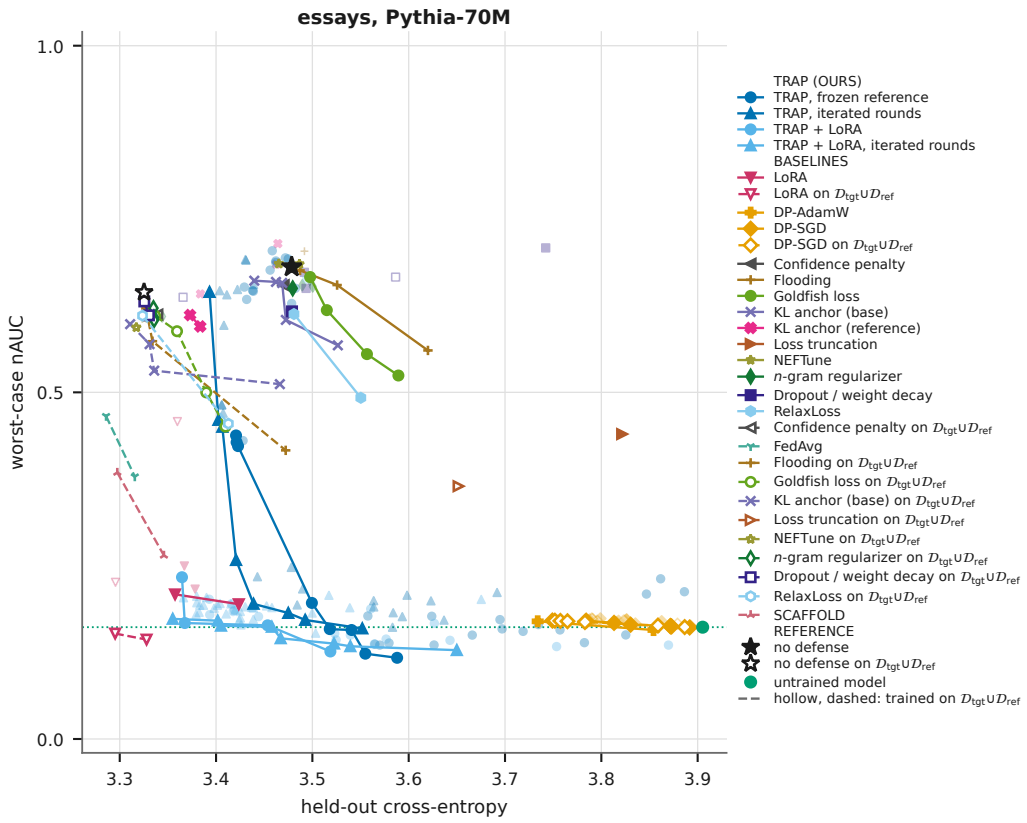


Figure 28: *Essays* at Pythia-70M, every method on its own, best-val checkpoint. Lower-left is better.

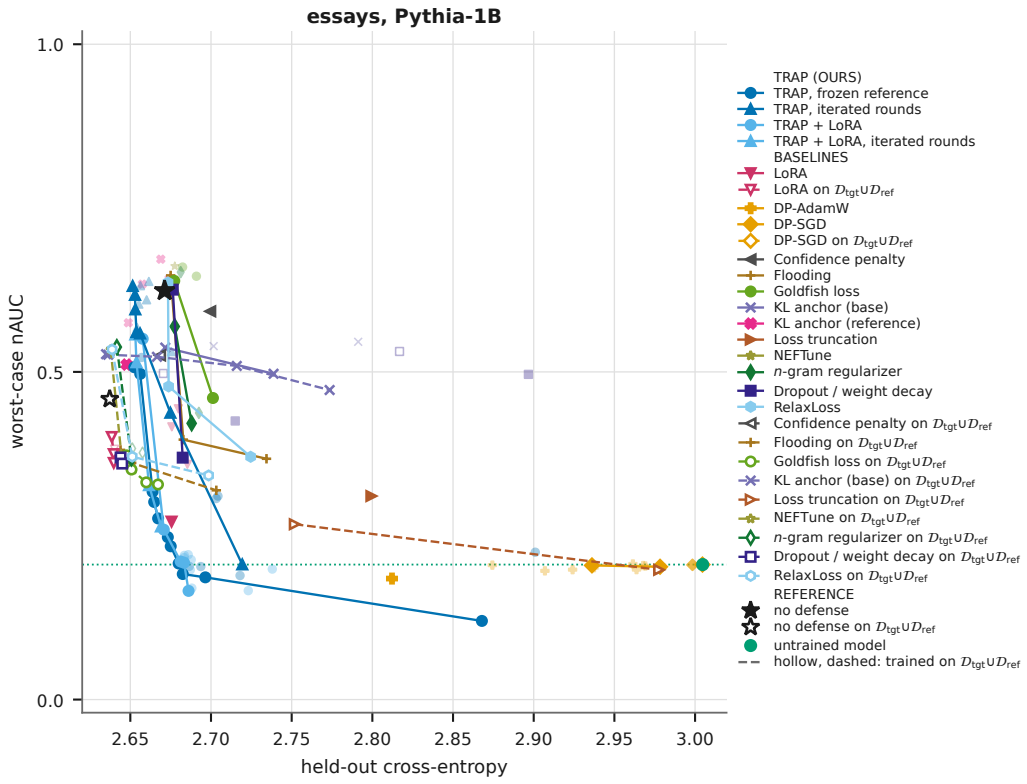


Figure 29: *Essays* at Pythia-1B, every method on its own, best-val checkpoint.

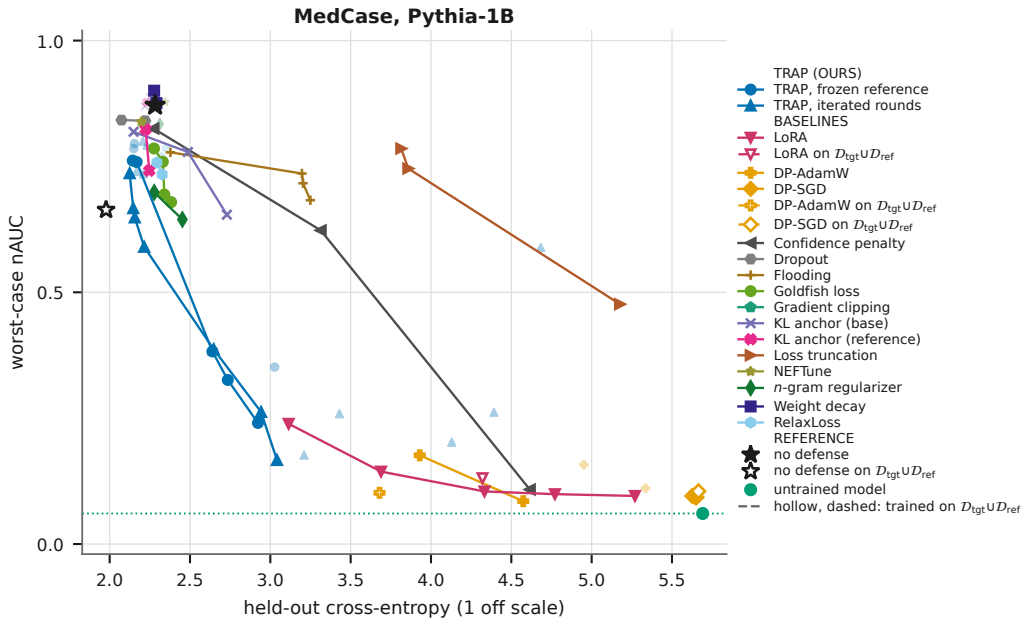


Figure 30: *MedCase* at Pythia-1B, every method on its own, best-val checkpoint, seed means over three seeds; twenty member spans, so differences near the floor are directions rather than rankings. One cell is off scale, DP-AdamW at $\varepsilon = 1$, at 5.9 nats against an undefended 2.28.

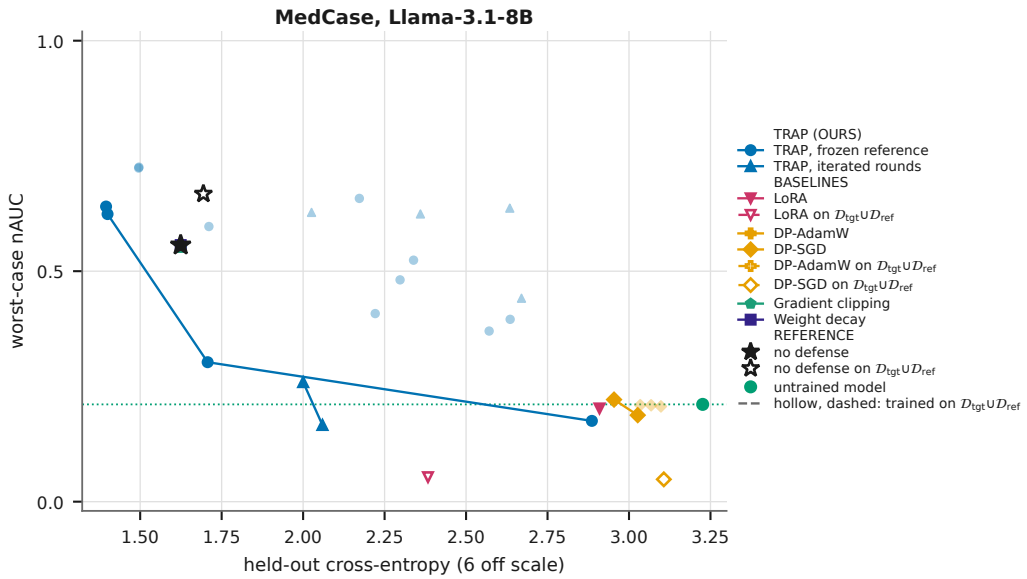


Figure 31: *MedCase* at Llama-3.1-8B-Instruct, every method on its own, best-val checkpoint. The six cells off scale, beyond $2.5\times$ the undefended cross-entropy of 1.62 nats, are all DP-AdamW: $\varepsilon = 16$ at 9.7, 8 at 11.9, 2 at 13.3, 4 at 13.5, 1 at 14.1 nats, and the $\varepsilon = 8$ twin trained on $\mathcal{D}_{\text{tgt}} \cup \mathcal{D}_{\text{ref}}$ at 9.1.

