Pandora's White-Box: Precise Training Data Detection and Extraction in Large Language Models

Jeffrey G. Wang [*1] **Marvin Li** [*1] **Jason Wang** [*1]
Seth Neel [2]

Abstract

In this paper we develop state-of-the-art privacy attacks against Large Language Models (LLMs). Our headline results are new membership inference attacks (MIAs) against pretrained LLMs hundreds of times better than baseline attacks, and a pipeline showing over 50% (!) of the fine-tuning dataset can be extracted from in natural settings. We consider varying degrees of access to the underlying model, pretraining and fine-tuning data, and both MIAs and training data extraction. For pretraining data, we propose a supervised neural network MIA that predicts membership with (dimensionality-reduced) model gradients, closing the gap between MIA attack success against LLMs and the strongest known attacks for other machine learning models. For fine-tuning data, we find a loss ratio between base and fine-tuned models achieves near-perfect MIA performance and can extract a large fraction of the fine-tuning dataset from Pythia and Llama models. Taken together, these results represent the strongest existing privacy attacks against both pretrained and fine-tuned LLMs for MIAs and training data extraction, which have important practical implications for LLM security, privacy, and copyright issues.

1. Introduction

As large language models (LLMs) are increasingly trained and fine-tuned on potentially sensitive data in domains ranging from financial services to healthcare, the issue of *privacy* has emerged as an important consideration in the responsible deployment of these models. Recent work in the ML community has probed the privacy properties of LLMs, focusing on two types of privacy attacks that have been widely studied in classification and regression settings (Liu et al., 2021) and are especially pertinent to LLMs: membership inference attacks (MIA) (Shokri et al., 2017), and training data extraction (Carlini et al., 2020). In an MIA, an adversary with some level of access to the model seeks to reliably distinguish training samples from test samples. Highly accurate MIAs against LLMs would be useful not only to demonstrate privacy leakage, but for other applications like detecting train/test contamination and verifying if copyrighted data was used to train a given model. For example, suppose an LLM company wants to run a new evaluation on a previously trained model and needs to verify that the examples seen during evaluation were not already seen during training. Rather than querying the gargantuan database of web text used to train the model, they could use an MIA to estimate the probability a model was trained on that sample. On the flip side, suppose a copyright holder suspects that their protected asset had been used to train a popular LLM. Since the training datasets of these models are typically not released, partially due to copyright liability, the only ways to argue conclusively that their data had been used during training would be with an MIA, or by prompting the model to regurgitate the training data verbatim as the New York Times claims they did in their recent lawsuit (Kang & Metz, 2023). MIAs have also found use as a way to quantify other privacy properties of ML models, including memorization during training (Zhou et al., 2023), and empirical evaluation of unlearning success (Kurmanji et al., 2023; Pawelczyk et al., 2023; Hayes et al., 2024).

Despite a recent string of papers studying MIAs against LLMs (Duan et al., 2024; Li et al., 2023; Mattern et al., 2023), existing MIAs barely outperform random guessing when evaluated on aggregate, and some actually perform *worse than random guessing* at low FPRs. The latter metric, the True Positive Rate (TPR) at low False Positive Rate (FPR), is particularly important because any attack which can extract even a small subset of the training data with high confidence poses a serious privacy risk, and is used widely in the literature (Carlini et al., 2022). In fact, this failure across a series of MIAs prompted (Duan et al., 2024) to speculate that the failure of MIAs against pretrained LLMs

[*]Equal contribution [1]Harvard College [2]Harvard Business School. Correspondence to: Seth Neel <sneel@hbs.edu>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

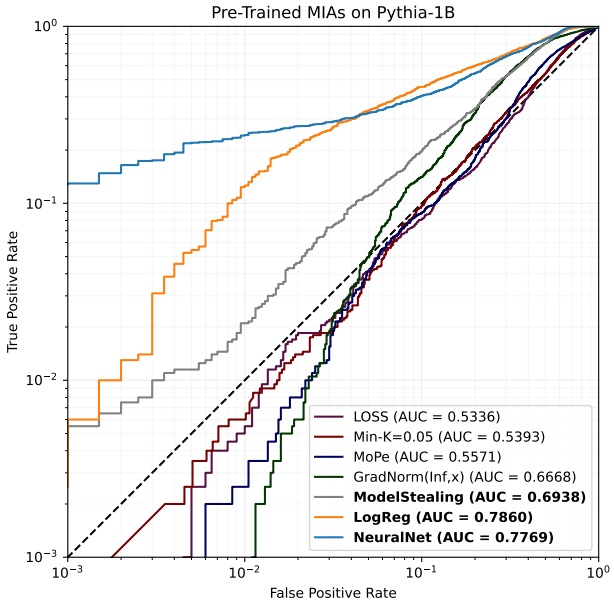


Figure 1. New attacks **NeuralNet**, **LogReg**, & **ModelStealing**.

might be an inherent property of the training process and distribution of train/test examples rather than a shortcoming of existing attacks.

In this work we develop the first strong MIAs against pretrained LLMs, with TPRs at low FPRs that are hundreds of times better than baseline attacks (Figure 1). See Table 5 for full results.

The technique we use to develop our most effective attacks is to train a supervised attack that predicts training dataset membership given access to a very small ($<.001\%$) subset of training and test data. As input to the classifier we use features derived from the model gradient $\nabla_\theta \ell(\theta, x)$, as well as the gradient with respect to the input embedding. This approach is similar to that of (Nasr et al., 2019) who train supervised white-box attacks against classifiers, but with significant additional complexity due to the high dimensionality of model gradients in LLMs. The technique we use to develop our most effective attacks is to train a supervised attack that predicts training dataset membership given access to a very small ($<.001\%$) subset of training and test data. As input to the classifier we use features derived from the model gradient $\nabla_\theta \ell(\theta, x)$, as well as the gradient with respect to the input embedding. This approach is similar to that of (Nasr et al., 2019) who train supervised white-box attacks against classifiers, but with significant additional complexity due to the high dimensionality of model gradients in LLMs.

Many of the most performant models like GPT-4o or Claude 3 remain closed source, and supervised MIAs on the gradients cannot be implemented against them. In Section 3.2 we

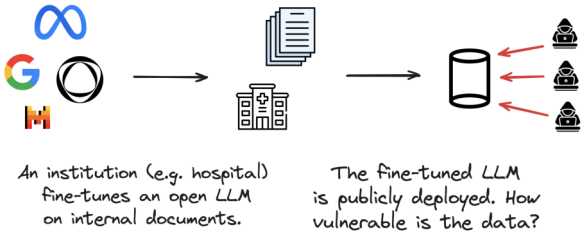


Figure 2. Deploying LLMs fine-tuned on internal data poses serious security risks, as we will show.

leverage the recent model stealing attack of (Carlini et al., 2024) where, given logit access to the model (e.g. via an API), one can recover the projection layer up to a rotation; we use this technique to conduct a less restrictive "gray-box" variant of our supervised attack. This represents, to the best of our knowledge, the first MIA with a model-stealing component. Figure 1 shows this attack (`ModelStealing`) outperforms all prior attacks, especially in low FPR regimes.

Next, we consider fine-tuned models. These are increasingly relevant from a privacy perspective since companies and other institutions are increasingly fine-tuning open source pretrained language models on sensitive internal data rather than training their own LLM from scratch. Prior work on memorization finds that data seen more recently during training is more vulnerable to membership inference (Zhou et al., 2023), and other work on MIAs against classifiers shows that access to multiple checkpoints can be combined to boost MIA attack accuracy (Jagielski et al., 2022). It is thus unsurprising that fine-tuned MIAs might be more vulnerable to privacy attacks. What was unknown, and what our results in Section 3.3 show, is that a simple MIA based on the ratio of the loss on the fine-tuned model to the pretrained model is *nearly perfect* at membership inference against fine-tuned LLMs. In Section 4, we turn to *data extraction*, where rather than predicting training set membership, the adversary wishes to extract verbatim training examples.

Using our MIAs in the generate-then-rank data extraction pipeline for LLMs (Carlini et al., 2020), in Section 4.1.2 we demonstrate that *significant* portions of the fine-tuning dataset ($>50\%$ after just 3 epochs of fine-tuning) can be efficiently extracted with just 20 generations, with even more extractability at larger model sizes and more generations. Our results show strong, realistic extraction attacks are possible on fine-tuned LLMs, with only black-box access.

In the pretrained setting, data extraction is more difficult due to lower levels of memorization; furthermore, even our strong supervised white-box attacks prove to be poor at distinguishing pretraining samples from generated samples (Section 4.2). Nevertheless, we find that if we directly train an attack to distinguish *generated samples* from train-

ing samples in the style of our white-box attacks, we are able to effectively identify training samples when they are generated.

2. Preliminaries and Related Work

Denote an autoregressive language model over a vocabulary $\mathcal{V}$ by $\theta : \mathcal{V}^* \to \Delta(\mathcal{V})$ and the autoregressive loss $\ell(\theta, x) = \sum_{t=1}^{T} -\log p_\theta(x_t|x_{<t})$. In this section, we define our two primary privacy attacks: membership inference attacks (MIAs) and training data extraction. Then, we outline the different levels of model access an adversary might have, and our methods for evaluating these privacy attacks on LLMs.

Definition 2.1 (Membership Inference Attack)**.** A regressor $\mathcal{M} : \mathcal{X} \times \theta \to \mathbb{R}$ is a membership inference score (Shokri et al., 2017) that outputs low values when $x \in \mathcal{X}_{\texttt{TRAIN}}$ and high values otherwise.

Given a membership inference score, for any threshold τ, there is a corresponding membership inference attack that predicts $x \in \mathcal{X}_{\texttt{TRAIN}}$ if and only if $\mathcal{M}(x, \theta) < \tau$. We note that in practice $\mathcal{M}$ may not have white-box access to the model parameters θ, which we discuss shortly.

Evaluation. Like (Li et al., 2023), we evaluate every MIA in this paper using 2,000 train/test points, on models across the Pythia suite (Biderman et al., 2023). In pre-training we are limited to evaluating models that have publicly released weights and training data, and we use Pythia's train/validation split of the The Pile to evaluate our MIAs (Gao et al., 2020). In the fine-tuning setting, we also fine-tune and evaluate the Llama-7B and Llama-7B-chat models. We follow standard conventions for evaluating MIA efficacy (Carlini et al., 2022; Shokri et al., 2017; Ye et al., 2022): we report full ROC curves with the AUC as well as achievable TPRs at small FPR rates τ. For every evaluation metric/plot in this paper, we provide 95% confidence intervals computed over 1,000 bootstraps.

MIA Background. MIAs against machine learning models were first proposed by (Shokri et al., 2017). (Yeom et al., 2018) proposed thresholding on $\mathcal{M}(x, \theta) = -\ell(x, \theta)$, which we refer to throughout as $\texttt{LOSS}_\theta$. Current state of the art MIAs (Ye et al., 2022; Carlini et al., 2022) calibrate the threshold τ in an example-specific manner. One such attack is called the likelihood ratio attack; in this attack, adversaries train shadow models with/without the candidate point and use the loss of these models to approximate the likelihood that the sample is a train point vs. the likelihood it was not. (Carlini et al., 2022) argue this attack is approximately optimal by the Neyman-Pearson Lemma, although it is computationally infeasible to apply to pretrained LLMs because it requires the training of many (often large) shadow models. A simpler attack, which only requires having access to a model θ_{Base} that the adversary knows a point has not been trained on, is the *loss ratio*: thresholding over $\frac{l(\theta,x)}{l(\theta_{\text{Base}},x)}$. Such an attack easily lends itself to a fine-tuned setting where we can query the base model; we call this **f**ine-tuned **lo**ss **ra**tio $\texttt{FLoRa}_{\theta,\theta'}$. In settings without a base model, `zlib` entropy can be used in the denominator instead (Carlini et al., 2020), which we call $\texttt{zlib\_LoRa}_\theta$.

In setting where $\mathcal{M}$ has access to θ and a small dataset of train and test samples, (Nasr et al., 2019) first proposed directly training a supervised MIA using the gradient $\nabla_\theta \ell(\theta, x)$ as an input feature. They find it is a highly effective MIA against vision classifiers. They also show that even without training a supervised attack, the gradient norms $||\nabla_\theta \ell(\theta, x)||$ themselves are highly correlated with train set membership. To the best of our knowledge, thresholding on gradient norms has not been evaluated for LLMs; we do so in Section 3 and illustrate that it's a performant baseline. Our white-box attacks in Section 3.1 can be viewed as an extension of the attacks in (Nasr et al., 2019) to LLMs, but because training a supervised classifier directly on the gradients is infeasible for LLMs, we employ gradient norms layerwise as a means of *dimensionality reduction*.

MIAs on LLMs. Specific MIAs have been tailored to LLMs. $\texttt{MoPe}_\theta$ is a white-box MIA that approximates the trace of the Hessian of the model loss with respect to model parameters (Li et al., 2023). $\texttt{Min-K}_\theta$ is a black-box MIA that thresholds the average loss of the $k\%$ of tokens with smallest loss in a candidate input (Shi et al., 2023). (Mattern et al., 2023) propose an attack that measures the drop in the log-likelihood after perturbing the input, which (Li et al., 2023) find to be worse than $\texttt{MoPe}_\theta$ across all model sizes. (Duan et al., 2024) evaluate the above attacks on the Pythia suite, corroborating the findings in (Li et al., 2023) and in this paper that none of these methods significantly outperform random guessing for large model sizes. Concurrent with our work, (Zhang et al., 2024) proposed an MIA for LLMs inspired by (Mattern et al., 2023) and (Shi et al., 2023) which achieves AUCs ranging from 0.52 to 0.59 across Pythia model sizes and TPRs 1-2x FPR at FPR=5%.

Model Access. We study MIAs that require three distinct levels of access to the model θ throughout this paper, which we call the black-box, gray-box, and white-box settings. In the *black-box* setting, the MIA can evaluate the model loss $\ell(x, \theta)$ and generate samples from the model—the kind of access afforded by standard APIs. In the *white-box* setting, the MIA has access to θ directly or to other statistics like model gradients (we discuss this finer point in Section 5). Black-box attacks include $\texttt{LOSS}_\theta$ and $\texttt{Min-K}_\theta$. White-box baselines include $\texttt{MoPe}_\theta$ (needs model weights) and $\texttt{GradNorm}_p$ (needs model gradients). Our state-of-the-art attacks, $\texttt{LogReg}_\theta$ and $\texttt{NN}_\theta$, are white-box. Our new model-stealing attack is *gray-box*, as it only requires the

Figure 3. Pretrained MIA access levels.

ability to compute model logits. Notably, the assumption is weaker still in practice: (Carlini et al., 2024) note that their model stealing attack can be implemented by an adversary who only has access to the top-K log probabilities and can specify a logit bias. Finally, in the fine-tuned setting, we evaluate $\texttt{zlib\_LoRa}_\theta$, $\texttt{LOSS}_\theta$, and introduce $\texttt{FLoRa}_{\theta,\theta'}$. All are *black-box* attacks.

Next, we define the canonical attack and definitions we use in the training data extraction sections and adapt the definition of discoverable memorization (Nasr et al., 2023) to the setting of data extraction.

Definition 2.2 (Generate-then-Rank Extraction Attack)**.** Given a prompt/prefix $\mathbf{a}|\mathbf{b}$ and score $\mathcal{M}$, we define a generate-then-rank extraction attack $\texttt{Extract}_\mathcal{M}$ (Carlini et al., 2023a; Nasr et al., 2023; Ishihara, 2023) as the following procedure: prompt the model with prefix $\mathbf{a}$ and generate n suffixes $\mathbf{x}_1 \dots \mathbf{x}_n$. Rank the generations using $\mathcal{M}$, output the top ranked generation $\mathbf{x}_{i*}$, where $i^* = \text{argmax}_i \mathcal{M}(\mathbf{a}, \mathbf{x}_i)$.

Definition 2.3 ((k,m)-Discoverable Extraction).**.** Given a model θ, score function $\mathcal{M}$, and a sample $\mathbf{ab}$, where $k = \text{len}(\mathbf{a})$ and $m = \text{len}(\mathbf{b})$, we say that a sample is (k,m)-discoverably extractable if $\texttt{Extract}_\mathcal{M}(\mathbf{a}) = \mathbf{b}$.

Different values of k and m correspond to different assumptions about a possible attack. Like (Carlini et al., 2023b), we consider one setting where the adversary knows a significant chunk of the string to extract ($k = m = 50$) and refer to this setting as "discoverable extraction." As k gets lower, however, an adversary may be able to more realistically know the beginning of a sample (or guess it from commonly-occurring Internet text, as (Nasr et al., 2023) do). For example, small values of k may correspond to an adversary knowing the patient name that corresponds to a given medical record they want to extract. In this paper, we consider examples of this setting where $k \in \{2, 4, 8\}$ and $m \in \{25, 50\}$; we refer to this setting as "non-discoverable extraction." We use these names for these two settings often throughout this paper.