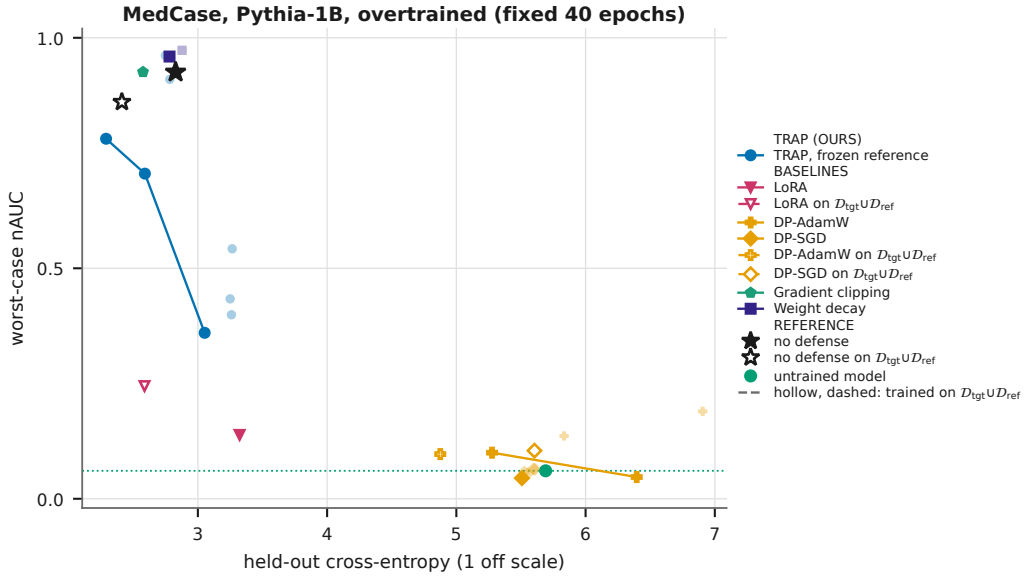


Figure 32: *MedCase* at Pythia-1B trained for a fixed forty epochs instead of to the best-val epoch. One cell is off scale, DP-AdamW at $\varepsilon = 1$, at 8.3 nats against an undefended 2.83.

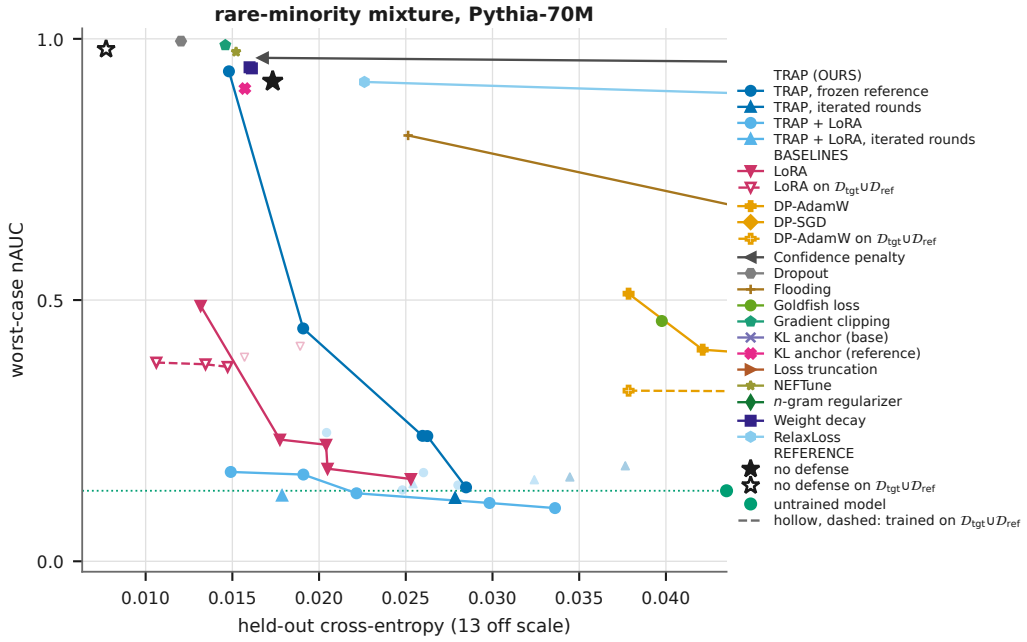


Figure 33: The rare-minority mixture at Pythia-70M, every method on its own, best-val checkpoint; seed means over three seeds, twenty minority secrets as the sensitive spans, so differences near the floor are directions rather than rankings. The undefended cross-entropy is 0.017 nats, so the 2.5× window leaves thirteen cells off scale: flooding at level 0.03 (0.07 nats), DP-AdamW at $\varepsilon = 16$ on a lot of 1024 (0.12), KL anchor to the base model at $\lambda = 0.5$ (0.15), the n-gram regularizer at $\lambda = 2$ (0.23), DP-AdamW at $\varepsilon = 64$ (0.24) and at $\varepsilon = 8$ on a lot of 1024 (0.32), RelaxLoss at $\alpha = 0.03$ (0.43), DP-SGD at $\varepsilon = 8$ (0.56), DP-AdamW at $\varepsilon = 8$ on $\mathcal{D}_{\text{tgt}} \cup \mathcal{D}_{\text{ref}}$ (0.94) and on $\mathcal{D}_{\text{tgt}}$ (1.15), the confidence penalty at $\beta = 0.5$ (1.59), loss truncation at 0.2 (3.39), and DP-AdamW at $\varepsilon = 1$ (10.0).

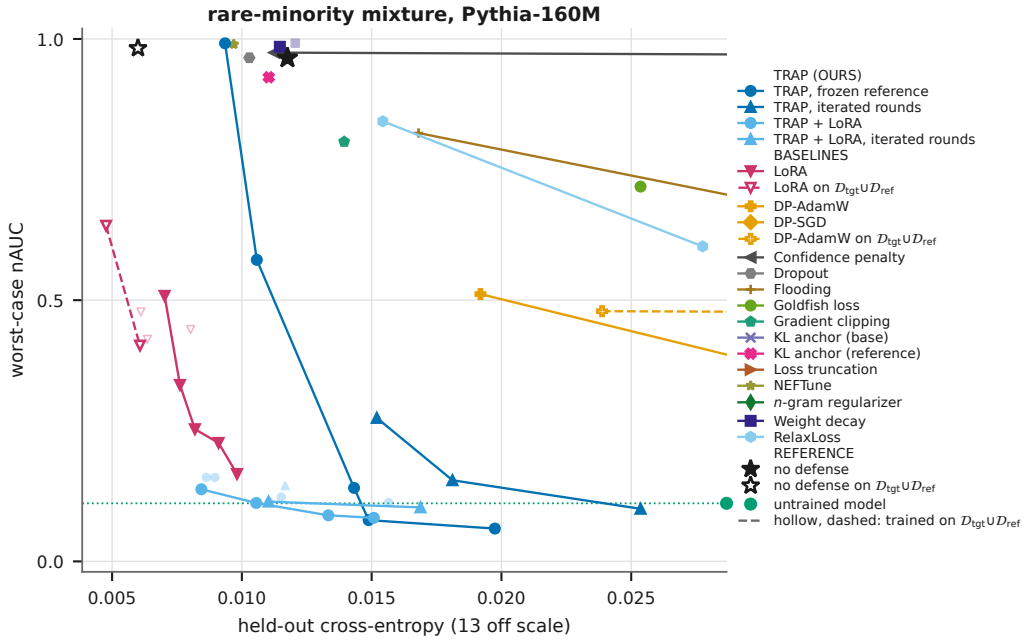


Figure 34: The rare-minority mixture at Pythia-160M, every method on its own, best-val checkpoint; seed means over three seeds. The undefended cross-entropy is 0.012 nats; the thirteen cells off scale are flooding at level 0.03 (0.03 nats), DP-AdamW at $\varepsilon = 64$ on a lot of 1024 (0.05), KL anchor to the base model at $\lambda = 0.5$ (0.07), the n-gram regularizer at $\lambda = 2$ (0.15), DP-AdamW at $\varepsilon = 64$ (0.23), DP-SGD at $\varepsilon = 8$ (0.30), DP-AdamW at $\varepsilon = 16$ and 8 on a lot of 1024 (0.37 and 1.36), DP-AdamW at $\varepsilon = 8$ on $\mathcal{D}_{\text{tgt}} \cup \mathcal{D}_{\text{ref}}$ (1.07) and on $\mathcal{D}_{\text{tgt}}$ (1.46), the confidence penalty at $\beta = 0.5$ (1.53), loss truncation at 0.2 (4.06), and DP-AdamW at $\varepsilon = 1$ (13.7).

Table 4: *Essays* at Pythia-70M: the settings on each method's lower-left frontier in Figure 28, with the point's held-out cross-entropy and worst-case nAUC in gray. Every run uses the uniform recipe of Appendix C: learning rate 10^{-4}, constant schedule with 200 warmup steps, 8 documents per step, block size 2048, fp32, no weight decay, patience 4 on the validation split, a 60-epoch ceiling; DP runs clip at 0.1 with $\delta{=}10^{-5}$ and, unless the setting says otherwise, the same lot and learning rate; only the listed hyperparameter varies. w and ϵ are the TRAP weight and threshold, r the adapter rank, ε under DP the privacy budget.