3. Membership Inference Attacks

In this section, we present the results of several new and existing MIAs for detecting pretraining data (Subsections 3.1, 3.2) and fine-tuning data (Subsection 3.3). We apply these methods to the Pythia suite of models (Biderman et al., 2023) from size 70M-6.9B in pretraining, and for fine-tuning add Llama-7B and Llama-7B-chat models. We discuss and benchmark the baselines further in Appendix C. Our main findings are that access to a very small ($< .001\%$) random sample of the training dataset, along with the ability to compute model gradients, enables the training of incredibly strong supervised MIAs against pretrained models based on gradient norms ($\texttt{LogReg}_\theta$ and $\texttt{NN}_\theta$). In addition, we demonstrate that with much weaker model access to just gray-box logits, a supervised attack can still obtain competitive MIA performance via a model stealing paradigm (Carlini et al., 2024). In the fine-tuning setting, a simple black-box MIA that exploits the fact that the likelihood of samples seen during fine-tuning nearly always increases more relative to the base model likelihood than samples not including in fine-tuning, achieves near perfect MIA accuracy.

3.1. Supervised White-Box Attacks

In this section, we try to find the upper limit of training data extraction from a pretrained LLM: we imagine that the adversary has full white-box access to the model, as well as a random sample of known training and test data with which to calibrate their attack. Given labeled training and test samples, and model θ, there is an obvious approach to building a strong MIA: train a binary classifier $h : \mathcal{X} \times \Theta \to \{0, 1\}$ to predict the train or test label, given x, θ. One clear difficulty here is that dimensionality of θ in the LLM setting is too large to train a classifier with input θ directly, and so we replace the inputs x, θ with a collection of feature statistics $(f_1(x,\theta), f_2(x,\theta), \dots f_n(x,\theta))$. Of course, this will only be an effective strategy to the degree that the features $f_i(x,\theta)$ capture valuable information about whether x is a training point of θ.

Taking inspiration from promising MIA results based on thresholding the norm of the full model gradient at a given point (see the $\texttt{GradNorm}_p$ results in Appendix D), we use the various p-norms of the gradient of loss with respect to θ at different layers in the network as well as the p-norms of the gradient with respect to x, the input embedding of a prompt, as our features ($p \in \{1, 2, \infty\}$). To be more concrete, if $\{\theta_t\}_{t=1}^L$ are the weights at L layers and $\phi : \mathcal{V}^* \to \mathbb{R}^h$ maps prompts to the input embeddings, the features for a prompt p are $\{\{\|\nabla_{\theta_t} \ell(\theta, p)\|_r)\}_{r \in \{1,2,\infty\}}\}_{t=1}^L$ and

$\{\|\nabla_{\phi(p)}\ell(\theta,p)\|_r\}_{r\in\{1,2,\infty\}}$. This results in roughly 450 features for the smallest 70M parameter model and 1170 for the largest model at size 6.9B. We assume the adversary has access to 10,000 training sequences each of length 2,048, and 10,000 test examples. Practically, a training set subset of this size for a pretrained LLM is not difficult to guess even if it is not explicitly available—many are trained on Wikipedia, for instance. Test examples can be harder to guess given the size of the training sets—we are able to take advantage of clean train/test splits on The Pile here—but could be constructed e.g., based on knowledge of an LLM's release date.

For all model sizes, we train a logistic regression $\texttt{LogReg}_\theta$ and a MLP $\texttt{NN}_\theta$, and present the AUCs and TPRs in Figure 4, with more experimental details and results results in Appendix E. Figure 1 shows the ROC curves for all methods on the 1B parameter model, with more details in Appendix F. Relative to all black-box attacks studied, both supervised attacks are highly accurate, achieving AUCs $> .7$ and high TPRs at low FPRs. For Pythia-1B, $\texttt{NN}_\theta$ achieves a 22% TPR at FPR of 0.5%, representing a $44\times$ increase over random guessing, and over $100\times$ other baselines such as $\texttt{LOSS}_\theta$ which gets 0.25%. Thus, the vast majority of data the $\texttt{NN}_\theta$ predicts is training data is *actually* training data.

Attack Data Efficiency. One question about a supervised MIA's practicality arises in how much training data is necessary for the attack to succeed. We thus repeat our $\texttt{LogReg}_\theta$ and $\texttt{NN}_\theta$ experiments by varying the number of training data samples between 100 and 10,000, and find that around 1,000 points is already sufficient to get high AUCs and TPRs at low FPRs, or about $1/100{,}000$ of the entire training set (Biderman et al., 2023). Full details are available in Appendix G.1.

Feature Ablation. While we chose model gradient norms as our features to feed into our trained classifier based on the success of $\texttt{GradNorm}_p$, we also show that this is a reasonable choice over other alternatives. A good set of features drastically reduces the dimension of the model (e.g., 1,170 features for a 6.9B model is a dimensionality reduction of a factor of $\approx$ 6M) while retaining enough distinguishing information for deducing membership in the train set. We consider two other dimensionality reduction techniques in Appendix G.3, and we find that our layerwise norm features are better than these candidate features.

While the requirement for a small sliver of training data to calibrate an attack is easy to justify in practice, access to the model to create gradient features is not always possible. Next, we show that we can relax this assumption, albeit at the expense of some attack accuracy.

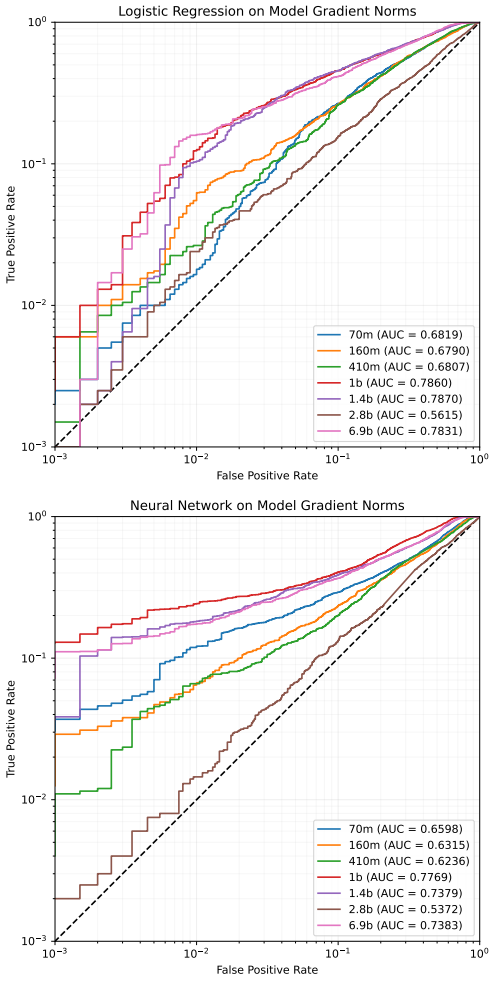


Figure 4. AUCs of $\texttt{LogReg}_\theta$ and $\texttt{NN}_\theta$ trained on norms of layers for Pythia-70M to 6.9B.

3.2. Gray-Box Attacks via Model-Stealing

By leveraging recent advances in model-stealing in (Carlini et al., 2024), we can implement variants of $\texttt{NN}_\theta$ and $\texttt{LogReg}_\theta$ in the gray-box setting where an adversary only has standard API access to the top-K log-probabilities and does not have access to the underlying weights. (Carlini et al., 2024) shows that using only standard API access to top-K softmax log-probabilities for production LLMs, one can steal the output logits for a given prompt, embedding dimension size h, and the *embedding projection layer* $\mathbf{W} \in \mathbb{R}^{|\mathcal{V}|\times h}$ up to some symmetries (see Appendix G.4). The crux here is the observation that given access to the logits and $\mathbf{W}$ up to symmetries, we can compute the gradient with respect to $\mathbf{W}$ (up to symmetries) and use that to train last-layer variants of $\texttt{NN}_\theta$ and $\texttt{LogReg}_\theta$. **This attack achieves state-of-the-art AUCs among black- or gray-box MIAs on pretrained LLMs.**

Method. Assume we have an oracle $\mathcal{O} : \mathcal{V}^* \to \mathbb{R}^{|\mathcal{V}|}$ that maps prompts to logits and, following (Carlini et al., 2024), have used a SVD decomposition of logits to compute $\tilde{\mathbf{W}} = \mathbf{W} \cdot \mathbf{A}$, for some unknown invertible $\mathbf{A} \in \mathbb{R}^{h \times h}$. For a prompt p, we can calculate the gradient of loss with respect to the outputs logits, $\nabla_{\mathcal{O}(p)} \ell(\theta, p)$, by taking the derivative of the softmax. We take the outer product between this gradient and the embedding of $\mathcal{O}(p)$ using $\tilde{\mathbf{W}}$, returning $\nabla_{\tilde{\mathbf{W}}} \ell(\theta, p) = \nabla_{\mathcal{O}(p)} \ell(\theta, p) \cdot (\tilde{\mathbf{W}}^\dagger \mathcal{O}(p))^\intercal$. By some simple arithmetic, this is equivalent to $\nabla_{\tilde{\mathbf{W}}} \ell(\theta, p) = \nabla_{\mathbf{W}} \ell(\theta, p) \cdot \mathbf{A}^{-\intercal}$, the true gradient of the embedding projection layer modulo RHS symmetries (Lemma G.1). We then use this gradient to derive features to train a supervised attack in the style of Section 3.1. To reduce the dimensionality of the gradient, we compute a Johnson-Lindenstrauss transformation $\Pi : \mathbb{R}^{|\mathcal{V}|} \to \mathbb{R}^{512}$ and apply it to every column of $\nabla_{\tilde{\mathbf{W}}} \ell \in \mathbb{R}^{|\mathcal{V}| \times h}$, producing a tensor $[\Pi((\nabla_{\tilde{\mathbf{W}}} \ell)_{:,i})]_{i=1}^{h}$ of size $\mathbb{R}^{512 \times h}$. As opposed to flattening the gradient and then projecting, this projection scheme makes it easier for the supervised MIAs to learn an implicit representation of the basis change $\mathbf{A}^{-\intercal}$, which is constant across all data, and thus access the true gradient. We also keep the p-gradient norms of $\nabla_{\tilde{\mathbf{W}}} \ell$ as features as before and use the same neural network architecture.

As with before, we train and evaluate these classifiers on $4{,}000$ points with a train/test split of 50-50. Our attacks perform very well in this setting across all model sizes, with a max AUC of 0.69, and achieve state-of-the-art results in the gray-box setting (Figures 18 and 19). However, they still are worse than the white-box supervised MIAs, especially in the low FPR range (Tables 13 and 14). Given that achieving strong attack performance against pretrained LLMs seems to require training a supervised attack using a small amount of known training data and some mechanism to compute model gradients (via model access or model-stealing), we ask next: is this also the case for fine-tuned LLMs, or do simple and highly accurate black-box attacks exist?

3.3. MIAs Against Fine-tuned Models

While the previous section's results suggest white-box access to the model and a subset of training data is needed to execute strong attacks in the pretrained setting, the fine-tuned setting is much simpler. In particular, the simple black-box attack $\texttt{FLoRa}_{\theta,\theta'}$, a loss ratio between the fine-tuned and base model, exhibits near perfect MIA accuracy. Using $1{,}000$ points from The Pile's validation set, we fine-tune Pythia models up to 2.8B parameters, Llama-7B and Llama-7B-chat models for a single epoch. We use the resulting models to compute $\texttt{FLoRa}_{\theta,\theta'}$ statistics on $1{,}000$ points from The Pile not used in fine-tuning, and $1{,}000$ points from the fine-tuning set. Figure 5 shows that even at 70M parameters, $\texttt{FLoRa}_{\theta,\theta'}$ achieves near perfect MIA accuracy, with strikingly high TPR at very low FPRs. Unsurprisingly, similarly good results also hold for the larger Llama-7B models.

Since models memorize points seen recently during training at a higher rate, one might wonder if these strikingly good extraction results are due only to the recency or the MIA leveraging the differences between the fine-tuned and base model in a clever way. To evaluate this, we also run a thresholding attack using $\texttt{zlib\_LoRa}_{\theta}$, as well as the plain fine-tuned loss of a point. We find that $\texttt{zlib\_LoRa}_{\theta}$ and fine-tuned loss perform similarly on aggregate (AUC), although $\texttt{zlib\_LoRa}_{\theta}$ performs much worse at low FPRs. $\texttt{FLoRa}_{\theta,\theta'}$ obtains near perfect membership inference performance. Full results for all three attacks can be found in Appendix G.5.

4. Targeted Training Data Extraction

In this section, we study *targeted extraction*: given model access (black-box or white-box), do efficient and practical attacks that extract training data samples exist? As in the MIA setting, we are interested in how the success and practicality of these attacks changes as we vary model access, pretraining vs. fine-tuning, and other information available to the attacker. Following (Carlini et al., 2020) that studies training data extraction in earlier models (GPT-2), our attacks follow a basic "generate-then-rank" strategy that exploits the propensity for LLMs to memorize their training samples verbatim (Nasr et al., 2023) (Definition 2.2). We first find that $50\%+$ of fine-tuning datasets can be extracted with $\texttt{FLoRa}_{\theta,\theta'}$ as the ranking metric $\mathcal{M}$ with just 2-3 epochs of fine-tuning (a very standard amount to fine-tune). Next, we show that in the pretrained setting, even strong MIAs can be poor ranking metrics for extraction. Nonetheless, given white-box access to the model, we can explicitly train an effective $\mathcal{M}$; thus, there remains substantial risk of training data extraction, although it is more difficult to measure than in the fine-tuning setting.

4.1. Fine-tuned LLMs

We consider the setting where the adversary has both the ability to prompt and generate samples from a fine-tuned LLM, and has black-box access to the loss of the base and fine-tuned models, and so can implement the $\texttt{FLoRa}_{\theta,\theta'}$ MIA. As discussed in Subsection 3.3, this is quite realistic given the prevalence of fine-tuning widely available open foundation LLMs. In this section, we show conclusively that after fine-tuning a base LLM from either the Pythia or Llama model suite (Touvron et al., 2023) for ≥ 3 epochs, the simple strategy of (i) prompting the model 20 times and (ii) selecting the generation with the highest $\texttt{FLoRa}_{\theta,\theta'}$ score as a candidate training sample succeeds in extracting more than half of the fine-tuning set across all models. Subsection 4.1.1, where we report discoverable extraction, can

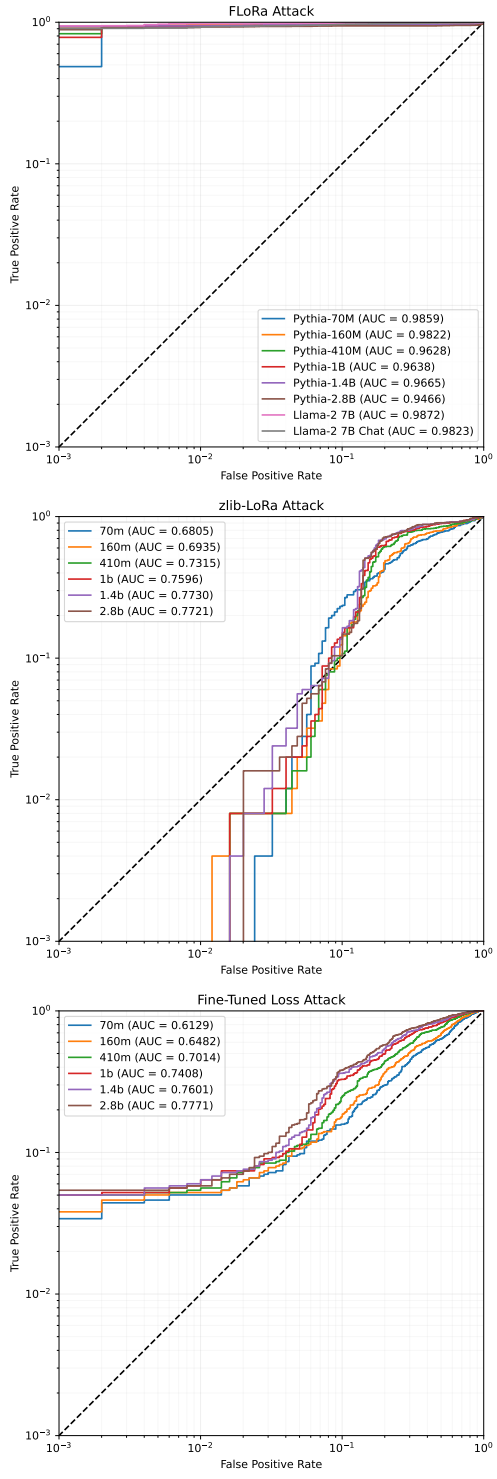


Figure 5. ROC curves of $\texttt{FLoRa}_{\theta,\theta'}$, $\texttt{zlib\_LoRa}_{\theta}$, and fine-tuned loss as MIAs. Plots are log-scaled.

really be viewed as the "upper bound" of extractability: the adversary has access to a near-perfect MIA, can prompt the model with the first $k = 50$ tokens of the example, and the memorization on the finetuning set is much higher than for pretraining data. In Subsection 4.1.2 we consider settings with as few as $k = 2$ tokens.

To evaluate extraction, we fine-tuned 5 models (Pythia-1B, Pythia-1.4B, Pythia-2.8B, Llama-7B, Llama-7B-chat) on 1,000 randomly selected points from The Pile's validation set for 1-4 epochs. In total, these points consisted of 441,176 tokens; at .75 words per token and 500 words/page, this corresponds to ≈ 660 pages of single-spaced text. For all fine-tuning, we used a learning rate of 5×10^{-5} with the AdaFactor optimizer and 4 steps of gradient checkpointing. We then evaluate data extractability in the discoverable ($k = m = 50$) and non-discoverable settings ($k \in \{2, 4, 8\}$ and $m \in \{25, 50\}$). As in other sections, we use the sampling parameters defined in Appendix J.

4.1.1. Discoverable Extraction ($k = 50$).

We begin by running a generate-then-extract pipeline with 20 generations for 500 samples in the discoverable extraction setting. In Table 20, we report the attack *precision* (the proportion of top-ranked generations that match the true suffix), the 01-*Hamming* distance, which is the average proportion of tokens that match between the top-ranked generation and true suffix across samples, and the proportion of suffixes with generation probability greater than .1, .05, .01, and 0.001. We see in the table that after 3 epochs of fine-tuning, more than half the fine-tuning dataset is discoverably extractable for both models, which goes up to about 90% after the 4^{th} Epoch. Even when the top ranked generation is not an exact match, there is substantial data leakage; in Llama-7B-chat, 48%, 81%, and 95%+ of the tokens in the top-ranked generation match the true suffix after 2, 3, and 4 epochs of fine-tuning, respectively. This is a huge fraction of the fine-tuning dataset!

A successful data extraction attack has two components: generating the actual suffix (memorization) and then identifying it correctly. In Table 20, we see that attack precision does not jump until 2-3 epochs of fine-tuning, which corresponds to when most of the suffixes are likely to be generated in 20 samples. On the largest models (Llama-7B), after 2 epochs of fine-tuning as much as 10-30% of the suffixes have probability $> .001$, and over 10% have probability $> .1$, a substantial increase over smaller model sizes. Across all models, we see a sharp rise in memorization during Epoch 2-3, with near perfect memorization by Epoch 4. See Figures 25 and 26 for histograms of the probabilities across all 4 epochs of fine-tuning in the discoverable setting for all models.

After the true suffix is well-memorized and can be generated, a successful extraction attack must correctly identify the true suffix. Inspecting Table 20, we can infer that $\texttt{FLoRa}_{\theta,\theta'}$ works quite well in ranking: with 20 generations per sample,

we can expect that only suffixes with that have probability $p > .05$ are likely to even be generated, and the proportion of top-ranked suffixes that are the true suffix (precision) closely matches the proportion of suffixes that are likely to be generated.

4.1.2. Non-Discoverable Extraction ($k = \{2, 4, 8\}$).

We now weaken the assumption that the adversary has access to a 50-token prefix of the input; instead, we suppose they only have a few tokens of the sample they wish to extract. For instance, suppose a hospital has fine-tuned an LLM on patient records (with the patient name as the title, and description underneath) to simulate patient conversations. Given only the patient names and knowledge of the base model, can an adversary extract their records? The answer, we find, is yes.

When the adversary has fewer tokens to prompt the LLM with, the task of correctly *selecting* the training sample (if it is generated) becomes easier as the generated samples will in general be farther from the training sample. On the other hand, as we decrease the prefix length, the probability of generating the correct suffix decreases. The question of whether discoverable extraction results are preserved with fewer prefixes made available to the adversary then, is a question of how much these suffix generation probabilities decrease. Inspecting Table 21 in the Appendix, we see that even with only $k = 2$ token prefixes, by Epoch 2 there is substantial memorization across all model sizes, when the prefix is $k = 2$ tokens and the suffix is of length 25, 25-45% of samples have suffix probability $p > 0.001$. In the largest models (Llama-7B), by Epoch 2, nearly 30% of samples have suffix probability > 0.1 given just 2 tokens of context, and 13.6% of samples also have $p > 0.1$ of the next 50 tokens given just 2 tokens of context! As such, the number of tokens that the adversary uses to prompt is not the primary cause of eliciting memorized content; instead, it is multiple epochs of fine-tuning. Across all model sizes, given black-box access to both the fine-tuned and base model, along with just a few tokens of context, *a large number of tokens of the fine-tuning dataset can be extracted after just* 2-3 *epochs of fine-tuning*.

4.2. Pretrained Extraction.

We defer the full presentation of pretraining extraction results to Appendix H. Unlike with fine-tuned models, even when prompted with a 50-token prefix $\mathbf{a}$, across all the pretrained models $p_\theta(\mathbf{b}|\mathbf{a})$ remains low, and so the probability of extracting a *specific* example is also low. As such, due to compute limitations, we don't run a generate-then-rank pipeline for extracting pretrained LLM data. Recent work has found, however, that the upper limits of discoverable extraction are relatively high. By prompting with common 5-token prefixes, generating suffixes of length 50, and brute force searching of terabytes of Internet text data, (Nasr et al., 2023) show that $> 1\%$ of generations from LLMs including Pythia and Llama-7B/Llama-7B-chat models are memorized training samples. This suggests that even though memorization levels are low, a mechanism $\mathcal{M}$ identify training examples with high accuracy at low FPRs is an important threat vector. Interestingly, we find that unlike in the fine-tuned setting, the success of our pretrained MIAs *does not* translate into ranking generated samples. Thus, rather than using an MIA to rank, we instead explicitly train supervised white-box attacks (denoted `Gen-LogReg`, `Gen-NN`) for the task of classifying train points versus samples that share the same 50-token prefix but have a model-generated suffix. On Pythia-1.4B, this results in attacks that can identify the correct suffix 90.1% and 88.3% of the time out of 20 generations, whereas naively applying the `LogReg`, `NN` MIAs only succeed 15.2%, 22.1% of the time, respectively. See Appendix H for full results.

5. Discussion

This paper firmly establishes the following: pretrained LLMs are vulnerable to very strong MIAs if an adversary can train a supervised attack, and fine-tuned LLMs are vulnerable to strong MIAs if the adversary has black-box access to the model losses. In the pretrained setting, white-box supervised MIAs can obtain extremely high attack success rates; they can be extended with model stealing techniques to the gray-box setting, where they outperform existing baselines. Supervised MIAs can also be trained to distinguish between training data and generations, greatly improving a crucial component of any pretrained data extraction attack. In the fine-tuned setting, when LLMs are fine-tuned for over an epoch, they start to memorize significant portions of their fine-tuning dataset and are vulnerable to extraction attacks.

Implications for the Practitioner. While our results are of independent scientific interest to the LLM privacy community, there are three main takeaways for LLM practitioners:

1. Strong MIAs are possible for both pretrained and fine-tuned models, with many applications outside of privacy. Currently, pretrained models are only vulnerable to white-box MIAs.

2. Model-stealing blurs the line between white-box and black-box MIAs, and so in highly sensitive domains it may be prudent to assume an adversary may gain white-box access.

3. Great care must be taken to limit the ability of attackers to access models fine-tuned on sensitive data; a significant fraction of the fine-tuning dataset is vulnerable to extraction.

References

Biderman, S., Schoelkopf, H., Anthony, Q., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., Skowron, A., Sutawika, L., and van der Wal, O. Pythia: A suite for analyzing large language models across training and scaling, 2023.

Carlini, N., Tramèr, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T. B., Song, D. X., Erlingsson, Ú., Oprea, A., and Raffel, C. Extracting training data from large language models. In *USENIX Security Symposium*, 2020. URL https://api.semanticscholar.org/CorpusID:229156229.

Carlini, N., Chien, S., Nasr, M., Song, S., Terzis, A., and Tramer, F. Membership inference attacks from first principles, 2022.

Carlini, N., Cooper, H., Tramer, F., and Zhang, C. Training data extraction challenge, 2023a. URL https://github.com/google-research/lm-extraction-benchmark.

Carlini, N., Ippolito, D., Jagielski, M., Lee, K., Tramer, F., and Zhang, C. Quantifying memorization across neural language models, 2023b.

Carlini, N., Paleka, D., Dvijotham, K. D., Steinke, T., Hayase, J., Cooper, A. F., Lee, K., Jagielski, M., Nasr, M., Conmy, A., Wallace, E., Rolnick, D., and Tramèr, F. Stealing part of a production language model, 2024.

Duan, M., Suri, A., Mireshghallah, N., Min, S., Shi, W., Zettlemoyer, L., Tsvetkov, Y., Choi, Y., Evans, D., and Hajishirzi, H. Do membership inference attacks work on large language models?, 2024.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S., and Leahy, C. The pile: An 800gb dataset of diverse text for language modeling, 2020.

Hayes, J., Shumailov, I., Triantafillou, E., Khalifa, A., and Papernot, N. Inexact unlearning needs more careful evaluations to avoid a false sense of privacy, 2024.

Ishihara, S. Training data extraction from pre-trained language models: A survey, 2023.

Jagielski, M., Wu, S., Oprea, A., Ullman, J., and Geambasu, R. How to combine membership-inference attacks on multiple updated models, 2022.

Kang, C. and Metz, C. F.t.c. opens investigation into chatgpt maker over technology's potential harms, Jul 2023. URL https://www.nytimes.com/2023/07/13/technology/chatgpt-investigation-ftc-openai.html.

Kurmanji, M., Triantafillou, P., Hayes, J., and Triantafillou, E. Towards unbounded machine unlearning, 2023.

Li, M., Wang, J., Wang, J., and Neel, S. MoPe: Model perturbation based privacy attacks on language models. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13647–13660, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.842. URL https://aclanthology.org/2023.emnlp-main.842.

Liu, B., Ding, M., Shaham, S., Rahayu, W., Farokhi, F., and Lin, Z. When machine learning meets privacy: A survey and outlook. *ACM Comput. Surv.*, 54(2), mar 2021. ISSN 0360-0300. doi: 10.1145/3436755. URL https://doi.org/10.1145/3436755.

Mattern, J., Mireshghallah, F., Jin, Z., Schoelkopf, B., Sachan, M., and Berg-Kirkpatrick, T. Membership inference attacks against language models via neighbourhood comparison. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 11330–11343, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.719. URL https://aclanthology.org/2023.findings-acl.719.

Nasr, M., Shokri, R., and Houmansadr, A. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, May 2019. doi: 10.1109/sp.2019.00065. URL http://dx.doi.org/10.1109/SP.2019.00065.

Nasr, M., Carlini, N., Hayase, J., Jagielski, M., Cooper, A. F., Ippolito, D., Choquette-Choo, C. A., Wallace, E., Tramèr, F., and Lee, K. Scalable extraction of training data from (production) language models, 2023.

Pawelczyk, M., Neel, S., and Lakkaraju, H. In-context unlearning: Language models as few shot unlearners, 2023.

Shi, W., Ajith, A., Xia, M., Huang, Y., Liu, D., Blevins, T., Chen, D., and Zettlemoyer, L. Detecting pretraining data from large language models, 2023.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. Membership inference attacks against machine learning models, 2017.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models, 2023.

Ye, J., Maddi, A., Murakonda, S. K., Bindschaedler, V., and Shokri, R. Enhanced membership inference attacks against machine learning models, 2022.

Yeom, S., Giacomelli, I., Fredrikson, M., and Jha, S. Privacy risk in machine learning: Analyzing the connection to overfitting, 2018.

Yu, W., Pang, T., Liu, Q., Du, C., Kang, B., Huang, Y., Lin, M., and Yan, S. Bag of tricks for training data extraction from language models, 2023.

Zhang, J., Sun, J., Yeats, E., Ouyang, Y., Kuo, M., Zhang, J., Yang, H., and Li, H. Min-k%++: Improved baseline for detecting pre-training data from large language models. *arXiv preprint arXiv:2404.02936*, 2024.

Zhou, Z., Xiang, J., Chen, C., and Su, S. Quantifying and analyzing entity-level memorization in large language models, 2023.

A. Impact Statement

This paper presents work that illustrates the vulnerabilities of LLMs to membership inference and data extraction, which can compromise the privacy of internal data that the LLMs are trained on. There are many potential downstream consequences of our work, most immediately in adversarial attacks against fine-tuned LLMs. Nevertheless, we hope that our research will encourage companies with LLMs trained on internal data to be more cautious with their deployment—and spur further development into privacy risk mitigation with generative models. Furthermore, all attacks in this paper were done on open source models with public datasets, so there is limited risk of confidential data exposure from our direct results.

B. Compute

To run the experiments, we used a compute node with an NVIDIA A100 80GB GPU. All experiments in this paper can be run on a single one of these GPUs.

All results for pretrained MIAs are on model sizes 70M, 160M, 410M, 1B, 2.8B, and 6.9B. As noted previously, in the pretrained setting, we evaluate all MIAs on 2,000 points from each of The Pile's train and validation sets. For both baselines $\texttt{LOSS}_\theta$ and $\texttt{Min-K}_\theta$, this requires running inference 4,000 times across all model sizes. $\texttt{GradNorm}_p$ requires 4,000 forward and backward passes (to compute the gradients), across all model sizes. For $\texttt{LogReg}_\theta$ and $\texttt{NN}_\theta$, we first compute gradients for 10,000 points in each of the Pile's training and validation sets (these are the points we will train our classifier on). Next, we compute norms across layers and we trained our supervised classifier on this labeled data. Finally, we compute gradients and norms for a separate 2,000 points in each of The Pile's train and validation sets to evaluate our MIA. In total, these attacks took ≈ 3 A100 gpu-days.

For $\texttt{FLoRa}_{\theta,\theta'}$, we fine-tune all model sizes up to 2.8B, as well as Llama-7B and Llama-7B-chat, for one epoch, and then run inference 4,000 times for evaluation (2,000 train/val). For $\texttt{zlib\_LoRa}_\theta$ and thresholding on fine-tuned loss, we run inference 4,000 times for all model sizes up to 2.8B. In total, this took 1 gpu-day.

For fine-tune extraction, we fine-tuned each of Pythia-1B, Pythia-1.4B, Pythia-2.8B, Llama-7B, and Llama-7B-chat on 1,000 points from The Pile's validation set for 4 epochs. This represents at least 20,000 backward passes. Next, in the discoverable extraction setting: for each of the 5 model size and 4 fine-tune epochs, we took five hundred 50-token prefixes, ran 20 generations on each, and then ranked them by computing loss ratios (which requires running two additional forward inferences, on the fine-tuned and base model). In total, this represents at least 600,000 forward passes. In the non-discoverable setting, because of compute limitations, we did not actually run generations but instead queried probabilities for $(k, x) \in \{2, 4, 8\} \times \{25, 50\}$. For each of the 6 k-x tuples, across all 5 model sizes, across the 4 epochs of fine-tuning, we computed suffix probabilities for 500 samples. In total, this represents 60,000 forward inferences. In total, these extraction results took ≈ 7 gpu-days.

To train a classifier to identify the true suffix from model generations in the pretrained setting, we first sample 10,000 points from each of The Pile's train and validation splits. Next, we compute gradients norms for the 20,000 points. To evaluate the MIA, we take a disjoint set of 2,000 points from the train set, run 20 suffix generations on each, and then compute the gradient norms for those 40,000 points. We then train the supervised classifier on the points and evaluate its ability to identify the true suffix on each of the 2,000 points.

C. Baseline Pretrain MIA Results

Several MIAs have been proposed for the pretraining setting of LLMs, although their performance is relatively weak. The simplest of these is loss thresholding, also denoted as $\texttt{LOSS}_\theta$; other recent attacks include $\texttt{MoPe}_\theta$ and $\texttt{Min-K}_\theta$ (Li et al., 2023; Shi et al., 2023). $\texttt{MoPe}_\theta$ convolves the model parameters with noise and uses the decrease in log-likelihood for a queried point as a MIA statistic, which they show approximates the trace of the Hessian matrix. When evaluating on 1,000 training and 1,000 validation samples across model sizes in the Pythia Suite, we find that $\texttt{LOSS}_\theta$ achieves AUCs from 0.5-0.52 and $\texttt{MoPe}_\theta$ achieves peaks AUCs of 0.64 and 0.65 for the model sizes of 160M and 410M, with AUCs that outperform $\texttt{LOSS}_\theta$ across model sizes.

(Shi et al., 2023) proposes $\texttt{Min-K}_\theta$, which thresholds on the average loss of the $k\%$ of tokens with smallest loss in a candidate input. We implement their attack and present the best results for each model size searching over $k = 0.5\%, 1\%, 5\%, 10\%$ (see Table 1). We find that $\texttt{Min-K}_\theta$ performs comparably or worse than $\texttt{LOSS}_\theta$ on the model sizes we tested, with < 0.52 AUC on model sizes 160M to 2.8B. Notably, our evaluation method using a train/validation split from The Pile is different from theirs (WikiMIA, a dataset of Wikipedia articles written before/after Pythia was trained). We suspect this is because the method's intuition, that non-training data will contain some outlier tokens with low probability compared to training data, holds for the temporally different data in WikiMIA but may not hold for training and non-training data drawn from the exact time period. At low FPR rates (e.g. $0.01, 0.1, 0.25$), $\texttt{Min-K}_\theta$ and $\texttt{LOSS}_\theta$ markedly underperform $\texttt{MoPe}_\theta$, performing worse than random.