Method	Settings on the frontier (cross-entropy, worst-case nAUC)
TRAP, frozen reference	$w{=}1,\ \epsilon{=}0.05$ (3.42, 0.44); $w{=}1,\ \epsilon{=}0.02$ (3.42, 0.43); $w{=}1,\ \epsilon{=}0$ (3.42, 0.42); $w{=}4,\ \epsilon{=}1$ (3.50, 0.20); $w{=}2,\ \epsilon{=}0.2$ (3.52, 0.16); $w{=}2,\ \epsilon{=}0.1$ (3.54, 0.16); $w{=}2,\ \epsilon{=}0.05$ (3.55, 0.12); $w{=}2,\ \epsilon{=}-0.05$ (3.59, 0.12)
TRAP, iterated rounds	$w{=}0.9,\ \epsilon{=}0$, round 2 (first round alternating) (3.39, 0.64); $w{=}1,\ \epsilon{=}0.02$, round 2 (3.40, 0.46); $w{=}1,\ \epsilon{=}0.05$, round 2 (3.41, 0.45); $w{=}1.25,\ \epsilon{=}0$, round 3 (3.42, 0.26); $w{=}1.5,\ \epsilon{=}0$, round 3 (first round alternating) (3.44, 0.19); $w{=}2,\ \epsilon{=}0$, round 3 (first round alternating) (3.47, 0.18); $w{=}2,\ \epsilon{=}-0.05$, round 3 (first round alternating) (3.49, 0.17); $w{=}3,\ \epsilon{=}0$, round 2 (3.55, 0.16)
TRAP + LoRA	$r{=}16,\ w{=}0.5,\ \epsilon{=}0$ (3.36, 0.23); $r{=}64,\ w{=}1,\ \epsilon{=}0$ (3.37, 0.17); $r{=}4,\ w{=}1,\ \epsilon{=}0.02$ (3.45, 0.16); $r{=}128,\ w{=}2,\ \epsilon{=}0$ (3.52, 0.13)
TRAP + LoRA, iterated rounds	$r{=}16,\ w{=}0.9,\ \epsilon{=}0$, round 2 (3.36, 0.17); $r{=}4,\ w{=}0.5,\ \epsilon{=}0$, round 2 (3.40, 0.17); $r{=}8,\ w{=}0.9,\ \epsilon{=}0$, round 1 (3.41, 0.16); $r{=}4,\ w{=}1.1,\ \epsilon{=}0$, round 1 (3.46, 0.16); $r{=}16,\ w{=}1.5,\ \epsilon{=}0$, round 1 (3.47, 0.15); $r{=}64,\ w{=}2,\ \epsilon{=}0$, round 1 (3.52, 0.14); $r{=}16,\ w{=}2,\ \epsilon{=}0$, round 1 (3.54, 0.13); $r{=}64,\ w{=}3,\ \epsilon{=}0$, round 1 (3.65, 0.13)
LoRA	$r{=}64$ (3.36, 0.21); $r{=}4$ (3.42, 0.19)
LoRA on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$	$r{=}64$ (3.30, 0.15); $r{=}16$ (3.33, 0.14)
DP-AdamW	$\varepsilon{=}8$ ($L{=}1024$, $T{=}10$, lr 10^{-4}) (3.73, 0.17); $\varepsilon{=}8$ ($L{=}1024$, $T{=}60$, lr 10^{-3}) (3.85, 0.16)
DP-SGD	$\varepsilon{=}64$ ($T{=}120$) (3.78, 0.17); $\varepsilon{=}64$ (3.81, 0.17); $\varepsilon{=}64$ ($L{=}128$, $T{=}60$, lr 10^{-3}) (3.83, 0.16); $\varepsilon{=}8$ ($L{=}128$, $T{=}10$, lr 10^{-3}) (3.87, 0.16); $\varepsilon{=}1$ ($L{=}128$, $T{=}10$, lr 10^{-3}) (3.87, 0.16); $\varepsilon{=}1$ ($L{=}1024$, $T{=}60$, lr 10^{-3}) (3.87, 0.16); $\varepsilon{=}64$ ($L{=}1024$, $T{=}10$, lr 10^{-3}) (3.89, 0.16)
DP-SGD on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$	$\varepsilon{=}64$ ($T{=}120$) (3.75, 0.17); $\varepsilon{=}16$ ($T{=}120$) (3.75, 0.17); $\varepsilon{=}4$ ($T{=}120$) (3.76, 0.17); $\varepsilon{=}2$ ($T{=}120$) (3.76, 0.17); $\varepsilon{=}64$ (3.78, 0.17); $\varepsilon{=}8$ ($L{=}128$, $T{=}10$, lr 10^{-3}) (3.86, 0.16); $\varepsilon{=}8$ ($L{=}1024$, $T{=}10$, lr 10^{-3}) (3.89, 0.16)
Goldfish loss	$k{=}32$ (3.50, 0.67); $k{=}8$ (3.51, 0.62); $k{=}4$ (3.56, 0.56); $k{=}3$ (3.59, 0.52)
Goldfish loss on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$	$k{=}32$ (3.34, 0.61); $k{=}8$ (3.36, 0.59); $k{=}4$ (3.39, 0.50); $k{=}3$ (3.41, 0.45)
Flooding	$b{=}2$ (3.49, 0.68); $b{=}2.5$ (3.53, 0.65); $b{=}3$ (3.62, 0.56)
Flooding on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$	$b{=}1.5$ (3.33, 0.62); $b{=}2$ (3.33, 0.57); $b{=}3$ (3.47, 0.42)
Confidence penalty	$\beta{=}0.1$ (3.47, 0.65)
Confidence penalty on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$	$\beta{=}0.1$ (3.34, 0.61)
Loss truncation	$c{=}0.1$ (3.82, 0.44)
Loss truncation on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$	$c{=}0.1$ (3.65, 0.36)
RelaxLoss	$\alpha{=}2.5$ (3.48, 0.61); $\alpha{=}3$ (3.55, 0.49)
RelaxLoss on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$	$\alpha{=}2$ (3.32, 0.61); $\alpha{=}3$ (3.41, 0.45)
KL anchor (base)	$\lambda{=}0.5$ (3.44, 0.66); $\lambda{=}0.1$ (3.46, 0.66); $\lambda{=}0.05$ (3.47, 0.66); $\lambda{=}1$ (3.47, 0.60); $\lambda{=}2$ (3.53, 0.57)
KL anchor (base) on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$	$\lambda{=}0.1$ (3.31, 0.60); $\lambda{=}0.05$ (3.33, 0.57); $\lambda{=}0.5$ (3.34, 0.53); $\lambda{=}2$ (3.47, 0.51)
KL anchor (reference)	$\lambda{=}1$ (3.37, 0.61); $\lambda{=}2$ (3.38, 0.60)
n-gram regularizer	$\lambda{=}0.5,\ \tau{=}0$ (3.48, 0.65)
n-gram regularizer on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$	$\lambda{=}2,\ \tau{=}0.05$ (3.34, 0.62); $\lambda{=}0.5,\ \tau{=}0$ (3.34, 0.61); $\lambda{=}2,\ \tau{=}0$ (3.34, 0.60)
NEFTune	$\alpha{=}10$ (3.47, 0.69); $\alpha{=}5$ (3.49, 0.68)
NEFTune on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$	$\alpha{=}10$ (3.32, 0.59)
Dropout / weight decay	dropout $p{=}0.1$ (3.48, 0.62)
Dropout / weight decay on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$	weight decay 0.1 (3.33, 0.63); weight decay 0.01 (3.33, 0.61)
FedAvg	$E{=}1$ local epochs (3.29, 0.47); $E{=}0.5$ local epochs (3.32, 0.38)
SCAFFOLD	$E{=}1$ local epochs (3.30, 0.38); $E{=}0.5$ local epochs (3.35, 0.26)

Table 5: *Essays* at Pythia-1B: the settings on each method's lower-left frontier in Figure 29, with the point's held-out cross-entropy and worst-case nAUC in gray. Every run uses the recipe of Table 4 at learning rate 10^{-5}; DP runs list their lot L, epoch budget T and learning rate; only the listed hyperparameter varies. w and ϵ are the TRAP weight and threshold, r the adapter rank, ε under DP the privacy budget.

Method	Settings on the frontier (cross-entropy, worst-case nAUC)
TRAP, frozen reference	w=1, ϵ=0 (2.65, 0.51); w=1, ϵ=0.05 (2.66, 0.50); w=1.5, ϵ=0.1 (2.66, 0.32); w=1.5, ϵ=0.05 (2.66, 0.30); w=1.5, ϵ=0 (2.67, 0.28); w=2, ϵ=0.1 (2.67, 0.25); w=2, ϵ=0.05 (2.68, 0.23); w=2, ϵ=0 (2.68, 0.21); w=2, ϵ=-0.02 (2.68, 0.19); w=2, ϵ=-0.05 (2.70, 0.19); w=2, ϵ=0.05 (2.87, 0.12)
TRAP, iterated rounds	w=0.5, ϵ=0, round 2 (first round mutual) (2.65, 0.63); w=0.5, ϵ=0, round 2 (first round alternating) (2.65, 0.62); w=1, ϵ=0, round 2 (first round mutual) (2.65, 0.60); w=1, ϵ=0, round 2 (reference: undefended model) (2.65, 0.56); w=0.6, ϵ=0, round 2 (2.66, 0.56); w=0.25, ϵ=0, round 3 (2.67, 0.44); w=2, ϵ=0, round 2 (2.72, 0.21)
TRAP + LoRA	r=4, w=1, ϵ=0 (2.66, 0.55); r=16, w=1.5, ϵ=0 (2.67, 0.26); r=8, w=2, ϵ=0 (2.68, 0.21); r=256, w=2, ϵ=0 (2.68, 0.21); r=16, w=2, ϵ=-0.02 (2.69, 0.17)
TRAP + LoRA, iterated rounds	r=4, w=0.9, ϵ=0, round 1 (2.65, 0.56); r=4, w=0.9, ϵ=0, round 2 (2.65, 0.51); r=4, w=1.3, ϵ=0, round 2 (2.66, 0.33); r=16, w=1.5, ϵ=0, round 1 (2.67, 0.26)
LoRA	r=256 (2.68, 0.27)
LoRA on $\mathcal{D}_{\text{tgt}} \cup \mathcal{D}_{\text{ref}}$	r=256 (2.64, 0.40); r=4 (2.64, 0.37); r=16 (2.64, 0.36)
DP-AdamW	ε=64 (L=1024, T=10, lr 10^{-4}) (2.81, 0.18)
DP-SGD	ε=8 (L=8, T=60, lr 10^{-4}) (2.94, 0.20); ε=8 (L=128, T=10, lr 10^{-3}) (2.98, 0.20)
DP-SGD on $\mathcal{D}_{\text{tgt}} \cup \mathcal{D}_{\text{ref}}$	ε=1 (L=1024, T=10, lr 10^{-5}) (3.00, 0.21)
Goldfish loss	k=32 (2.68, 0.64); k=3 (2.70, 0.46)
Goldfish loss on $\mathcal{D}_{\text{tgt}} \cup \mathcal{D}_{\text{ref}}$	k=32 (2.65, 0.37); k=8 (2.65, 0.35); k=4 (2.66, 0.33); k=3 (2.67, 0.33)
Flooding	b=1.5 (2.67, 0.65); b=2 (2.68, 0.40); b=2.5 (2.73, 0.37)
Flooding on $\mathcal{D}_{\text{tgt}} \cup \mathcal{D}_{\text{ref}}$	b=1.5 (2.64, 0.38); b=2 (2.65, 0.36); b=2.5 (2.70, 0.32)
Confidence penalty	β=0.1 (2.70, 0.59)
Confidence penalty on $\mathcal{D}_{\text{tgt}} \cup \mathcal{D}_{\text{ref}}$	β=0.1 (2.67, 0.53)
Loss truncation	c=0.1 (2.80, 0.31)
Loss truncation on $\mathcal{D}_{\text{tgt}} \cup \mathcal{D}_{\text{ref}}$	c=0.1 (2.75, 0.27); c=0.2 (2.98, 0.20)
RelaxLoss	α=2 (2.67, 0.64); α=1.5 (2.67, 0.48); α=2.5 (2.72, 0.37)
RelaxLoss on $\mathcal{D}_{\text{tgt}} \cup \mathcal{D}_{\text{ref}}$	α=1.5 (2.64, 0.53); α=2 (2.65, 0.37); α=2.5 (2.70, 0.34)
KL anchor (base)	λ=0.1 (2.67, 0.54); λ=1 (2.74, 0.50)
KL anchor (base) on $\mathcal{D}_{\text{tgt}} \cup \mathcal{D}_{\text{ref}}$	λ=0.1 (2.63, 0.53); λ=0.05 (2.64, 0.53); λ=0.5 (2.67, 0.52); λ=1 (2.72, 0.51); λ=2 (2.77, 0.47)
KL anchor (reference)	λ=2 (2.65, 0.51)
n-gram regularizer	λ=0.5, τ=0 (2.68, 0.57); λ=2, τ=0 (2.69, 0.42)
n-gram regularizer on $\mathcal{D}_{\text{tgt}} \cup \mathcal{D}_{\text{ref}}$	λ=0.5, τ=0 (2.64, 0.54); λ=2, τ=0 (2.65, 0.37)
NEFTune	α=5 (2.68, 0.63)
NEFTune on $\mathcal{D}_{\text{tgt}} \cup \mathcal{D}_{\text{ref}}$	α=10 (2.64, 0.53); α=5 (2.64, 0.37)
Dropout / weight decay	weight decay 0.01 (2.68, 0.63); weight decay 0.1 (2.68, 0.37)
Dropout / weight decay on $\mathcal{D}_{\text{tgt}} \cup \mathcal{D}_{\text{ref}}$	weight decay 0.1 (2.64, 0.37); weight decay 0.01 (2.64, 0.36)

Table 6: *MedCase* at Pythia-1B, twenty cases, best-val: the settings on each method's lower-left frontier in Figure 30, with the point's held-out cross-entropy and worst-case nAUC in gray. Every run uses learning rate $5 \cdot 10^{-5}$, 16 cases per step, bf16, 3% warmup, no weight decay, a 15-epoch ceiling with patience 4 on the validation split; LoRA adapters on the attention projections with $\alpha{=}2r$ and dropout 0.1; only the listed hyperparameter varies. w and ϵ are the TRAP weight and threshold, r the adapter rank, ε under DP the privacy budget.

Method	Settings on the frontier (cross-entropy, worst-case nAUC)
TRAP, frozen reference	$w{=}0.5,\ \epsilon{=}0$ (2.14, 0.76); $w{=}1,\ \epsilon{=}0$ (2.17, 0.76); $w{=}2,\ \epsilon{=}0$ (2.64, 0.38); $w{=}2,\ \epsilon{=}0.05$ (2.74, 0.33); $w{=}4,\ \epsilon{=}0$ (2.92, 0.24)
TRAP, iterated rounds	$w{=}0.5,\ \epsilon{=}0$, round 4 (2.12, 0.74); $w{=}1,\ \epsilon{=}0$, round 5 (2.15, 0.67); $w{=}1,\ \epsilon{=}0$, round 2 (2.16, 0.65); $w{=}1,\ \epsilon{=}0$, round 3 (2.21, 0.59); $w{=}2,\ \epsilon{=}0$, round 2 (2.65, 0.39); $w{=}2,\ \epsilon{=}0$, round 4 (2.94, 0.26); $w{=}2,\ \epsilon{=}0$, round 3 (3.04, 0.17)
LoRA	$r{=}64$ (3.11, 0.24); $r{=}32$ (3.69, 0.14); $r{=}16$ (4.33, 0.10); $r{=}8$ (4.77, 0.10); $r{=}4$ (5.27, 0.10)
LoRA on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$	$r{=}8$ (4.32, 0.13)
DP-AdamW	$\varepsilon{=}16$ (lot 256) (3.93, 0.18); $\varepsilon{=}8$ (lot 256) (4.57, 0.09)
DP-SGD	$\varepsilon{=}1$ (lot 256) (5.63, 0.10); $\varepsilon{=}4$ (lot 256) (5.65, 0.09); $\varepsilon{=}2$ (lot 256) (5.65, 0.09)
DP-AdamW on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$	$\varepsilon{=}8$ (lot 256) (3.68, 0.10)
DP-SGD on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$	$\varepsilon{=}8$ (lot 256) (5.66, 0.10)
Goldfish loss	$k{=}8$ (2.28, 0.79); $k{=}32$ (2.33, 0.76); $k{=}3$ (2.34, 0.69); $k{=}4$ (2.38, 0.68)
Flooding	$b{=}1.5$ (2.38, 0.78); $b{=}2$ (3.20, 0.74); $b{=}2.5$ (3.20, 0.72); $b{=}3$ (3.25, 0.68)
Confidence penalty	$\beta{=}0.1$ (2.27, 0.83); $\beta{=}0.5$ (3.31, 0.62); $\beta{=}1$ (4.62, 0.11)
Loss truncation	$c{=}0.1$ (3.81, 0.79); $c{=}0.2$ (3.86, 0.75); $c{=}0.4$ (5.18, 0.48)
RelaxLoss	$\alpha{=}3$ (2.29, 0.76); $\alpha{=}2$ (2.33, 0.73)
KL anchor (base)	$\lambda{=}0.5$ (2.15, 0.82); $\lambda{=}1$ (2.49, 0.78); $\lambda{=}2$ (2.73, 0.65)
KL anchor (reference)	$\lambda{=}0.5$ (2.22, 0.82); $\lambda{=}1$ (2.25, 0.74)
n-gram regularizer	$\lambda{=}2$ (2.28, 0.70); $\lambda{=}8$ (2.45, 0.64)
NEFTune	$\alpha{=}10$ (2.20, 0.84)
Weight decay	weight decay 0.1 (2.28, 0.90); weight decay 0.3 (2.29, 0.88)
Dropout	dropout $p{=}0.1$ (2.07, 0.84); dropout $p{=}0.3$ (2.22, 0.84)
Gradient clipping	max norm 0.1 (2.29, 0.88)

Table 7: *MedCase* at Llama-3.1-8B-Instruct, twenty cases, best-val: the settings on each method's lower-left frontier in Figure 31, with the point's held-out cross-entropy and worst-case nAUC in gray. Every run uses learning rate $2 \cdot 10^{-5}$, 8 cases per step, bf16, the stopping rule of Table 6; the LoRA adapters (rank 16, $\alpha{=}32$, dropout 0.05) train at learning rate 10^{-4}; only the listed hyperparameter varies. w and ϵ are the TRAP weight and threshold, r the adapter rank, ε under DP the privacy budget.

Method	Settings on the frontier (cross-entropy, worst-case nAUC)
TRAP, frozen reference	$w{=}1,\ \epsilon{=}0.05$ (1.40, 0.64); $w{=}1,\ \epsilon{=}0$ (1.40, 0.62); $w{=}2,\ \epsilon{=}0$ (1.71, 0.30); $w{=}32,\ \epsilon{=}0.05$ (2.89, 0.18)
TRAP, iterated rounds	$w{=}2,\ \epsilon{=}0$, round 3 (2.00, 0.26); $w{=}2,\ \epsilon{=}0$, round 4 (2.06, 0.17)
LoRA	$r{=}16$ (2.91, 0.20)
LoRA on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$	$r{=}16$ (2.38, 0.05)
DP-AdamW	$\varepsilon{=}16$ (lot 256) (9.72, 0.26); $\varepsilon{=}8$ (lot 256) (11.88, 0.07)
DP-SGD	$\varepsilon{=}1$ (lot 256) (2.95, 0.22); $\varepsilon{=}2$ (lot 256) (3.03, 0.19)
DP-AdamW on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$	$\varepsilon{=}8$ (lot 256) (9.10, 0.14)
DP-SGD on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$	$\varepsilon{=}8$ (lot 256) (3.11, 0.05)
Weight decay	weight decay 0.3 (1.62, 0.56)
Gradient clipping	max norm 0.1 (1.63, 0.55)

Table 8: *MedCase* at Pythia-1B, twenty cases, overtrained: the settings on each method's lower-left frontier in Figure 32, with the point's held-out cross-entropy and worst-case nAUC in gray. Every run uses the recipe of Table 6 trained for a fixed 40 epochs; only the listed hyperparameter varies. w and ϵ are the TRAP weight and threshold, r the adapter rank, ε under DP the privacy budget.