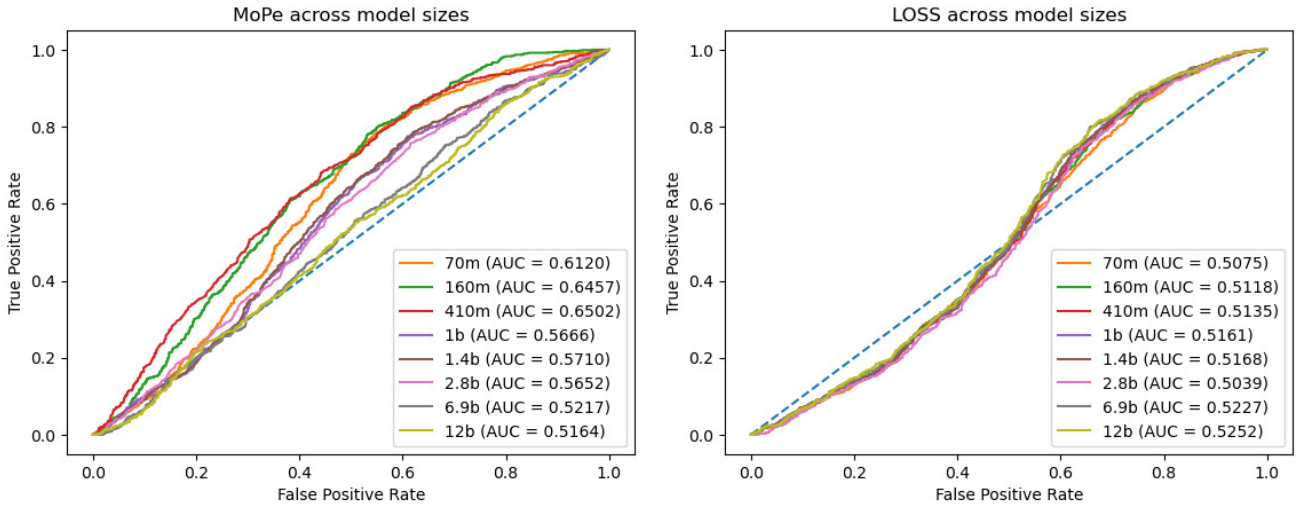


Figure 6. $\texttt{MoPe}_\theta$ and $\texttt{LOSS}_\theta$ ROC curves from (Li et al., 2023)

Table 1. For Pythia-1.4B, we report the AUC and TPR for the $\texttt{Min-K}_\theta$ attack at various levels of K.

K	AUC	$\text{TPR}_{0.5\%}$	$\text{TPR}_{1\%}$	$\text{TPR}_{5\%}$
0.005	0.531 ± 0.0180	0.00 ± 0.000694	0.00 ± 0.00156	0.00 ± 0.0102
0.01	0.536 ± 0.0179	0.00 ± 0.000753	0.00 ± 0.00169	0.0345 ± 0.0118
0.05	0.539 ± 0.0181	0.00250 ± 0.00281	0.00600 ± 0.00492	0.0370 ± 0.0155
0.1	0.549 ± 0.0169	0.00200 ± 0.00281	0.00750 ± 0.00615	0.0425 ± 0.0147
0.2	0.549 ± 0.0176	0.00150 ± 0.00264	0.00750 ± 0.00623	0.0450 ± 0.0108
0.4	0.540 ± 0.0197	0.00250 ± 0.00280	0.00450 ± 0.00519	0.0420 ± 0.0127
0.8	0.537 ± 0.0180	0.00300 ± 0.00289	0.00600 ± 0.00396	0.0385 ± 0.0129

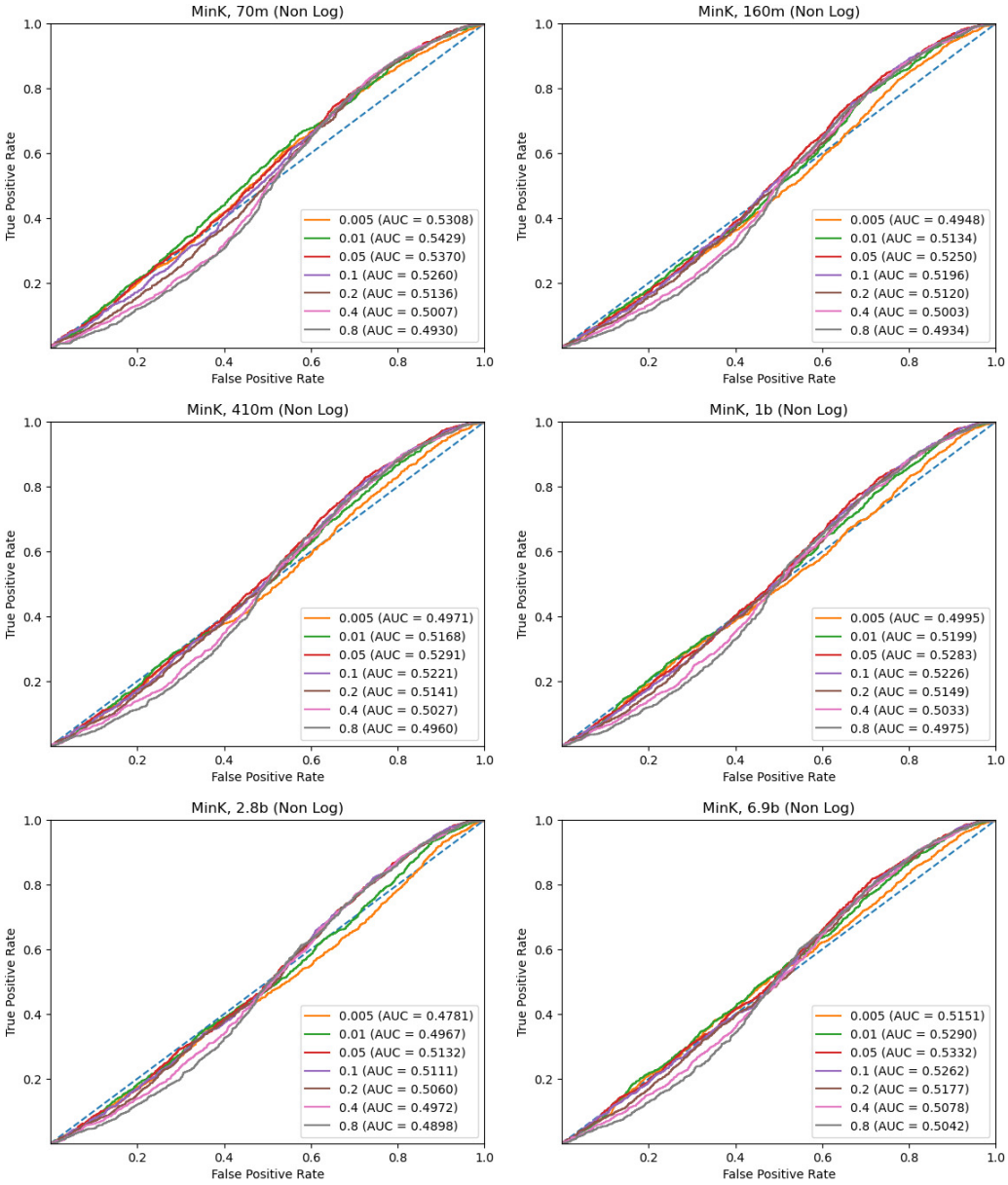


Figure 7. Min-K$_\theta$ across Pythia model sizes.

D. Gradient Norm Thresholding MIAs

Our gradient norm-based attacks follow from similar intuition as $\texttt{LOSS}_\theta$ and $\texttt{MoPe}_\theta$: θ is trained to approximately minimize the casual language modeling loss on the training dataset $\mathcal{X}_{\texttt{TRAIN}} \sim \mathcal{D}$ sampled from the true unknown data distribution $\mathcal{D} \in \Delta(\mathcal{V}^T)$: $\mathbb{E}_{x \sim \mathcal{X}_{\texttt{TRAIN}}}[\ell(\theta, x)]$. So if θ is an approximate local minimizer of the training loss, $||\mathbb{E}_{x \sim \mathcal{X}_{\texttt{TRAIN}}} \nabla_\theta \ell(\theta, x)|| \approx 0$. This is only a loose motivation; our actual attack of course does not have access to $\mathcal{X}_{\texttt{TRAIN}}$, and so given a candidate point x, we predict $x \in \mathcal{X}_{\texttt{TRAIN}}$ if the point-wise gradient norm $||\nabla_\theta \ell(\theta, x)||_p < \tau$ for some p norm. Further weakening the motivation, we find empirically that taking the gradient with respect to the input embedding $\nabla_x \ell(\theta, x)$ also yields effective MIAs. We include full MIA results on the Pythia models for $\texttt{GradNorm}_p$ with respect to θ and x, and with $p \in \{1, 2, \infty\}$ in Figure 8. We see that the best performing $\texttt{GradNorm}_p$ method, by far, is where the gradient is taken with respect to x and $p = \infty$. While $\texttt{GradNorm}_p$ attacks do not perform superbly alone, they form the basis of our $\texttt{LogReg}_\theta$ and $\texttt{NN}_\theta$ methods, which use layerwise gradient norms as an essential ingredient of training the attack.

Table 2. $\texttt{GradNorm}_\infty$ attack performance using ℓ_∞ norm on gradient with respect to input embedding, with 95% confidence interval over 1000 bootstraps.

MIA	AUC	$\textbf{TPR}_{0.5\%}$	$\textbf{TPR}_{1\%}$	$\textbf{TPR}_{5\%}$
70M	0.591 ± 0.0121	0.00 ± 0.000317	0.00160 ± 0.00393	0.0636 ± 0.0148
160M	0.571 ± 0.0122	0.00400 ± 0.00276	0.00850 ± 0.00448	0.0480 ± 0.0104
410M	0.534 ± 0.00552	0.00435 ± 0.000942	0.00830 ± 0.00146	0.0492 ± 0.00515
1B	0.654 ± 0.0102	0.00250 ± 0.00186	0.00400 ± 0.00275	0.0495 ± 0.0105
1.4B	0.614 ± 0.0165	0.00 ± 0.000407	0.00100 ± 0.00326	0.0495 ± 0.0149
2.8B	0.480 ± 0.0187	0.00 ± 0.00	0.000500 ± 0.00206	0.0395 ± 0.0139
6.9B	0.636 ± 0.0164	0.00 ± 0.00	0.00 ± 0.000964	0.0550 ± 0.0153

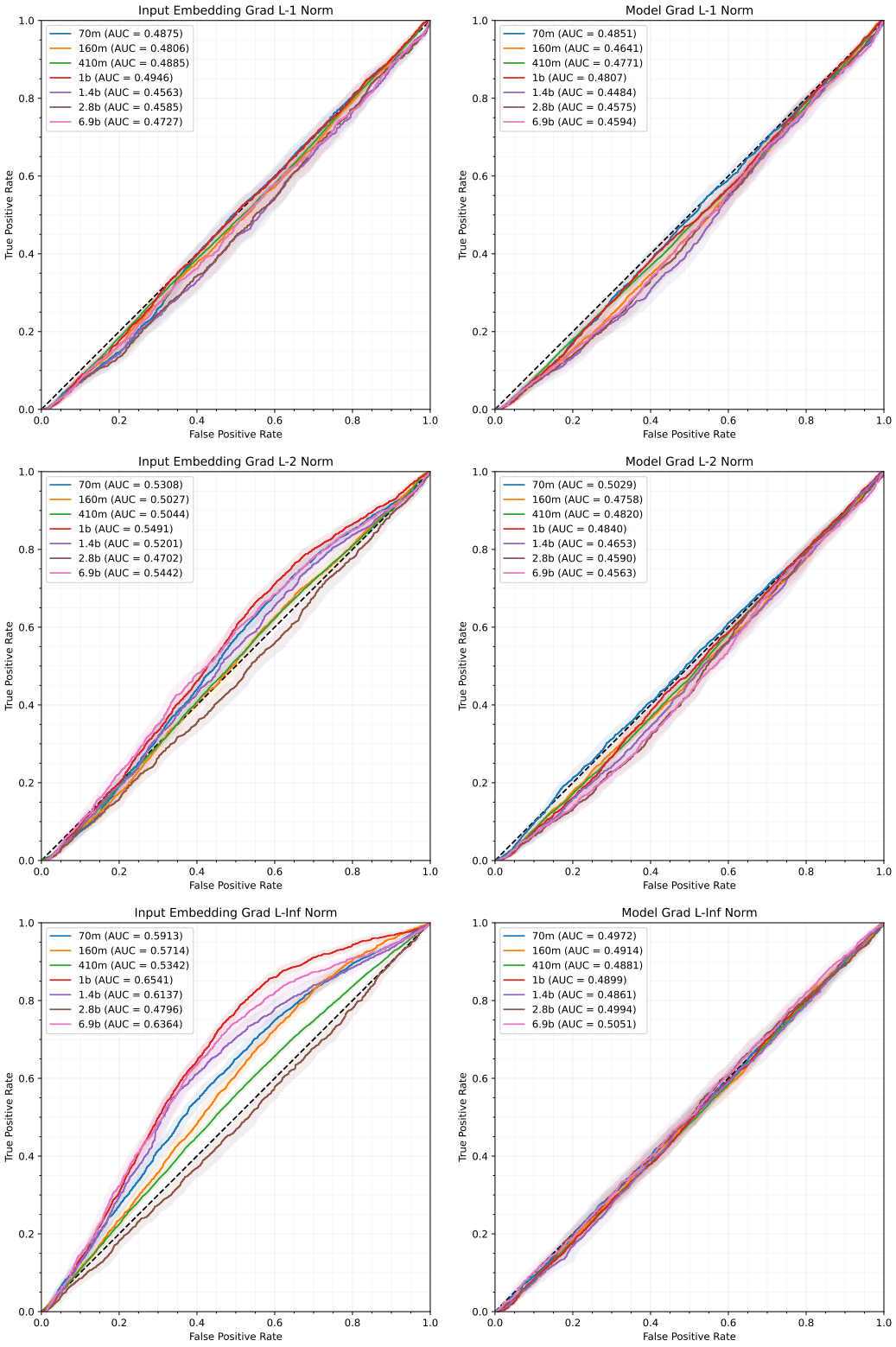


Figure 8. $\texttt{GradNorm}_p$ thresholding MIAs. ℓ_∞ of input embedding gradients is the only norm order and gradient type that works.

E. Pretrained Supervised MIAs Details and Full Results

For all model sizes, we first train a simple logistic regression to classify the training data. We also train a simple MLP with the default PyTorch Adam parameters, 10 epochs of training, and a batch size of 64. We used a neural network with hidden layers of size $[250, 100, 10]$. We evaluate all pretrained MIAs on the Pythia-deduped suite of models on sizes ranging from 70M to 6.9B (using the 1 epoch checkpoint step 98,000), evaluating the attacks using 2,000 randomly sampled training examples and 2,000 validation samples from the Pile dataset that the Pythia series was pretrained on. In this section, we outline all results, with CIs, for pretrain supervised MIAs.

Table 3. We train a **logistic regression** on all norm features ($\ell_1, \ell_2, \ell_\infty$ norms of gradient with respect to input embedding, each layer, and the entire model). The train set size of the classifier is $20,000$. We record AUC and TPR values for all model sizes on $2,000$ train Pile, $2,000$ validation Pile samples, with 95% confidence intervals bootstrapped over $1,000$ trials.

Model Size	AUC	$\text{TPR}_{0.5\%}$	$\text{TPR}_{1\%}$	$\text{TPR}_{5\%}$
70M	0.682 ± 0.0142	0.0100 ± 0.00667	0.0180 ± 0.00954	0.148 ± 0.0410
160M	0.679 ± 0.0177	0.0175 ± 0.0138	0.0625 ± 0.0314	0.161 ± 0.0310
410M	0.681 ± 0.0145	0.0145 ± 0.00922	0.0265 ± 0.0143	0.139 ± 0.0224
1B	0.786 ± 0.0128	0.0545 ± 0.0387	0.126 ± 0.0603	0.335 ± 0.0385
1.4B	0.787 ± 0.0132	0.0160 ± 0.0343	0.105 ± 0.0602	0.344 ± 0.0421
2.8B	0.562 ± 0.0182	0.0100 ± 0.00790	0.0240 ± 0.0149	0.0885 ± 0.0195
6.9B	0.783 ± 0.0136	0.0625 ± 0.0654	0.161 ± 0.0449	0.316 ± 0.0319

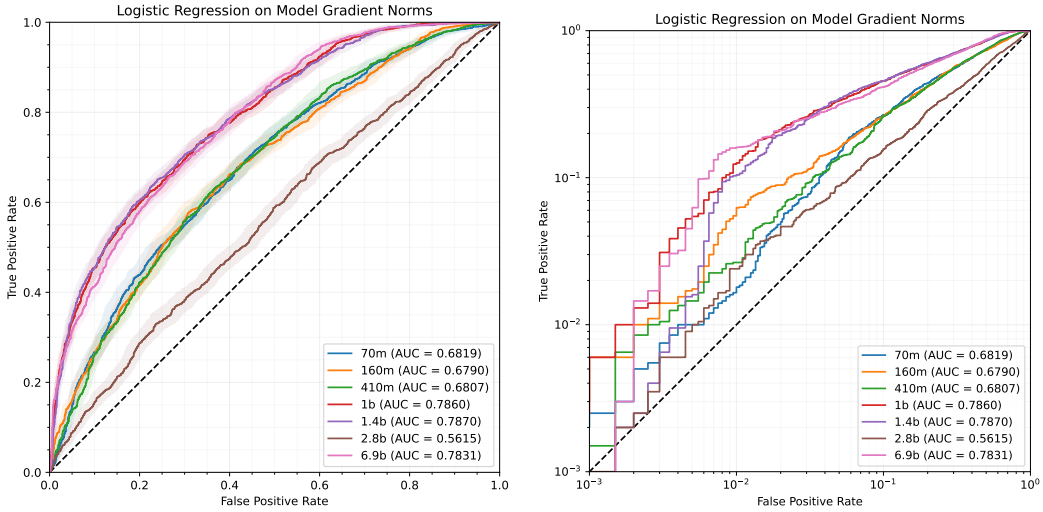


Figure 9. Logistic Regression on All Gradient Norms

Table 4. We train a **neural network** on all norm features ($\ell_1, \ell_2, \ell_\infty$ norms of gradient with respect to input embedding, each layer, and the entire model). The train set size of the classifier is 20000. We record AUC and TPR values for all model sizes on 2000 train Pile, 2000 validation Pile samples, with 95% confidence intervals bootstrapped over 1000 trials.