Method	Settings on the frontier (cross-entropy, worst-case nAUC)
TRAP, frozen reference	w=1, ϵ=0.05 (2.29, 0.78); w=1, ϵ=0 (2.59, 0.71); w=2, ϵ=0.05 (3.05, 0.36)
LoRA	r=8 (3.32, 0.14)
LoRA on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$	r=8 (2.59, 0.24)
DP-AdamW	ε=16 (lot 256) (5.28, 0.10); ε=4 (lot 256) (6.40, 0.05)
DP-SGD	ε=1 (lot 256) (5.51, 0.04)
DP-AdamW on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$	ε=8 (lot 256) (4.88, 0.10)
DP-SGD on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$	ε=8 (lot 256) (5.60, 0.10)
Weight decay	weight decay 0.3 (2.78, 0.96)
Gradient clipping	max norm 0.1 (2.58, 0.93)

Table 9: The rare-minority mixture at Pythia-70M: the settings on each method's lower-left frontier in Figure 33, with the point's held-out cross-entropy and worst-case nAUC in gray. Every run uses learning rate $5 \cdot 10^{-5}$, 16 examples per step, 3% warmup, fp32, no weight decay, a 40-epoch ceiling with patience 3 on 200 held-out examples; LoRA adapters on the attention projections with α=2r and dropout 0.1, ranks 1 and 2 with an 80-epoch ceiling; DP with lot 256, learning rate 10^{-3}, clipping 0.1 and δ=10^{-5}; only the listed hyperparameter varies. w and ϵ are the TRAP weight and threshold, r the adapter rank, ε under DP the privacy budget.

Method	Settings on the frontier (cross-entropy, worst-case nAUC)
TRAP, frozen reference	w=0.5, ϵ=0 (0.0148, 0.94); w=1, ϵ=0 (0.0191, 0.45); w=2, ϵ=0.05 (0.0260, 0.24); w=2, ϵ=0 (0.0262, 0.24); w=4, ϵ=0 (0.0285, 0.14)
TRAP, iterated rounds	w=2, ϵ=0, round 2 (0.0278, 0.12)
TRAP + LoRA	r=64, w=1, ϵ=0 (0.0149, 0.17); r=8, w=1, ϵ=0 (0.0191, 0.17); r=4, w=1, ϵ=0 (0.0222, 0.13); r=8, w=4, ϵ=0 (0.0298, 0.11); r=4, w=4, ϵ=0 (0.0336, 0.10)
TRAP + LoRA, iterated rounds	r=64, w=2, ϵ=0, round 2 (0.0179, 0.13)
LoRA	r=64 (0.0132, 0.49); r=8 (0.0177, 0.23); r=2 (0.0204, 0.22); r=4 (0.0205, 0.18); r=1 (0.0253, 0.16)
LoRA on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$	r=64 (0.0106, 0.38); r=8 (0.0135, 0.38); r=4 (0.0147, 0.37)
DP-AdamW	ε=256 (lot 1024) (0.0378, 0.51); ε=64 (lot 1024) (0.0421, 0.41); ε=16 (lot 1024) (0.12, 0.22); ε=8 (lot 1024) (0.32, 0.18); ε=1 (lot 256) (10.03, 0.09)
DP-AdamW on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$	ε=64 (lot 1024) (0.0378, 0.33); ε=8 (lot 256) (0.94, 0.19)
DP-SGD	ε=8 (lot 256) (0.56, 0.12)
Goldfish loss	k=3 (0.0398, 0.46)
Flooding	b=0.01 (0.0251, 0.82); b=0.03 (0.0669, 0.52)
Confidence penalty	β=0.1 (0.0168, 0.96); β=0.5 (1.59, 0.55)
Loss truncation	c=0.2 (3.39, 0.14)
RelaxLoss	α=0.01 (0.0226, 0.92); α=0.03 (0.43, 0.51)
KL anchor (base)	λ=0.5 (0.15, 0.68)
KL anchor (reference)	λ=0.5 (0.0157, 0.90)
n-gram regularizer	λ=2 (0.23, 0.65)
NEFTune	α=5 (0.0152, 0.98)
Weight decay	weight decay 0.3 (0.0160, 0.95); weight decay 0.1 (0.0161, 0.94)
Dropout	dropout p=0.1 (0.0120, 1.00)
Gradient clipping	max norm 0.1 (0.0146, 0.99)

Table 10: The rare-minority mixture at Pythia-160M: the settings on each method's lower-left frontier in Figure 34, with the point's held-out cross-entropy and worst-case nAUC in gray. Every run uses the recipe of Table 9; only the listed hyperparameter varies. w and ϵ are the TRAP weight and threshold, r the adapter rank, ε under DP the privacy budget.

Method	Settings on the frontier (cross-entropy, worst-case nAUC)
TRAP, frozen reference	w=0.5, ϵ=0 (0.0094, 0.99); w=1, ϵ=0 (0.0106, 0.58); w=2, ϵ=0.05 (0.0143, 0.14); w=2, ϵ=0 (0.0149, 0.08); w=4, ϵ=0 (0.0197, 0.06)
TRAP, iterated rounds	w=2, ϵ=0, round 2 (0.0152, 0.27); w=2, ϵ=0, round 3 (0.0181, 0.16); w=4, ϵ=0, round 2 (0.0254, 0.10)
TRAP + LoRA	r=64, w=1, ϵ=0 (0.0084, 0.14); r=64, w=2, ϵ=0 (0.0106, 0.11); r=64, w=4, ϵ=0 (0.0133, 0.09); r=4, w=4, ϵ=0 (0.0151, 0.08)
TRAP + LoRA, iterated rounds	r=64, w=2, ϵ=0, round 2 (0.0110, 0.11); r=8, w=4, ϵ=0, round 2 (0.0169, 0.10)
LoRA	r=64 (0.0070, 0.51); r=8 (0.0076, 0.34); r=4 (0.0082, 0.25); r=2 (0.0091, 0.23); r=1 (0.0098, 0.17)
LoRA on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$	r=64 (0.0048, 0.64); r=8 (0.0061, 0.41)
DP-AdamW	ε=256 (lot 1024) (0.0192, 0.51); ε=64 (lot 1024) (0.0497, 0.14); ε=1 (lot 256) (13.74, 0.07)
DP-AdamW on $\mathcal{D}_{\mathrm{tgt}} \cup \mathcal{D}_{\mathrm{ref}}$	ε=64 (lot 1024) (0.0239, 0.48); ε=8 (lot 256) (1.07, 0.29)
DP-SGD	ε=8 (lot 256) (0.30, 0.10)
Goldfish loss	k=3 (0.0254, 0.72)
Flooding	b=0.01 (0.0168, 0.82); b=0.03 (0.0313, 0.68)
Confidence penalty	β=0.1 (0.0113, 0.97); β=0.5 (1.53, 0.68)
Loss truncation	c=0.2 (4.06, 0.15)
RelaxLoss	α=0.01 (0.0154, 0.84); α=0.03 (0.0278, 0.60)
KL anchor (base)	λ=0.5 (0.0692, 0.80)
KL anchor (reference)	λ=0.5 (0.0110, 0.93)
n-gram regularizer	λ=2 (0.15, 0.98)
NEFTune	α=5 (0.0097, 0.99)
Weight decay	weight decay 0.3 (0.0115, 0.98)
Dropout	dropout p=0.1 (0.0103, 0.96)
Gradient clipping	max norm 0.1 (0.0139, 0.80)

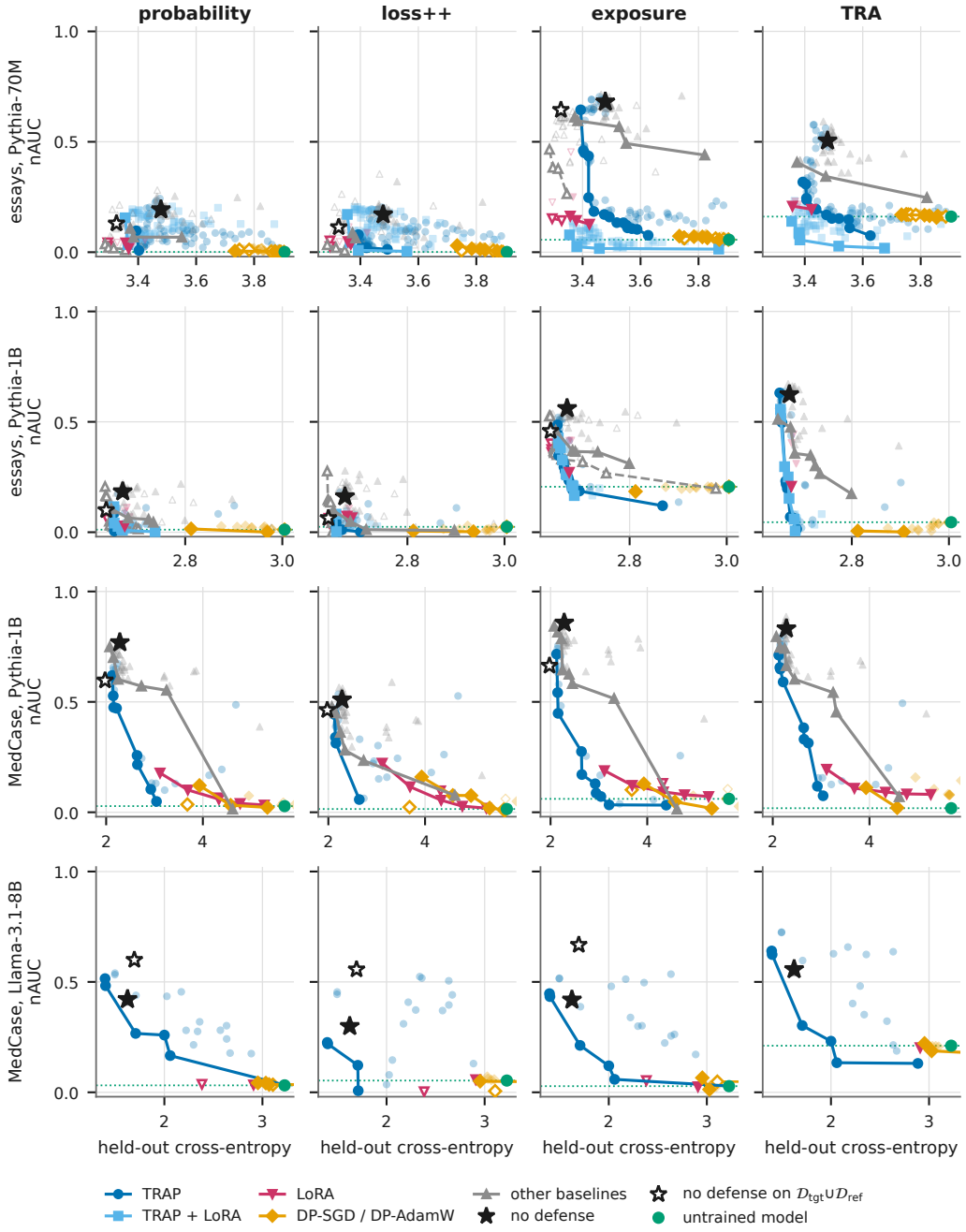


Figure 35: The family frontiers one score at a time, best-val checkpoint: rows are the *essays* at Pythia-70M and 1B, *MedCase* at Pythia-1B and Llama-3.1-8B (the second and third rows are Figure 2a and b); columns are the four scores; lines are the family frontiers.

Small values have to be read against two floors. With finitely many spans nAUC is not zero under the null: for a single score the 95% chance band is about 0.17 on the essay spans (772 members against 48 non-members) and about 0.26 on the 20-member *MedCase* runs, and values inside the band read as chance. And because the worst case takes a maximum over several scores, and member and held-out spans need not have the same composition, even an untrained model can sit above that band. Every frontier therefore marks the untrained model, the empirical floor for those scores, and defended models should be compared with it rather than with zero.

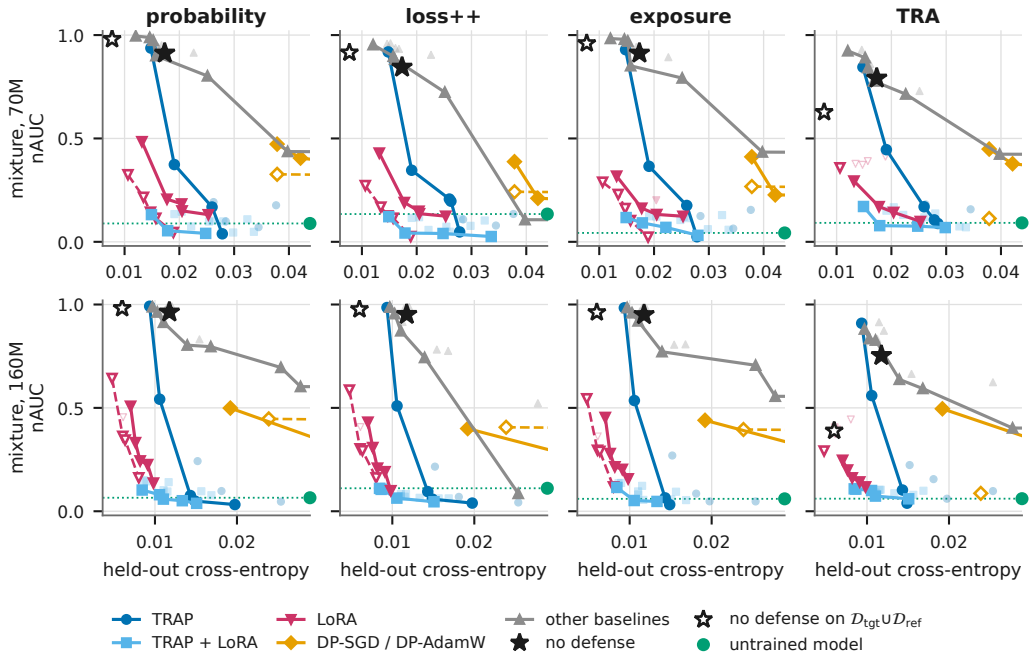


Figure 36: The rare-minority mixture one score at a time, best-val checkpoint: rows are the two model sizes (Figure 2c is the 160M row), columns the four scores, lines the family frontiers of the main figure.

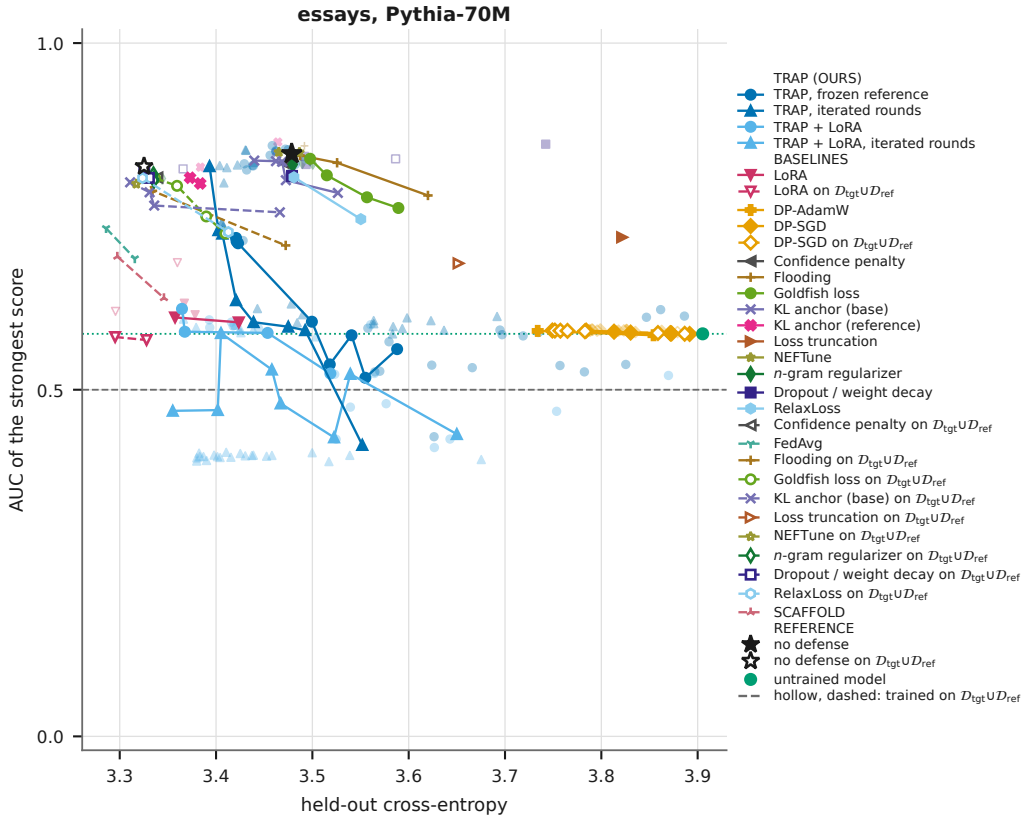


Figure 37: Directional AUC version of Figure 28: *essays* at Pythia-70M, with 0.5 marking chance; points below the line are over-corrected models that nAUC folds upward.

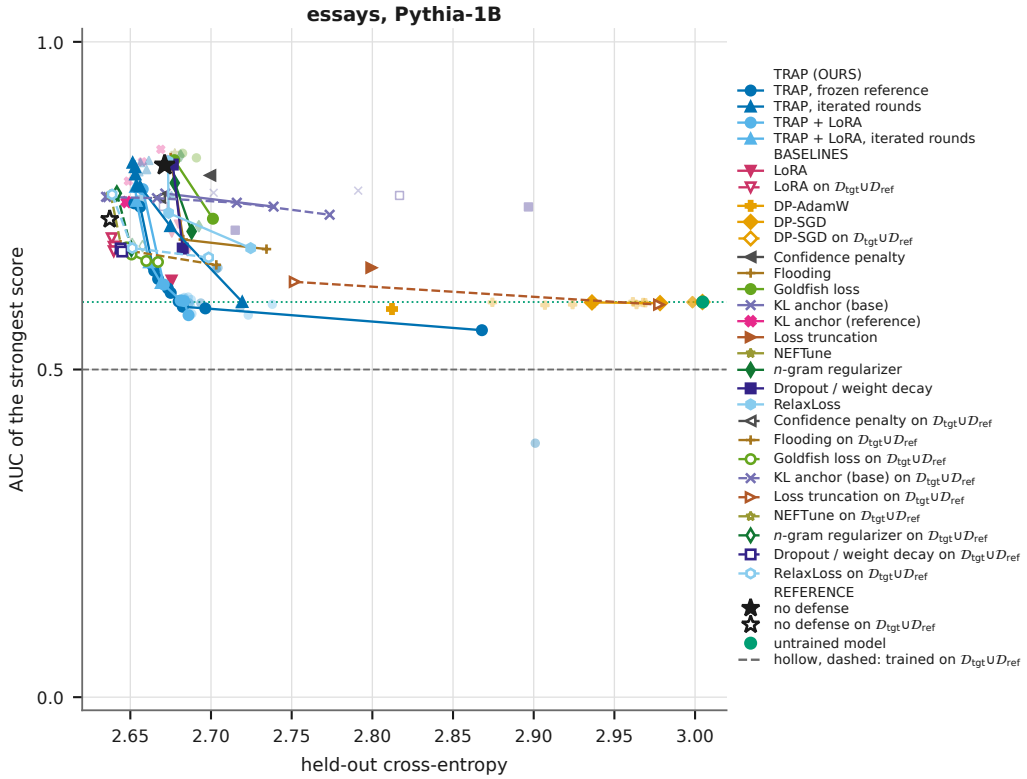


Figure 38: Directional AUC version of Figure 29: *essays* at Pythia-1B.