Model Size	AUC	$\text{TPR}_{0.5\%}$	$\text{TPR}_{1\%}$	$\text{TPR}_{5\%}$
70M	0.660 ± 0.0166	0.0705 ± 0.0387	0.122 ± 0.0311	0.219 ± 0.0269
160M	0.632 ± 0.0167	0.0470 ± 0.0150	0.0670 ± 0.0196	0.159 ± 0.0225
410M	0.624 ± 0.0169	0.0455 ± 0.0260	0.0660 ± 0.0201	0.132 ± 0.0164
1B	0.777 ± 0.0147	0.220 ± 0.0446	0.243 ± 0.0293	0.325 ± 0.0244
1.4B	0.738 ± 0.0150	0.161 ± 0.0391	0.183 ± 0.0231	0.311 ± 0.0260
2.8B	0.537 ± 0.0166	0.00750 ± 0.00501	0.0145 ± 0.00709	0.0675 ± 0.0177
6.9B	0.738 ± 0.0165	0.144 ± 0.0260	0.175 ± 0.0260	0.289 ± 0.0325

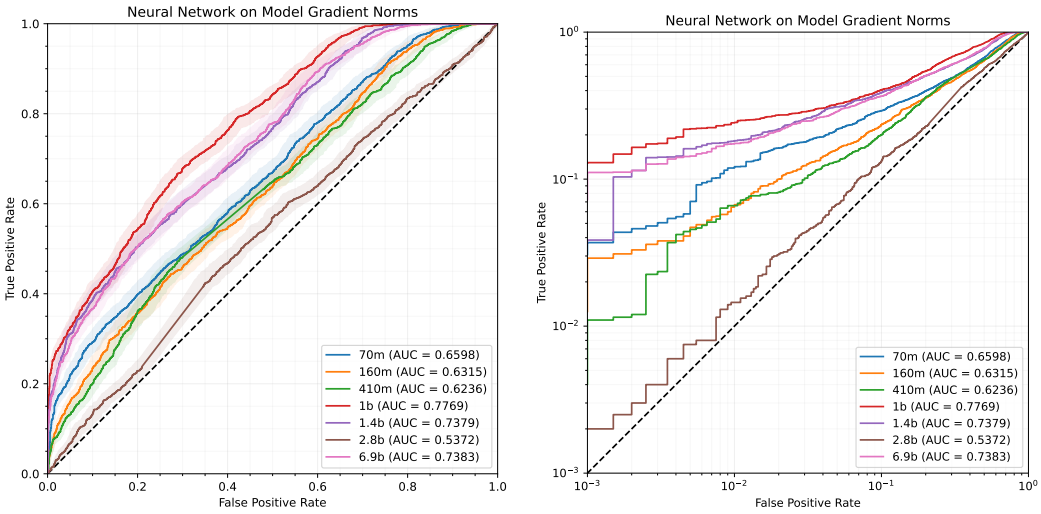


Figure 10. Neural Network on All Gradient Norms

F. Comparison of Pretrained MIAs

Table 5. We record AUC and TPR values on the Pythia-1B model for a variety of pretrained MIAs with 95% confidence intervals over 1,000 bootstraps, evaluated on 2,000 samples from the train Pile and 2,000 samples from the validation Pile. $\texttt{Min-K}_\theta$ below refers to $k = 0.05$.

MIA	AUC	$\text{TPR}_{0.5\%}$	$\text{TPR}_{1\%}$	$\text{TPR}_{5\%}$
$\texttt{LOSS}_\theta$	0.534 ± 0.0189	0.00250 ± 0.00276	0.00550 ± 0.00469	0.0415 ± 0.0148
$\texttt{Min-K}_\theta$	0.539 ± 0.0175	0.00250 ± 0.00295	0.00600 ± 0.00524	0.0370 ± 0.0142
$\texttt{MoPe}_\theta$	0.557 ± 0.0170	0.000500 ± 0.00133	0.00250 ± 0.00255	0.0410 ± 0.0148
$\texttt{GradNorm}_\infty$	0.667 ± 0.0145	0.00 ± 0.00	0.000500 ± 0.00193	0.0495 ± 0.0205
ModelStealing	0.666 ± 0.0172	0.0225 ± 0.00963	0.0325 ± 0.0145	0.109 ± 0.0214
$\texttt{LogReg}_\theta$	$\mathbf{0.786 \pm 0.0119}$	0.0545 ± 0.0379	0.126 ± 0.0596	$\mathbf{0.335 \pm 0.0370}$
$\texttt{NN}_\theta$	0.777 ± 0.0126	$\mathbf{0.220 \pm 0.0457}$	$\mathbf{0.243 \pm 0.0301}$	0.325 ± 0.0255

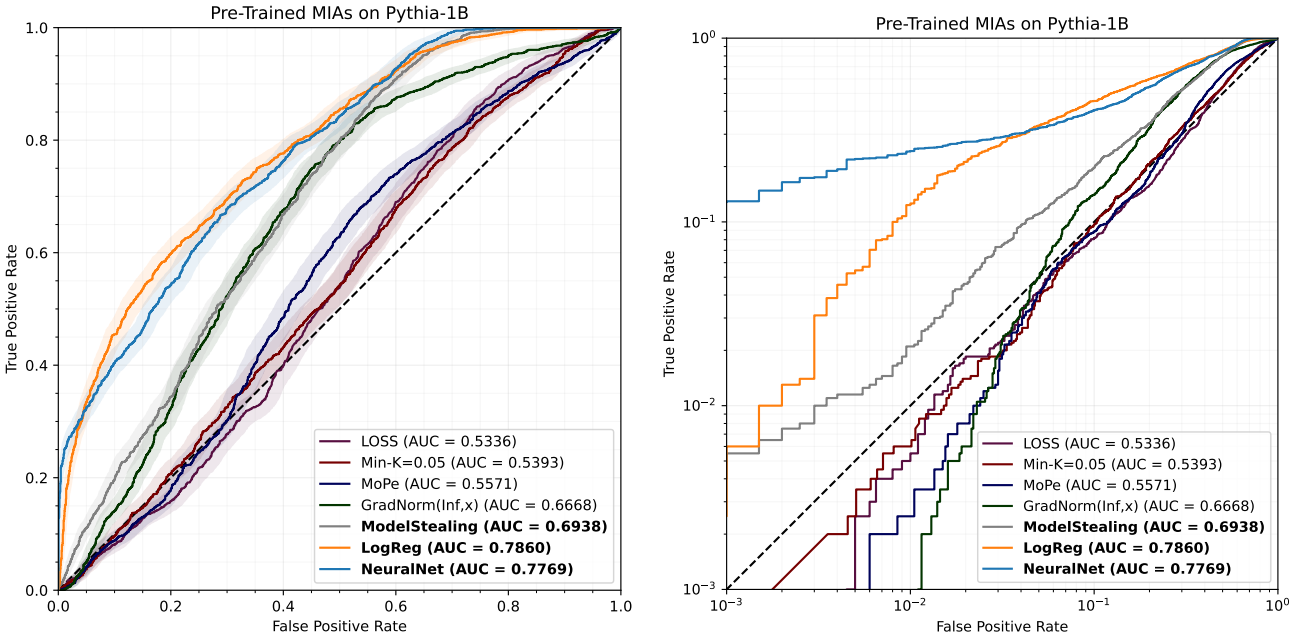


Figure 11. Pretrained MIAs on Pythia-1B

G. Pretrained Supervised MIAs Ablations

We consider two kinds of gradient information as our white-box features: gradient with respect to the input embedding and gradient with respect to each layer's parameters. To reduce the dimensionality of the gradient, we take the norm of each gradient with the ℓ_1, ℓ_2, and ℓ_∞ norms. For example, for a prompt p and layer W_t, we compute $\nabla_{W_t}\ell(\theta(p))$ and use $\|\nabla_{W_t}\ell(\theta(p))\|_1, \|\nabla_{W_t}\ell(\theta(p))\|_2, \|\nabla_{W_t}\ell(\theta(p))\|_\infty$ as features. Thus, we construct $3(1+L)$ features where L is the number of layers in the language model.

G.1. Number of Classifier Train Points Ablation

We repeat our $\texttt{LogReg}_\theta$ and $\texttt{NN}_\theta$ experiments by varying the number of training data samples between 100 and $10{,}000$ (see Section G.1). We show that while performance tends to increase by the logarithm of the number of samples used to train the classifier ($\texttt{LogReg}_\theta$ or $\texttt{NN}_\theta$), around $1{,}000$ points is already sufficient to get high AUCs and TPRs at low FPRs. For the Pythia-deduped model series at 1 epoch which has been trained on 207B tokens, $1{,}000$ samples of $2{,}048$ tokens is just $1/100{,}000$ of the entire training set. While training with more samples has vastly diminishing returns, it does greatly improve the TPR at low FPRs for $\texttt{NN}_\theta$.

Table 6. We train a **logistic regression** on all norm features over different subsamples of the original training dataset to view the effect of dataset size on classifier quality. We record AUC and TPR values for the Pythia-1B model.

Train Data Size	**AUC**	**TPR**$_{0.5\%}$	**TPR**$_{1\%}$	**TPR**$_{5\%}$
100	0.671 ± 0.0673	0.0600 ± 0.0846	0.0800 ± 0.0845	0.220 ± 0.124
200	0.665 ± 0.0566	0.0150 ± 0.0164	0.0200 ± 0.0198	0.110 ± 0.0787
300	0.656 ± 0.0452	0.0133 ± 0.0273	0.0267 ± 0.0302	0.107 ± 0.0589
400	0.697 ± 0.0404	0.0125 ± 0.0200	0.0325 ± 0.0324	0.0825 ± 0.0580
500	0.728 ± 0.0292	0.0280 ± 0.0200	0.0320 ± 0.0220	0.134 ± 0.0480
1000	0.742 ± 0.0216	0.0570 ± 0.0173	0.0810 ± 0.0331	0.216 ± 0.0413
1500	0.759 ± 0.0164	0.0407 ± 0.0381	0.107 ± 0.0542	0.245 ± 0.0342
2000	0.757 ± 0.0137	0.0365 ± 0.0166	0.0685 ± 0.0296	0.274 ± 0.0326
2500	0.763 ± 0.0154	0.0330 ± 0.0239	0.0810 ± 0.0461	0.258 ± 0.0379
5000	0.770 ± 0.0158	0.0220 ± 0.0257	0.0970 ± 0.0599	0.281 ± 0.0445
7500	0.777 ± 0.0133	0.0470 ± 0.0396	0.107 ± 0.0429	0.289 ± 0.0523
10000	0.786 ± 0.0158	0.0545 ± 0.0458	0.126 ± 0.0620	0.335 ± 0.0387
12500	0.780 ± 0.0133	0.0505 ± 0.0490	0.141 ± 0.0441	0.320 ± 0.0329
15000	0.776 ± 0.0146	0.0770 ± 0.0455	0.157 ± 0.0575	0.340 ± 0.0406

Table 7. We train a **neural network** on all norm features over different subsamples of the original training dataset to view the effect of dataset size on classifier quality. We record AUC and TPR values for the Pythia-1B model.

Train Data Size	**AUC**	**TPR**$_{0.5\%}$	**TPR**$_{1\%}$	**TPR**$_{5\%}$
100	0.529 ± 0.0876	0.00 ± 0.0334	0.0200 ± 0.0339	0.0700 ± 0.0651
200	0.550 ± 0.0573	0.0100 ± 0.0168	0.0150 ± 0.0209	0.0500 ± 0.0445
300	0.592 ± 0.0503	0.00667 ± 0.0157	0.00667 ± 0.0236	0.0500 ± 0.0330
400	0.659 ± 0.0421	0.0100 ± 0.0105	0.0150 ± 0.0163	0.0750 ± 0.0534
500	0.665 ± 0.0314	0.00600 ± 0.0108	0.0160 ± 0.0129	0.0680 ± 0.0357
1000	0.734 ± 0.0217	0.0580 ± 0.0347	0.0870 ± 0.0288	0.225 ± 0.0366
1500	0.724 ± 0.0182	0.0740 ± 0.0186	0.0933 ± 0.0209	0.227 ± 0.0350
2000	0.724 ± 0.0145	0.0805 ± 0.0237	0.0965 ± 0.0205	0.232 ± 0.0243
2500	0.743 ± 0.0139	0.121 ± 0.0276	0.145 ± 0.0271	0.265 ± 0.0355
5000	0.736 ± 0.0169	0.124 ± 0.0289	0.155 ± 0.0271	0.284 ± 0.0384
7500	0.752 ± 0.0139	0.132 ± 0.0254	0.138 ± 0.0194	0.260 ± 0.0288
10000	0.777 ± 0.0142	0.220 ± 0.0509	0.243 ± 0.0288	0.325 ± 0.0246
12500	0.765 ± 0.0120	0.198 ± 0.0418	0.230 ± 0.0318	0.326 ± 0.0244
15000	0.763 ± 0.0149	0.195 ± 0.0337	0.226 ± 0.0221	0.314 ± 0.0250

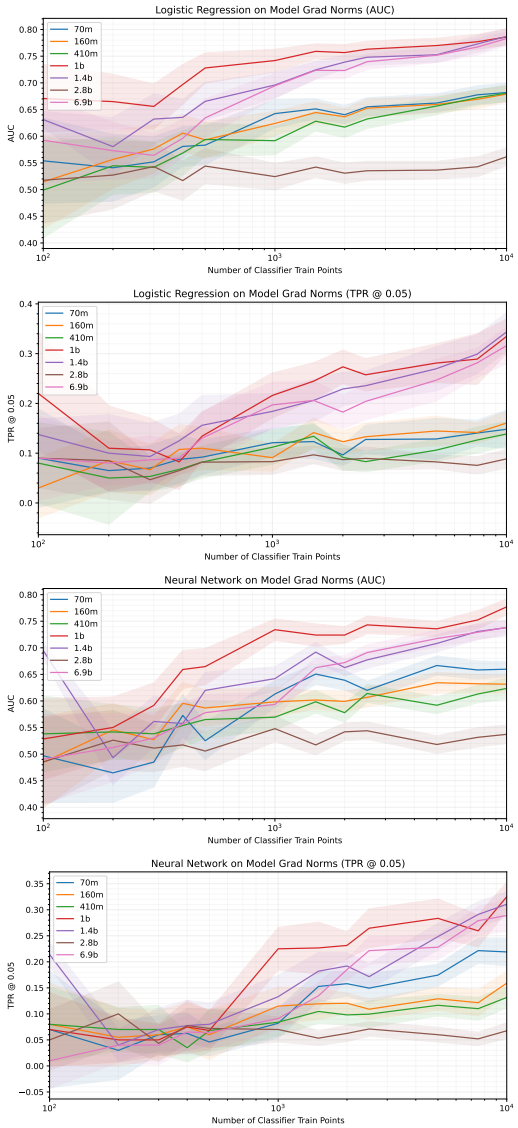


Figure 12. Training Data Size Ablation

G.2. Feature Ablation

We observe that the predictive power of our white-box MIA comes more from each layer's gradient information. This is perhaps unsurprising, given that there are many more features that compose the layerwise norms than the embedding layer norms. However, the combination of the two types of gradients still provides better performance.

G.2.1. Norm of the Input Embedding Gradient

In this section, we consider the norms of the input embedding gradient as a feature ablation. If $\{\theta_t\}_{t=1}^{L}$ are the weights at L layers and $\phi : \mathcal{V}^* \to \mathbb{R}^h$ maps prompts to the input embeddings, then we only use the features $\{\|\nabla_{\phi(p)}\ell(\theta, p)\|_\ell\}_{\ell\in\{1,2,\infty\}}$. This is a subset of the features we use for our supervised MIAs.

Table 8. We train a **logistic regression** on just the **norms of the input embedding gradient**. We record AUC and TPR values for each Pythia model size.