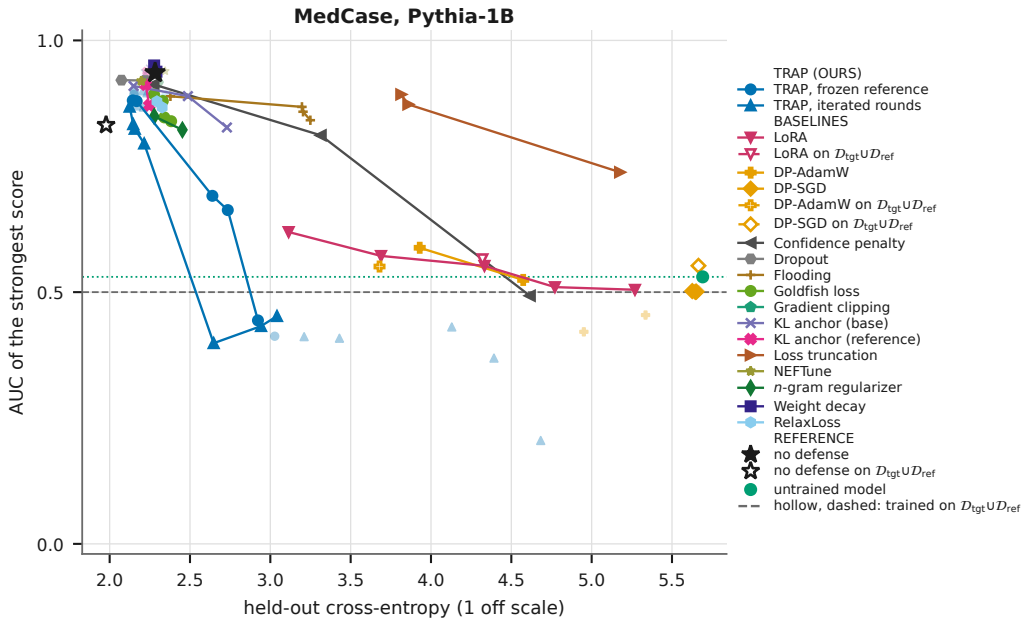


Figure 39: Directional AUC version of Figure 30: *MedCase* at Pythia-1B.

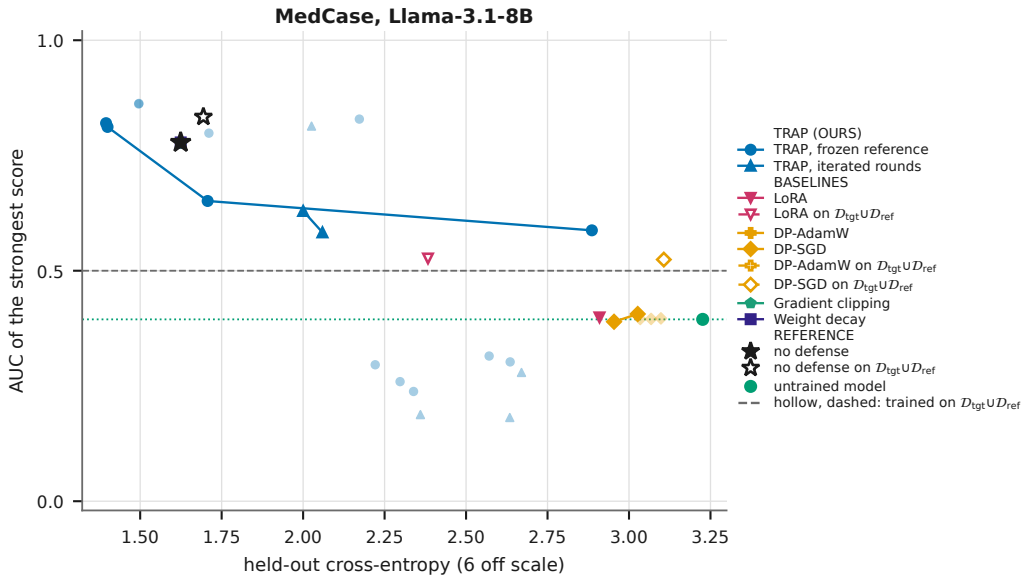


Figure 40: Directional AUC version of Figure 31: *MedCase* at Llama-3.1-8B-Instruct.

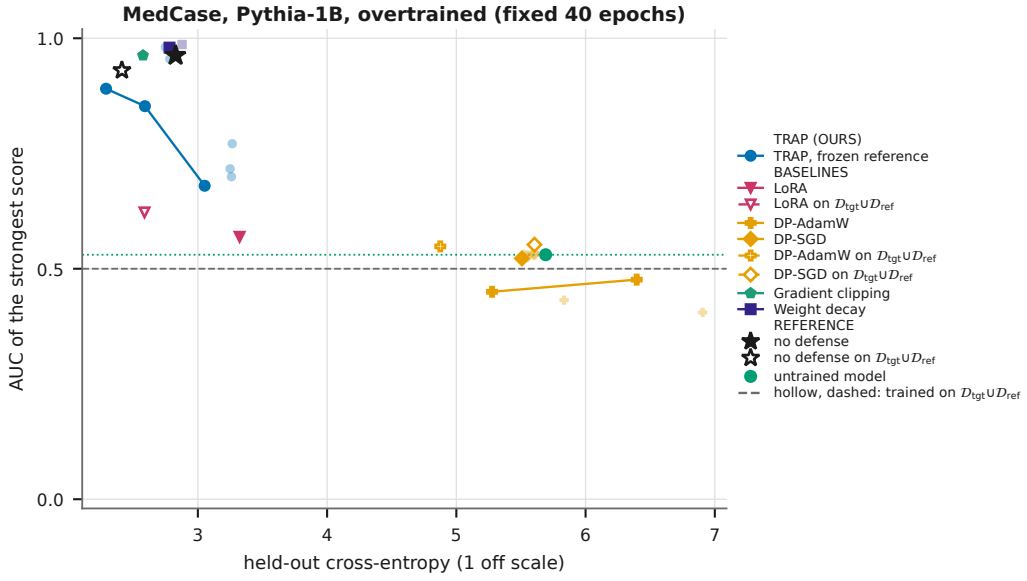


Figure 41: Directional AUC version of Figure 32: *MedCase* at Pythia-1B trained for a fixed forty epochs.

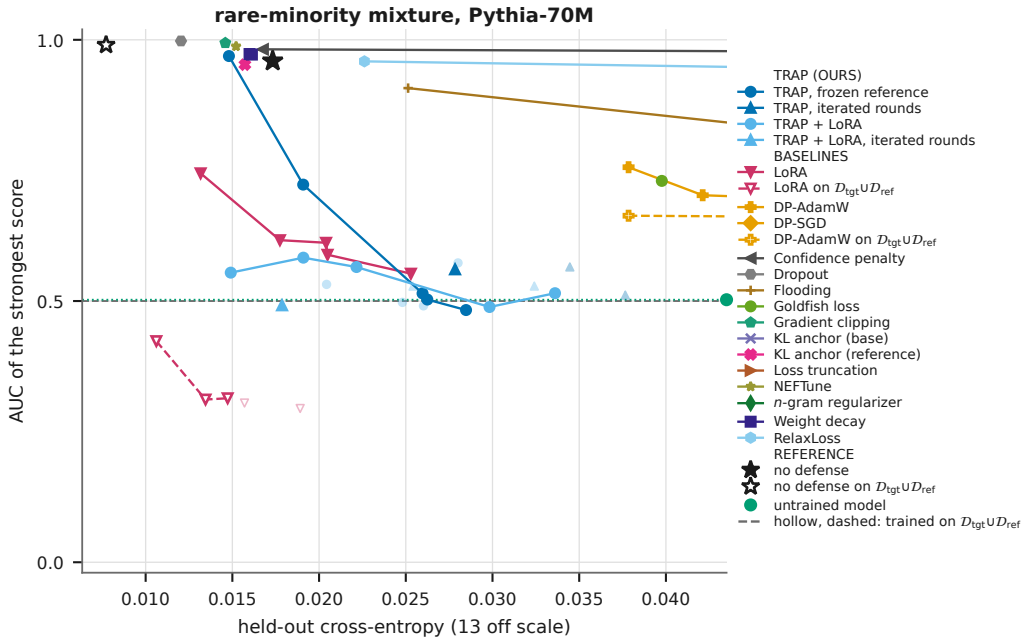


Figure 42: Directional AUC version of Figure 33: the rare-minority mixture at Pythia-70M.

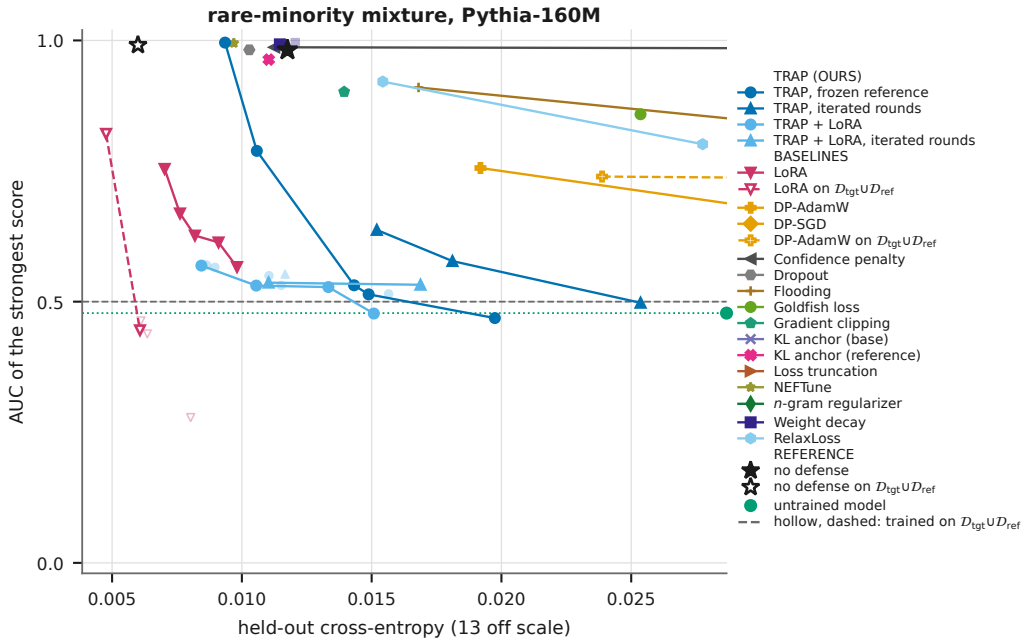


Figure 43: Directional AUC version of Figure 34: the rare-minority mixture at Pythia-160M.

F REFERENCE SCHEDULES

Equation 3 defines the penalty but not where the reference comes from. We use two schedules, and either can train the full model or a LoRA adapter (TRAP + LoRA in Section 4).

Frozen reference. Train an ordinary model on $\mathcal{D}_{\mathrm{ref}}$ with the testbed's recipe, keep its best-val checkpoint, freeze it, and train the target from the pretrained initialization on $\mathcal{D}_{\mathrm{tgt}}$ with Equation 3. The reference does not change and never sees the target's examples, which makes this the cleanest schedule for the bound of Equation 22, which assumes a fixed reference.

Iterated rounds. The first round is frozen-reference training (on the *essays* a few round-2 cells were instead seeded by an alternating or mutual first round, or by the undefended model; the tables mark them). Each later round freezes the best-val checkpoint of the previous round as the new reference, resets the target to the pretrained initialization, swaps the roles of the two halves, and repeats the sweep over the penalty weight; we ran up to five rounds, and every round enters the sweep as its own setting. The reference thus tracks how far the target has got with the task. After the first swap, however, the reference depends on the target's half through its own reference in the round before, so the gaps of later rounds are no longer taken against a model that never saw the target's data. This has a failure mode. If a round's target has memorized a value that also occurs on the other half, that model becomes the next reference and gives the value a high probability there too; the next target's advantage on it then falls below the threshold, and the penalty stops firing on exactly the token it should catch. Unique secrets, as in our testbeds, do not have this problem; recurring ones do, and for them the frozen reference, which never sees the target's half, is the schedule to use. This is also why the bound of Equation 22 is read for the final pair of models rather than for a schedule.

F.1 OTHER REFERENCES, AND WHY THEY ARE NOT USED

On the *essays* we also trained TRAP against every other reference we could think of, with the same recipe and the same sweep over the penalty weight. Lockstep: both models start from the pretrained checkpoint and train side by side, the reference on $\mathcal{D}_{\mathrm{ref}}$ with the ordinary loss and the target on $\mathcal{D}_{\mathrm{tgt}}$ with Equation 3, so the reference is at the same stage of training as the target but never sees its examples. Mutual: both models train with Equation 3, each using the other as its current reference. Alternating: two persistent models swap the target and reference roles every fixed number of steps within one run; in the asymmetric version the model meant for release gets a larger penalty weight than the one meant to serve as reference. Pretrained base: the unmodified base model, which has seen neither half, is the reference; the larger version uses the pretrained Pythia-1B-deduped, which has likewise seen neither half, as the reference for the 70M target.

Figure 44 shows the result. The frozen reference sets the frontier at both scales, with the iterated rounds close behind. Lockstep reaches the same floor, but at 70M only after giving up utility. Mutual training gets only part of the way (worst-case $\mathrm{nAUC}\ 0.46$ at 70M against 0.71 undefended, and no lower than the undefended model at 1B): when both models raise the same token probabilities together, the gap closes without the memorization going away. Alternating reaches the floor only near the cross-entropy of the untrained model, and its asymmetric version stops halfway. The pretrained base is the worst reference: it discounts only what any language model would predict, so the penalty falls on everything the task teaches, and the target loses utility while its memorization stays close to the undefended model; a larger pretrained reference helps a little and does not change the picture. The conclusion is the one Section 2.1 argues for: the reference has to be trained on the same task and must not have seen the target's examples, and the frozen reference is the simplest schedule with both properties.

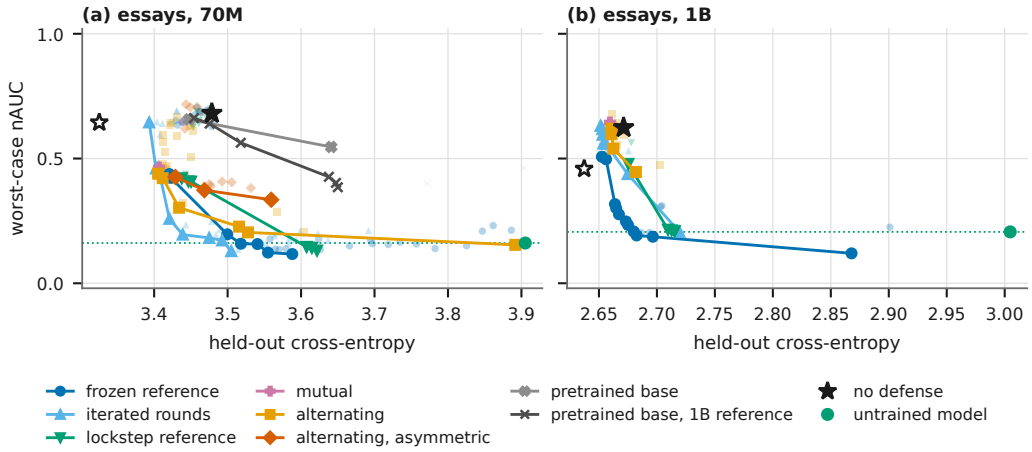


Figure 44: TRAP with different references on the *essays* at Pythia-70M (a) and 1B (b), best-val checkpoint: worst-case nAUC against held-out cross-entropy, one lower-left frontier per reference with its settings faint, the undefended model as the star and the untrained model as the green dot and dotted line. The asymmetric alternating and the two base references were run at 70M only.

G Properties of TRA and TRAP

This appendix states and derives the three properties that Section 2 uses: the expected TRA of a training example is the expected generalization gap, the penalty gates each token's gradient, and tokenwise control bounds what the target can assign to a whole continuation relative to the reference.

G.1 Expected TRA equals the expected generalization gap

For this statement, let $\mathcal{J}_x$ contain every token the model is trained on, that is, all the tokens the usual fine-tuning loss covers, rather than only the sensitive positions used in evaluation, and define the sample negative log-likelihood as

$$\ell(\theta, x) := -\frac{1}{|\mathcal{J}_x|} \sum_{t \in \mathcal{J}_x} \log p_\theta(x_t \mid x_{<t}). \tag{18}$$

Suppose the corpus contains $2n$ IID samples from a distribution $\mathcal{P}$, the balanced partition into $\mathcal{D}_{\text{tgt}}$ and $\mathcal{D}_{\text{ref}}$ is drawn independently of their values, and the same randomized learning procedure is applied to each split. After training, draw $X_{\text{train}} \mid \mathcal{D}_{\text{tgt}} \sim \text{Unif}(\mathcal{D}_{\text{tgt}})$ independently of the learner randomness. Combining Equation 2 with the loss definition gives

$$\text{TRA}(X_{\text{train}}) = \ell(\theta_{\text{ref}}, X_{\text{train}}) - \ell(\theta, X_{\text{train}}). \tag{19}$$

Although X_{train} is used to train θ, it is absent from $\mathcal{D}_{\text{ref}}$ and therefore independent of θ_{ref}; marginally, it is distributed as $\mathcal{P}$. If $X_{\text{test}} \sim \mathcal{P}$ is independent of the data and both learners, split symmetry also gives $\theta_{\text{ref}} \overset{d}{=} \theta$. Consequently,

$$\begin{aligned}
\mathbb{E}[\text{TRA}(X_{\text{train}})] &= \mathbb{E}_{\theta_{\text{ref}}} \mathbb{E}_{X_{\text{test}} \sim \mathcal{P}} \ell(\theta_{\text{ref}}, X_{\text{test}}) - \mathbb{E}_{\mathcal{D}_{\text{tgt}}, \theta} \frac{1}{n} \sum_{x \in \mathcal{D}_{\text{tgt}}} \ell(\theta, x) \\
&= \mathbb{E}_{\mathcal{D}_{\text{tgt}}, \theta} \left[\mathbb{E}_{X_{\text{test}} \sim \mathcal{P}} \ell(\theta, X_{\text{test}}) - \frac{1}{n} \sum_{x \in \mathcal{D}_{\text{tgt}}} \ell(\theta, x) \right].
\end{aligned} \tag{20}$$

The expected full-sequence TRA on target training examples is therefore exactly the expected generalization gap, the test loss minus the training loss. The span-restricted TRA used in our evaluation is the corresponding localized diagnostic on the sensitive tokens.

The identity needs the two models to be trained the same way. Once the target is trained with the penalty of Equation 3 and the reference is not, $\theta_{\text{ref}} \overset{d}{=} \theta$ no longer holds, so during TRAP training TRA is a training signal rather than an estimate of the gap.

G.2 The penalty gates each token's gradient

Let

$$g_{x,t} = -\nabla_\theta \log p_\theta(x_t \mid x_{<t})$$

denote the ordinary next-token gradient contributed by token t. Away from the threshold $\mathrm{TRA}_t(x) = \epsilon$, the corresponding contribution under Equation 3 is

$$\nabla_\theta[-\log p_\theta(x_t \mid x_{<t}) + w\ \mathrm{ReLU}(\mathrm{TRA}_t(x) - \epsilon)] = (1 - w\mathbf{1}\{\mathrm{TRA}_t(x) > \epsilon\})\, g_{x,t}, \tag{21}$$

since the reference term has no dependence on θ. Tokens below the threshold receive their ordinary gradient. Above it, that contribution is attenuated when $0 < w < 1$, removed when $w = 1$, and reversed when $w > 1$. This describes the raw contribution of an individual token; minibatch aggregation, gradient clipping, momentum and Adam state alter the resulting parameter update. The point of comparison with the baselines of Section 4 is the criterion: TRAP changes a token's contribution only when its TRA exceeds the threshold, whereas random token dropping or norm-based gradient clipping decide which signal to modify on grounds that have nothing to do with how target-specific the token is.

G.3 A reference-relative bound on continuations

Tokenwise control also bounds complete continuations. For any contiguous span $\mathcal{I}$ of length L,

$$\log \frac{\prod_{t\in\mathcal{I}} p_\theta(x_t \mid x_{<t})}{\prod_{t\in\mathcal{I}} p_{\theta_{\text{ref}}}(x_t \mid x_{<t})} \le L\epsilon + \sum_{t\in\mathcal{I}} \mathrm{ReLU}(\mathrm{TRA}_t(x) - \epsilon)\,, \tag{22}$$

which follows by writing the left side as $\sum_{t\in\mathcal{I}} \mathrm{TRA}_t(x)$ and bounding each term by $\epsilon + \mathrm{ReLU}(\mathrm{TRA}_t(x) - \epsilon)$. If $\mathrm{TRA}_t(x) \le \epsilon$ at every position of the span, the target assigns that continuation at most $e^{L\epsilon}$ times the probability the reference does, and because the hinge is taken token by token, one token far above the threshold cannot be offset by tokens below it. The bound is only as good as the reference: if the reference itself gives a span more probability than a model that never saw it would, that excess adds to the exponent and TRAP does not control it. It is a reference-relative predictive bound for the two models as trained, on the continuations one checks, not a differential-privacy guarantee.