Model Size	AUC	$\text{TPR}_{0.5\%}$	$\text{TPR}_{1\%}$	$\text{TPR}_{5\%}$
70M	0.630 ± 0.0176	0.00750 ± 0.00531	0.0135 ± 0.00737	0.0680 ± 0.0220
160M	0.617 ± 0.0183	0.0170 ± 0.00817	0.0260 ± 0.0143	0.0950 ± 0.0183
410M	0.568 ± 0.0178	0.0110 ± 0.00490	0.0165 ± 0.00629	0.0765 ± 0.0204
1B	0.680 ± 0.0144	0.00800 ± 0.00513	0.0140 ± 0.00801	0.0880 ± 0.0191
1.4B	0.633 ± 0.0166	0.00350 ± 0.00340	0.00700 ± 0.00540	0.0700 ± 0.0212
2.8B	0.449 ± 0.0183	0.00750 ± 0.00446	0.0100 ± 0.00504	0.0300 ± 0.00817
6.9B	0.671 ± 0.0147	0.0115 ± 0.00517	0.0215 ± 0.0101	0.0815 ± 0.0178

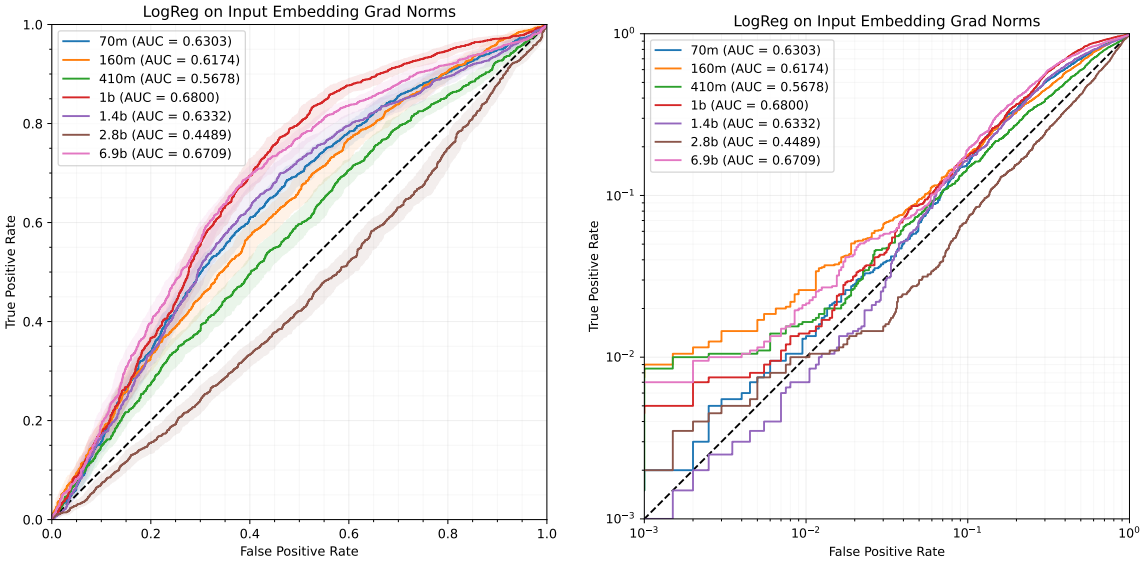


Figure 13. Logistic Regression on Input Embedding Gradient Norm

Table 9. We train a **neural network** on just the **norms of the input embedding gradient**. We record AUC and TPR values for each Pythia model size.

Model Size	AUC	$\text{TPR}_{0.5\%}$	$\text{TPR}_{1\%}$	$\text{TPR}_{5\%}$
70M	0.641 ± 0.0167	0.0125 ± 0.0105	0.0290 ± 0.0120	0.107 ± 0.0215
160M	0.611 ± 0.0190	0.0110 ± 0.00727	0.0165 ± 0.00871	0.0950 ± 0.0270
410M	0.570 ± 0.0158	0.00 ± 0.000780	0.00 ± 0.00176	0.00 ± 0.00956
1B	0.722 ± 0.0143	0.0420 ± 0.0237	0.0595 ± 0.0208	0.178 ± 0.0244
1.4B	0.675 ± 0.0170	0.00900 ± 0.00622	0.0370 ± 0.0234	0.137 ± 0.0262
2.8B	0.498 ± 0.0198	0.00500 ± 0.00507	0.0165 ± 0.00978	0.0480 ± 0.0127
6.9B	0.688 ± 0.0161	0.0170 ± 0.00624	0.0250 ± 0.0135	0.121 ± 0.0290

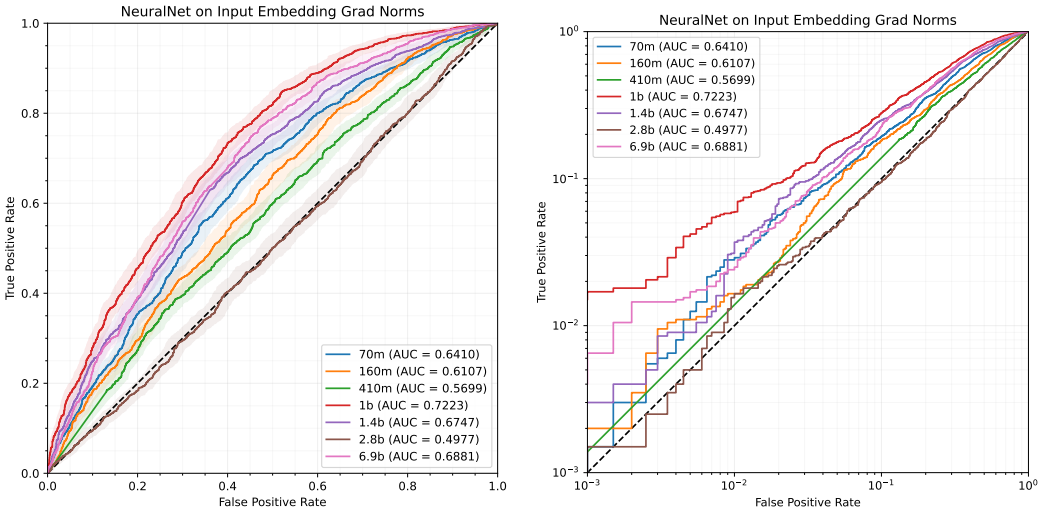


Figure 14. Neural Network on Input Embedding Gradient Norm

G.2.2. Norm of Each Layer's Gradient

In this section, we consider the norms of each layer's gradients as a feature ablation. If $\{\theta_t\}_{t=1}^{L}$ are the weights at L layers, then we use the features $\{\{\|\nabla_{\theta_t}\ell(\theta, p)\|_\ell)\}_{\ell\in\{1,2,\infty\}}\}_{t=1}^{L}$. This is a subset of the features we use for our supervised MIAs.

Table 10. We train a **logistic regression** on the **norms of each layer's gradient**. We record AUC and TPR values for each Pythia model size.

Model Size	AUC	$\text{TPR}_{0.5\%}$	$\text{TPR}_{1\%}$	$\text{TPR}_{5\%}$
70M	0.649 ± 0.0163	0.00800 ± 0.00560	0.0190 ± 0.00993	0.113 ± 0.0258
160M	0.665 ± 0.0168	0.0145 ± 0.0128	0.0400 ± 0.0271	0.158 ± 0.0284
410M	0.673 ± 0.0149	0.0150 ± 0.0108	0.0255 ± 0.0115	0.134 ± 0.0271
1B	0.763 ± 0.0148	0.0535 ± 0.0307	0.154 ± 0.0914	0.305 ± 0.0288
1.4B	0.764 ± 0.0144	0.0350 ± 0.0425	0.114 ± 0.0488	0.294 ± 0.0450
2.8B	0.549 ± 0.0183	0.0165 ± 0.00838	0.0235 ± 0.0101	0.0895 ± 0.0185
6.9B	0.778 ± 0.0131	0.0530 ± 0.0460	0.107 ± 0.0532	0.300 ± 0.0258

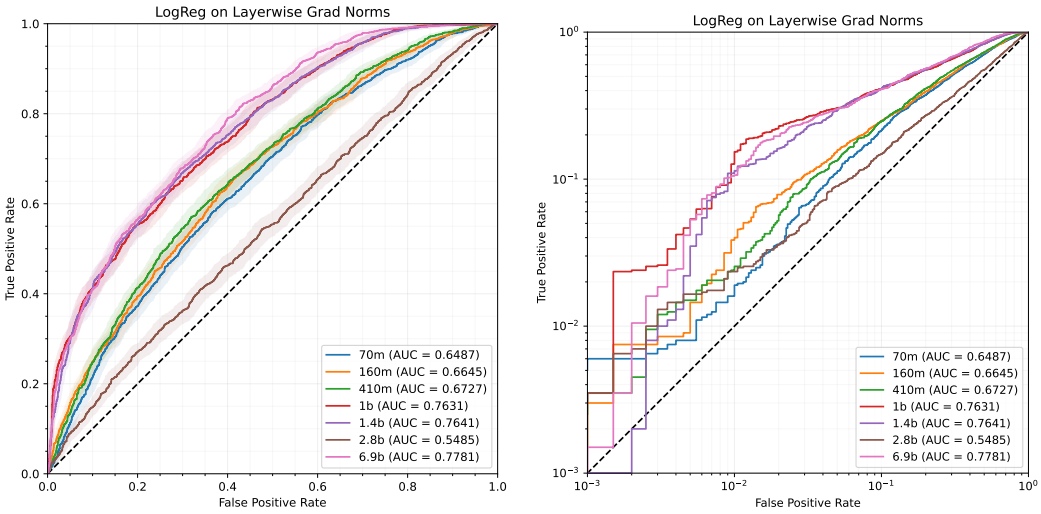


Figure 15. Logistic Regression on Layerwise Gradient Norm

Table 11. We train a **neural network** on the **norms of each layer's gradient**. We record AUC and TPR values for each Pythia model size.

Model Size	**AUC**	**TPR**$_{0.5\%}$	**TPR**$_{1\%}$	**TPR**$_{5\%}$
70M	0.674 ± 0.0163	0.0540 ± 0.0150	0.0835 ± 0.0205	0.207 ± 0.0386
160M	0.648 ± 0.0176	0.0590 ± 0.0148	0.0720 ± 0.0134	0.150 ± 0.0230
410M	0.623 ± 0.0178	0.0465 ± 0.0182	0.0555 ± 0.0135	0.123 ± 0.0309
1B	0.752 ± 0.0159	0.211 ± 0.0613	0.242 ± 0.0278	0.332 ± 0.0304
1.4B	0.732 ± 0.0161	0.146 ± 0.0304	0.171 ± 0.0206	0.259 ± 0.0230
2.8B	0.513 ± 0.0186	0.00900 ± 0.00526	0.0165 ± 0.00880	0.0765 ± 0.0231
6.9B	0.708 ± 0.0143	0.126 ± 0.0401	0.158 ± 0.0326	0.255 ± 0.0323

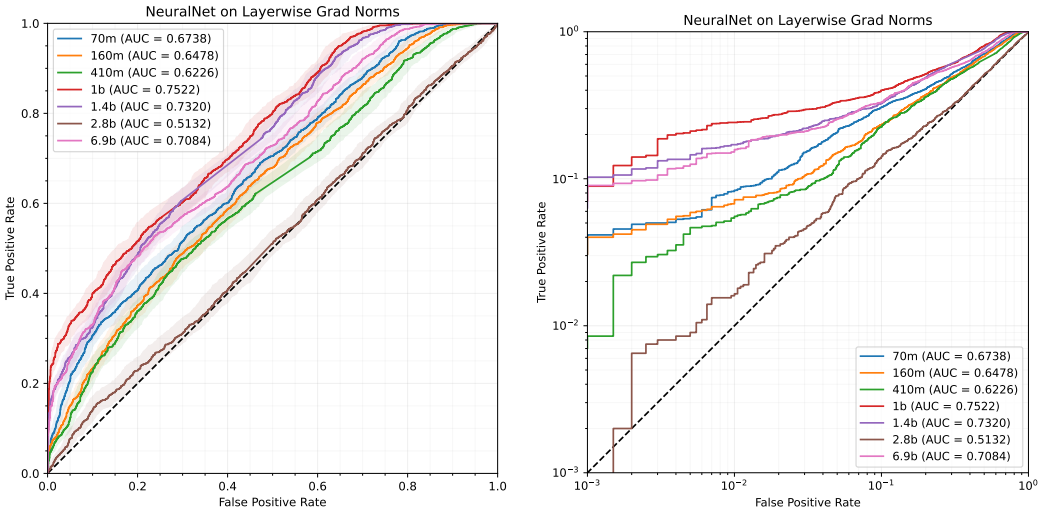


Figure 16. Neural Network on Layerwise Gradient Norm

G.3. Different Dimensionality Reductions

Table 12. Layerwise gradient norms are the key ingredient in our supervised MIA. To analyze its efficacy, we tried two other approaches for reducing the dimensionality of the gradient. First, we greedily concatenated gradients of layers of the model until reaching 8 approximately equal-sized vectors, and then applied a Johnson–Lindenstrauss transform for each to 512 dimensions. For instance, in Pythia-1B, each combined vector would be approximately 125 million parameters and projected to 512 dimensions. This is denoted below as "Combined JLs." The second approach we took was "balanced norms"; that is, for a model with M parameters, instead of taking the norms of each layer, we took the norms of the gradients of the first $\frac{M}{2000}$ parameters, then the next $\frac{M}{2000}$ parameters, etc. We record results below for the Pythia-1B model. Below, we see that our layerwise gradient norm solution performs the best across every metric, followed closely by the "balanced norm" approach, confirming that gradient norms contain important information for inferring membership.

Classifier	AUC	$\text{TPR}_{0.5\%}$	$\text{TPR}_{1\%}$	$\text{TPR}_{5\%}$
Original	0.786 ± 0.0127	0.0545 ± 0.0445	0.126 ± 0.0589	0.335 ± 0.0338
Balanced Norms	0.743 ± 0.0158	0.0355 ± 0.0202	0.0790 ± 0.0336	0.235 ± 0.0367
Combined JLs	0.591 ± 0.0196	0.0100 ± 0.00532	0.0190 ± 0.0109	0.0745 ± 0.0163

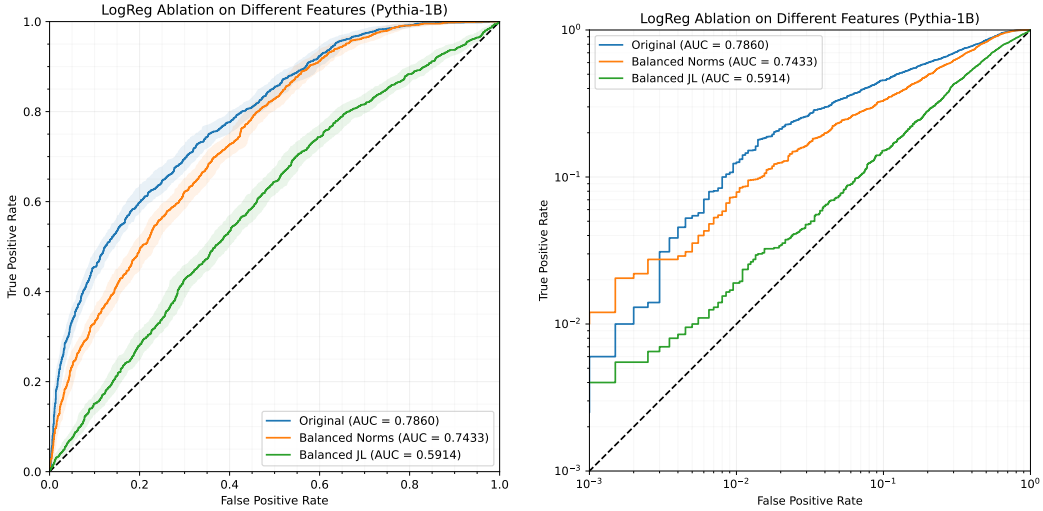


Figure 17. Logistic Regression on Different Dimensionality Reductions

G.4. Supervised MIA based on model stealing

(Carlini et al., 2024) developed an algorithm to steal the embedding projection layer of production LLMs using only standard API access. They successfully applied this attack to OpenAI's Ada and Babbage, and estimate it would only cost $2,000 to obtain the projection matrix of OpenAI's GPT-4. In our paper, we simulate this style of attack and design a gray-box MIA that uses only the information that can be stolen by the methods in (Carlini et al., 2024).

Lemma G.1. *Let h be the size of the domain of the embedding projection layer and $\mathcal{V}$ be the token dictionary, $|\mathcal{V}| = v, h \ll v$. Let the model be $\theta : \mathcal{V}^* \to \Delta(\mathcal{V})$ that map prompts to probability distributions over tokens. LLMs use a softmax to convert raw logits into probabilities, so we have $\theta(p) = \text{softmax}(\mathbf{W} \cdot g_\theta(p))$, where $\mathbf{W} \in \mathbb{R}^{v \times h}$ is the embedding projection matrix and $g_\theta : \mathcal{V}^T \to \mathbb{R}^h$ is the rest of the model. We let the logits $\mathcal{O}_\theta(p) = \mathbf{W} \cdot g_\theta(p)$. Given access to the logits $\mathcal{O}(p)$ for any prompt p, we can obtain the quantity $\nabla_{\mathbf{W}} \ell(\theta(p)) \cdot \mathbf{A}^{-\intercal}$ for some invertible $\mathbf{A} \in \mathbb{R}^{h \times h}$.*

Proof. To compute the embedding projection matrix, (Carlini et al., 2024) applies a SVD decomposition of $[\mathcal{O}_\theta(p)]_{p \in \mathcal{P}}$ across some set of prompts $\mathcal{P}$ to obtain a matrix $\tilde{\mathbf{W}} = \mathbf{W} \cdot \mathbf{A}$, where $\mathbf{A} \in \mathbb{R}^{h \times h}$ is some unknown invertible matrix. Thus we assume we have access to $\tilde{\mathbf{W}}$. Access to the logits allows us to compute the derivative $\nabla_{\mathcal{O}_\theta(p)} \ell(\theta(p))$, and we can also compute $\tilde{\mathbf{W}}^\dagger \mathcal{O}_\theta(p)$. The outer product of these two components is the loss gradient with respect to $\mathbf{W}$ up to multiplication

by $\mathbf{A}^{-\intercal}$,

$$\begin{aligned}\nabla_{\mathcal{O}_\theta(p)}\ell(\theta(p)) \cdot (\tilde{\mathbf{W}}^\dagger \mathcal{O}_\theta(p))^\intercal &= \nabla_{\mathcal{O}_\theta(p)}\ell(\theta(p)) \cdot (\mathbf{A}^{-1} g_\theta(p))^\intercal \\ &= \nabla_{\mathcal{O}_\theta(p)}\ell(\theta(p)) \cdot g_\theta(p)^\intercal \cdot \mathbf{A}^{-\intercal} \\ &= \nabla_{\mathbf{W}}\ell(\theta(p)) \cdot \mathbf{A}^{-\intercal}.\end{aligned}$$

□

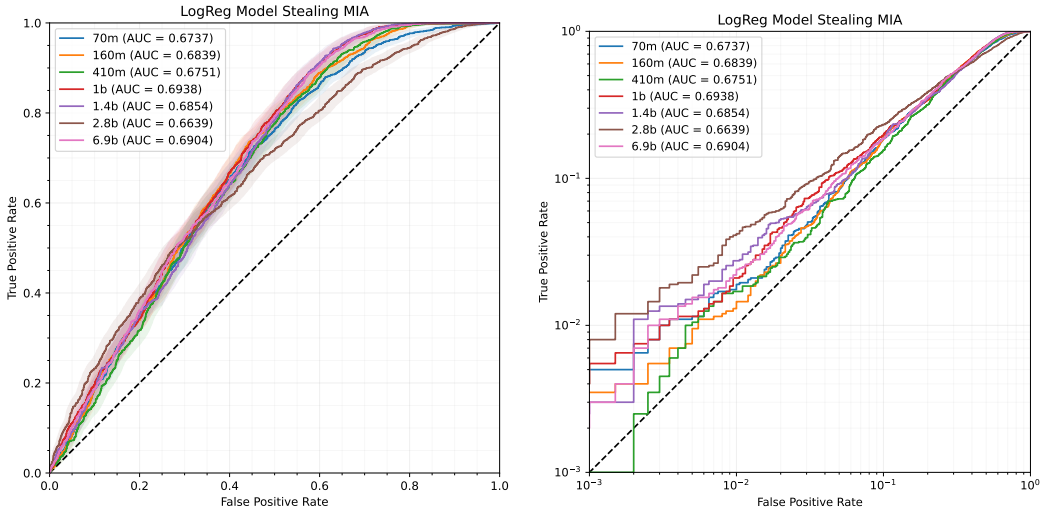


Figure 18. Logistic Regression on $\texttt{ModelStealing}_\theta$ Features

Table 13. $\texttt{ModelStealing}_\theta$ attack **logistic regression** on $2,000$ fine-tune and $2{,}000$ validation. We record AUC and TPR values for each Pythia model size with 95% confidence intervals over $1,000$ bootstraps.

Model Size	**AUC**	**TPR**$_{0.5\%}$	**TPR**$_{1\%}$	**TPR**$_{5\%}$
70M	0.674 ± 0.0162	0.0115 ± 0.00677	0.0190 ± 0.00734	0.0840 ± 0.0214
160M	0.684 ± 0.0169	0.00950 ± 0.00571	0.0145 ± 0.00703	0.0855 ± 0.0250
410M	0.675 ± 0.0172	0.0105 ± 0.00812	0.0170 ± 0.00704	0.0725 ± 0.0154
1B	0.694 ± 0.0160	0.0115 ± 0.00642	0.0210 ± 0.0109	0.111 ± 0.0202
1.4B	0.685 ± 0.0166	0.0150 ± 0.00860	0.0275 ± 0.0123	0.0920 ± 0.0192
2.8B	0.664 ± 0.0161	0.0220 ± 0.0107	0.0420 ± 0.0183	0.141 ± 0.0296
6.9B	0.690 ± 0.0158	0.0155 ± 0.00773	0.0240 ± 0.00945	0.102 ± 0.0230

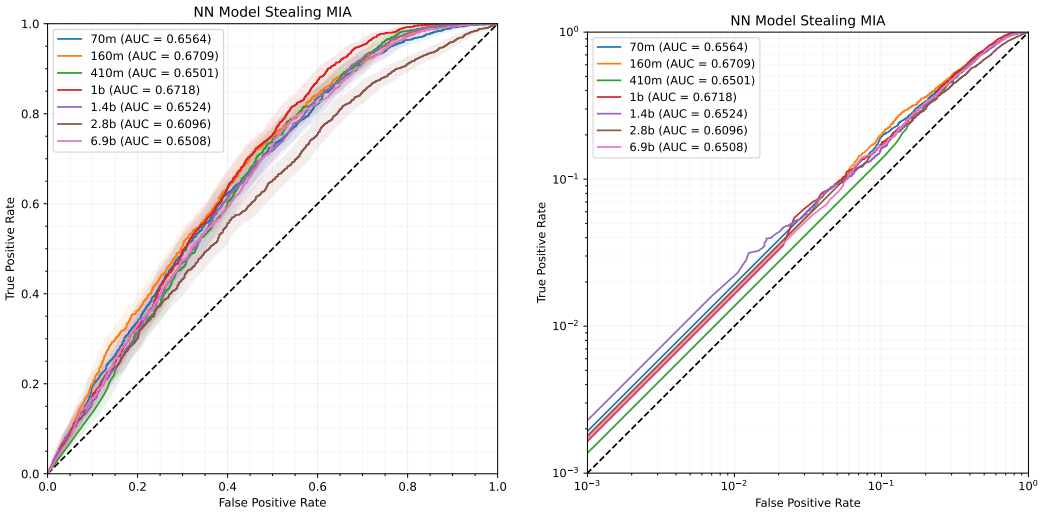


Figure 19. Neural Network on $\texttt{ModelStealing}_\theta$ Features

Table 14. $\texttt{ModelStealing}_\theta$ attack with **neural network** on 2,000 fine-tune and 2,000 validation. We record AUC and TPR values for each Pythia model size with 95% confidence intervals over 1,000 bootstraps.

Model Size	**AUC**	**TPR**$_{0.005}$	**TPR**$_{0.01}$	**TPR**$_{0.05}$
70M	0.656 ± 0.0171	0.00 ± 0.00195	0.00 ± 0.00439	0.0820 ± 0.0193
160M	0.671 ± 0.0160	0.00 ± 0.00211	0.00 ± 0.00475	0.0915 ± 0.0192
410M	0.650 ± 0.0166	0.00 ± 0.000921	0.00 ± 0.00207	0.00 ± 0.0113
1B	0.672 ± 0.0171	0.00 ± 0.00254	0.00 ± 0.00570	0.0945 ± 0.0201
1.4B	0.652 ± 0.0169	0.00 ± 0.00649	0.0160 ± 0.0113	0.0920 ± 0.0159
2.8B	0.610 ± 0.0178	0.00 ± 0.00149	0.00 ± 0.00335	0.00 ± 0.0183
6.9BM	0.651 ± 0.0164	0.00 ± 0.00235	0.00 ± 0.00528	0.0785 ± 0.0227

G.5. Fine-Tuned MIA Results

We fine-tuned 6 models (Pythia-70M, Pythia-160M, Pythia-410M, Pythia-1B, Pythia-1.4B, Pythia-2.8B) on 500 randomly selected points from the Pile's validation set for 1 epoch. In total, this consists of approximately $220,000$ tokens, or 330 pages of single-spaced text. Fine-tuned loss surprisingly does better than $\texttt{zlib\_LoRa}_\theta$ despite taking in less information. $\texttt{FLoRa}_{\theta,\theta'}$ is a near perfect MIA.

Table 15. Fine-tuned loss thresholding MIA on 500 fine-tune and 500 validation. We record AUC and TPR values for each Pythia model size with 95% confidence intervals over 1,000 bootstraps.

Model Size	**AUC**	**TPR**$_{0.5\%}$	TPR$_{1\%}$	**TPR**$_{5\%}$
70M	0.613 ± 0.0333	0.0461 ± 0.0230	0.0501 ± 0.0207	0.0962 ± 0.0398
160M	0.648 ± 0.0330	0.0501 ± 0.0207	0.0521 ± 0.0212	0.106 ± 0.0408
410M	0.701 ± 0.0281	0.0521 ± 0.0188	0.0541 ± 0.0225	0.112 ± 0.0415
1B	0.741 ± 0.0303	0.0521 ± 0.0188	0.0581 ± 0.0226	0.106 ± 0.0624
1.4B	0.760 ± 0.0306	0.0561 ± 0.0228	0.0601 ± 0.0241	0.132 ± 0.0701
2.8B	0.777 ± 0.0315	0.0541 ± 0.0167	0.0581 ± 0.0188	0.160 ± 0.0880

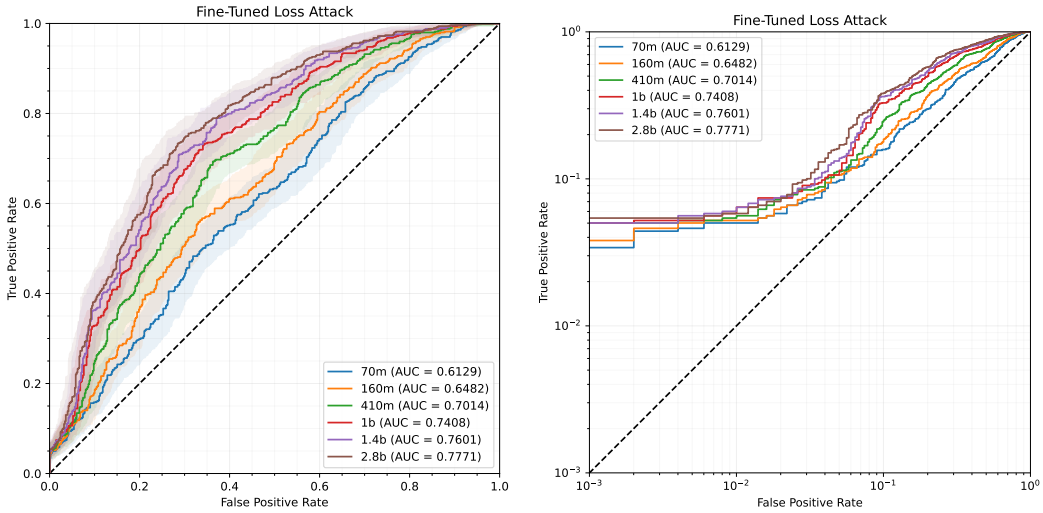


Figure 20. Fine-Tuned Loss MIA

Table 16. $\texttt{zlib\_LoRa}_\theta$ MIA on 500 fine-tune and 500 validation samples. We record AUC and TPR values for each Pythia model size with 95% confidence intervals over 1,000 bootstraps.

Model Size	**AUC**	**TPR**$_{0.5\%}$	**TPR**$_{0.1\%}$	**TPR**$_{5\%}$
70M	0.680 ± 0.0480	0.00 ± 0.00157	0.00 ± 0.00209	0.0200 ± 0.0772
160M	0.693 ± 0.0453	0.00 ± 0.00551	0.00 ± 0.00944	0.0200 ± 0.0311
410M	0.732 ± 0.0489	0.00 ± 0.00596	0.00 ± 0.00952	0.0160 ± 0.0457
1B	0.760 ± 0.0433	0.00 ± 0.00665	0.00 ± 0.0110	0.0200 ± 0.0477
1.4B	0.773 ± 0.0438	0.00 ± 0.00252	0.00 ± 0.00945	0.0560 ± 0.0539
2.8B	0.772 ± 0.0477	0.00 ± 0.00329	0.00 ± 0.0121	0.0280 ± 0.0432

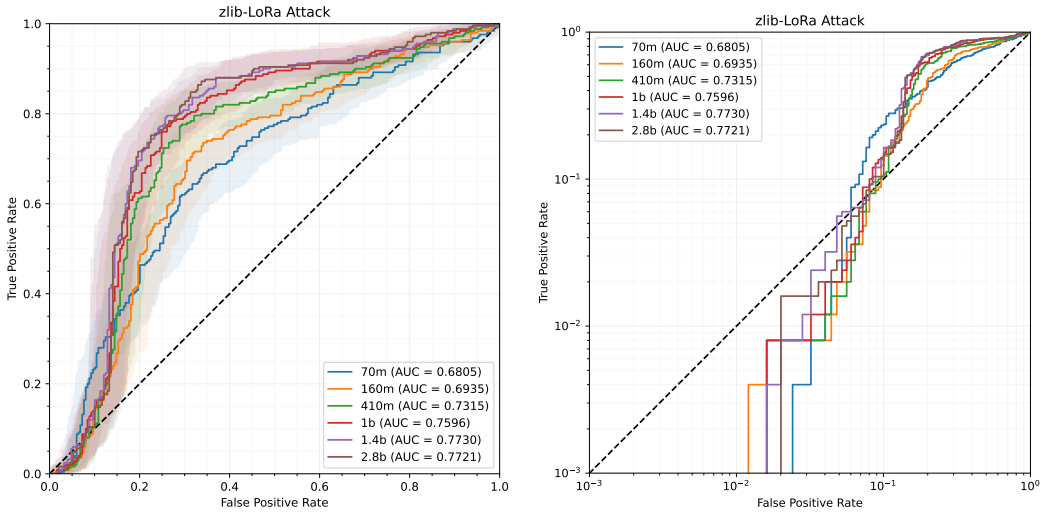


Figure 21. zlib-LoRa MIA

Table 17. $\texttt{FLoRa}_{\theta,\theta'}$ MIA on 500 fine-tune and 500 validation. We record AUC and TPR values for each Pythia model size and Llama-2 7B with 95% confidence intervals over 1,000 bootstraps.

Model Size	**AUC**	**TPR**$_{0.5\%}$	**TPR**$_{1\%}$	**TPR**$_{5\%}$
Pythia-70M	0.986 ± 0.00870	0.930 ± 0.315	0.948 ± 0.159	0.968 ± 0.0161
Pythia-160M	0.982 ± 0.0104	0.932 ± 0.0427	0.968 ± 0.0333	0.978 ± 0.0120
Pythia-410M	0.963 ± 0.0155	0.930 ± 0.0823	0.938 ± 0.0242	0.950 ± 0.0199
Pythia-1B	0.964 ± 0.0147	0.938 ± 0.128	0.944 ± 0.0216	0.956 ± 0.0176
Pythia-1.4B	0.967 ± 0.0150	0.942 ± 0.0225	0.942 ± 0.0209	0.960 ± 0.0185
Pythia-2.8B	0.947 ± 0.0201	0.914 ± 0.0326	0.926 ± 0.0244	0.936 ± 0.0211
Llama-2 7B	0.987 ± 0.00625	0.969 ± 0.0278	0.976 ± 0.0121	0.982 ± 0.00796
Llama-2 7B Chat	0.982 ± 0.00685	0.912 ± 0.0189	0.928 ± 0.0230	0.948 ± 0.0146

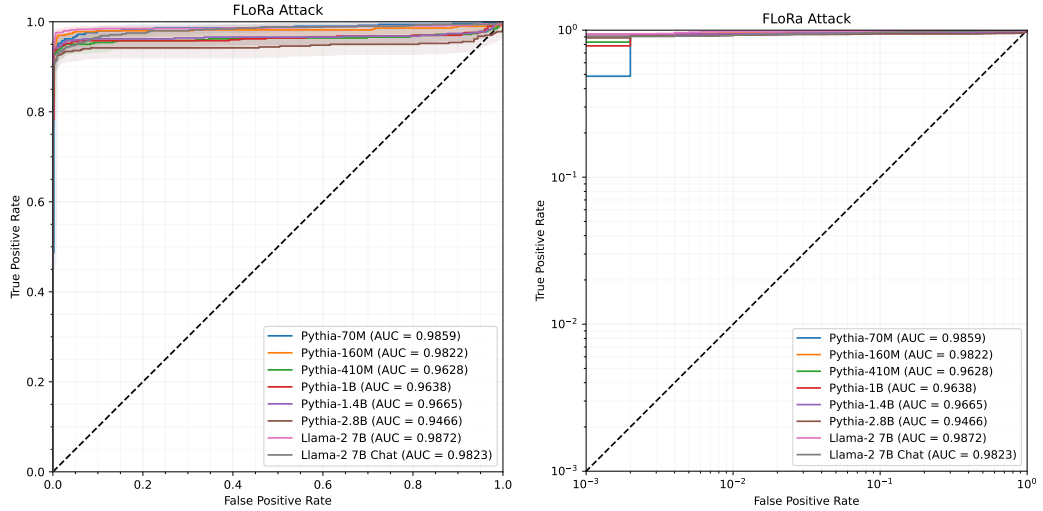


Figure 22. FLoRa MIA

H. Pretrained Extraction

As motivated in Section 4.2, we focus on developing a successful attack $\mathcal{M}$, as it gives a more realistic indication of the extraction risk for pretrained models.

Effective MIAs are not necessarily effective extraction rankers. Interestingly, we find that unlike in the fine-tuned setting, the success of our pretrained MIAs *does not* translate into ranking generated samples. For example, when we take 50-token prefixes of 500 randomly sampled points from Pythia's training split of The Pile, generate 20 suffixes on the Pythia-1.4B model, and use the supervised classifier from Section 3.1 that is meant to distinguish train vs. validation points (not train vs. generations), it often ranks the train point much lower than the generations (see Figure 24 in the Appendix).

We therefore ask if a performant ranker $\mathcal{M}$ for this extraction task is possible. We demonstrate below that our supervised white-box attacks (denoted `Gen-LogReg`, `Gen-NN`) can do so, where rather than training on gradient norms of train/validation points, we use gradient norms of training points vs. samples that share the same 50-token prefix but have a generated suffix. We evaluate our classifier in two ways. First, we use an AUC metric on a separate 2,000 train/generation pairs. Second, to test on an actual extraction task, we take the same 2,000 prefixes from true training data, but now try to rank one 100 token train point against 20 samples where we generate the last 50 tokens from the train prefix using our LLM. We then measure how often `Gen-LogReg` and `Gen-NN` can pick out the true train point amongst 20 generations. On Pythia-1.4B, our supervised attacks identify the correct suffix 90.1% and 88.3% of the time whereas naively apply the `LogReg`, `NN` MIAs only succeed $15.2\%, 22.1\%$ of the time, respectively. In Table H we outline full statistics comparing `Gen-LogReg` and `Gen-NN` against $\texttt{LogReg}_\theta$ and $\texttt{NN}_\theta$, with full distributions of rankings assigned to the true suffix, as well as AUC plots, in Appendix I.

Table 18. Table comparing $\texttt{LogReg}_\theta$ and $\texttt{NN}_\theta$, the MIAs, against classifiers specifically trained to distinguish training points vs. generation points that share the same prefix: `Gen-LogReg` and `Gen-NN`. The accuracy is the number of top-ranked suffixes that are the true suffix. The percentile rank of the classifier score for the train point is the order statistic of the true point's ranking (a percentile rank of 100% means the classifier correctly identified the train point).

Attack	AUC	Accuracy	Avg. Percentile	$\text{TPR}_{1\%}$	$\text{TPR}_{5\%}$
$\texttt{LogReg}_\theta$	0.631	0.152	0.548	0.035	0.138
$\texttt{NN}_\theta$	0.695	0.221	0.521	0.151	0.214
`Gen-LogReg`	0.948	0.901	0.975	0.448	0.757
`Gen-NN`	0.946	0.883	0.973	0.548	0.727

I. MIA vs. Generation Classifier in the Pretrain Extraction Setting

In this section, we consider the application of our MIAs, which were trained to distinguish between training and validation data, and Generation classifiers, which were trained to distinguish between training and generations, for a pre-train extraction pipeline–specifically on the goal to distinguish training data and generations. We see that the Generation classifiers greatly outperform the MIA on this task.

Table 19. Comparing the performance of a MIA vs. a specifically trained classifier for distinguishing generations from the Pythia-1.4B model.

Classifier	AUC	$\text{TPR}_{0.5\%}$	$\text{TPR}_{1\%}$	$\text{TPR}_{5\%}$
$\texttt{LogReg}_\theta$ MIA	0.631 ± 0.0179	0.0200 ± 0.00883	0.0345 ± 0.0147	0.138 ± 0.0293
$\texttt{NN}_\theta$ MIA	0.695 ± 0.0180	0.132 ± 0.0305	0.151 ± 0.0216	0.214 ± 0.0238
$\texttt{LogReg}_\theta$ Classifier	0.948 ± 0.00624	0.319 ± 0.134	0.448 ± 0.105	0.757 ± 0.0421
$\texttt{NN}_\theta$ Classifier	0.946 ± 0.00615	0.474 ± 0.0765	0.548 ± 0.0583	0.947 ± 0.0139

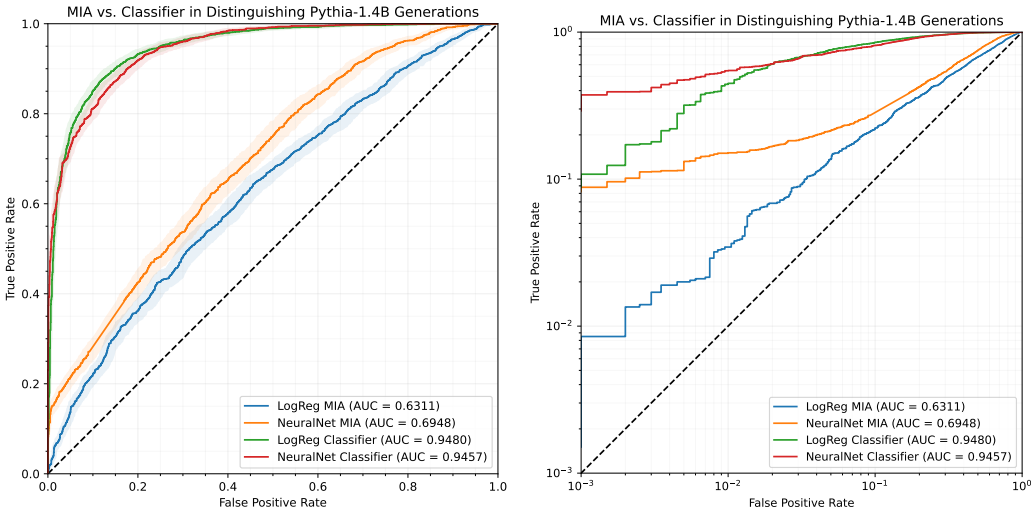


Figure 23. AUC of $\texttt{NN}_\theta$, $\texttt{LogReg}_\theta$, Gen-NN, and Gen-LogReg.

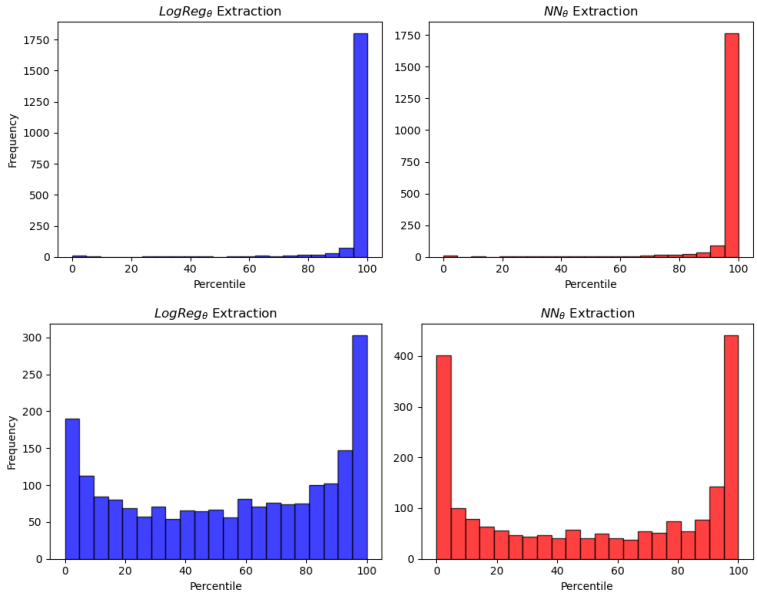


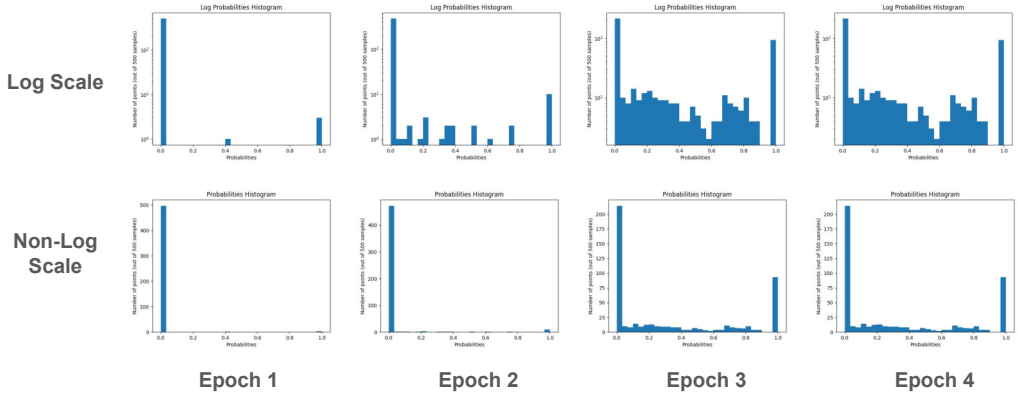
Figure 24. As noted in Section H, an effective MIA is not necessarily an effective extraction ranker (distinguishing true suffixes from generated ones). Here, we plot histograms of the percentile rank of the classifier score for the train point among the classifier scores for the twenty generations. A percentile rank of 100% means the classifier correctly identified the train point. **Above:** a classifier trained to distinguish train points vs. generations succeeds over 90% of the time on test data. **Below:** a classifier trained as an MIA (train vs. validation points) only achieves 15.2% and 22.1% accuracy on distinguishing train points from generations. See Table H for a concise, in-text summary.

J. Full Fine-Tune Extraction Results

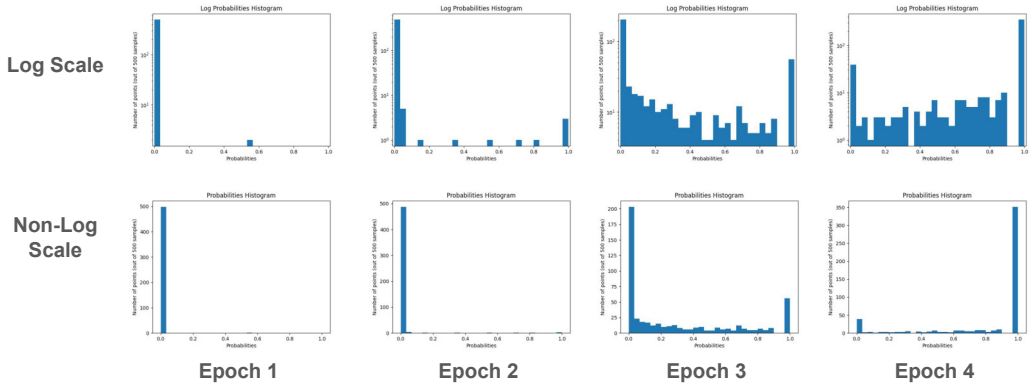
Hyperparameter Choice During Generation Sampling. Generation is sensitive to sampling hyperparameters. To save on computational cost, we use the best hyperparameters published in (Yu et al., 2023) for data extraction in this paper for sampling from our LLMs, which are top-k sampling with $k = 24$, nucleus/top-p sampling with $p = 0.8$, typical-p sampling with $p = 0.9$, temperature of 0.58, and a repetition penalty of 1.04.

Probability Dynamics in Discoverable Extraction We illustrate the evolution of suffix probabilities in the discoverable extraction setting (50 token prefixes and suffixes), for Pythia-1B, Pythia-1.4B, Pythia-2.8B, Pythia-6.9B, Llama-7B, and Llama-7B-chat (Figures 25 and 26). Notably, as the models get larger, more and more data is memorized in Epoch 2.

(a) Pythia-1b fine-tune suffix probability histograms, 50 token prefixes/suffixes, epochs 1-4



(b) Pythia-1.4b fine-tune suffix probability histograms, 50 token prefixes/suffixes, epochs 1-4



(c) Pythia-2.8b fine-tune suffix probability histograms, 50 token prefixes/suffixes, epochs 1-4

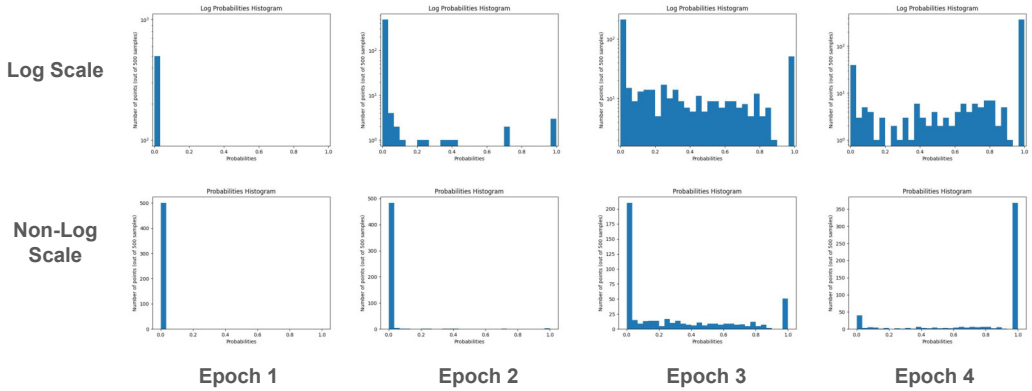
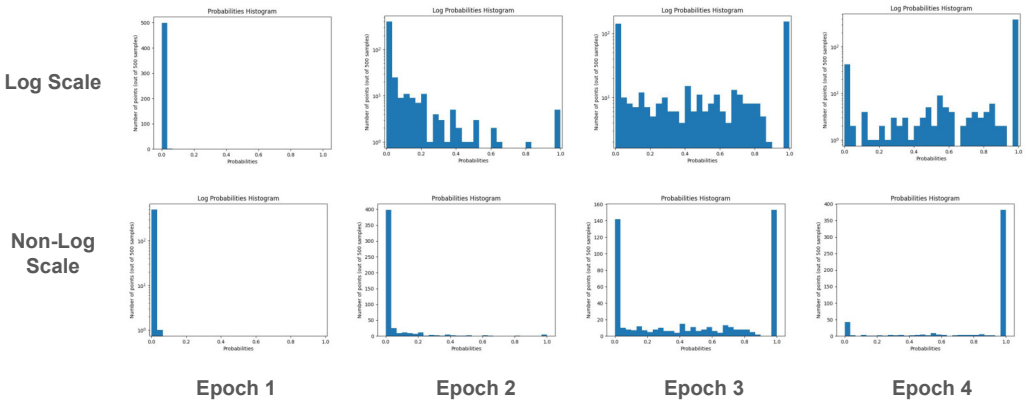


Figure 25. These are the histograms of suffix probabilities for the 500 samples across the 4 epochs of fine-tuning in Pythia-1.4B, Pythia-2.8B, and Pythia-6.9B.

Non-Discoverable Extraction. In Table 21, we show probabilities of suffixes in non-discoverable fine-tuning extraction results in the fine-tuning setting. As noted in Section H of the paper, successful extraction largely hinges on whether a sample is memorized well-enough during fine-tuning, since $\texttt{FLoRa}_{\theta,\theta'}$ almost always selects the true suffix when it is generated.

(a) Llama-7b fine-tune suffix probability histograms, 50 token prefixes/suffixes, epochs 1-4



(b) Llama-7b-chat fine-tune suffix probability histograms, 50 token prefixes/suffixes, epochs 1-4

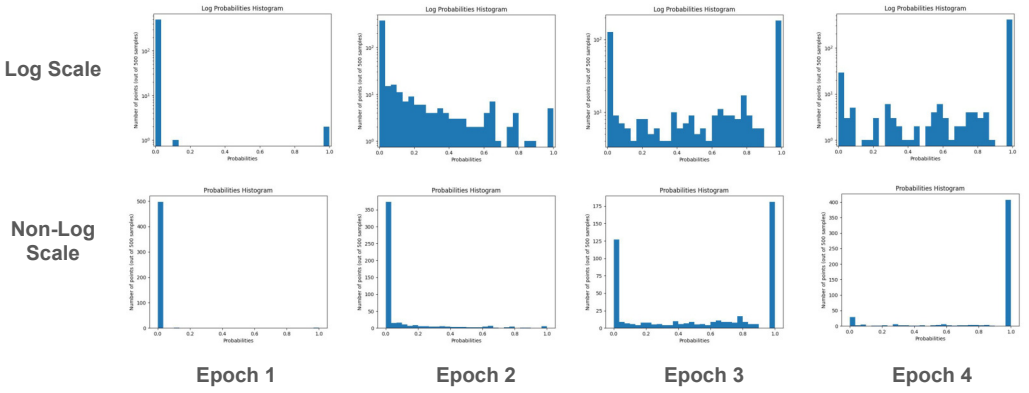


Figure 26. These are the histograms of suffix probabilities for the 500 samples across the 4 epochs of fine-tuning in Llama-7B and Llama-7B-chat.

Table 20. **Discoverable extraction** ($k = 50$, $m = 50$). We report the precision and 01-Hamming with 20 generations and $\texttt{FLoRa}_{\theta,\theta'}$ as a ranking metric (rows 1-2), and the proportion of suffixes with generation probabilities $p > .01, .05, .01, .001$ (rows 3-6). Notice the bolded **0.592** underneath Epoch 3 for Pythia-2.8B's precision. This number has the following implications: across all suffixes, 65.2% had probability >0.01 of being generated and 64.6% had probability >0.05 of being generated. After 20 generations for each sample and ranking with $\texttt{FLoRa}_{\theta,\theta'}$, we accurately recover 59.2% of suffixes (precision) and have an average token-by-token match rate between top-ranked generation and true suffix of 78.8% (01-Hamming).

Model Size	Metric	Epoch 1	Epoch 2	Epoch 3	Epoch 4
Pythia-1B	Precision	0	0.016	0.536	0.798
	01-Hamming	0.059	0.242	0.768	0.901
	$p>0.1$	0	0.018	0.562	0.908
	$p>0.05$	0	0.022	0.622	0.91
	$p>0.01$	0	0.046	0.662	0.914
	$p>0.001$	0	0.08	0.68	0.914
Pythia-1.4B	Precision	0	0.016	0.472	0.874
	01-Hamming	0.06472	0.240	0.757	0.940
	$p>0.1$	0.004	0.016	0.512	0.912
	$p>0.05$	0.004	0.022	0.566	0.92
	$p>0.01$	0.004	0.05	0.63	0.924
	$p>0.001$	0.008	0.08	0.646	0.924
Pythia-2.8B	Precision	0.008	0.078	0.592	0.898
	01-Hamming	0.0902	0.364	0.788	0.953
	$p>0.1$	0.008	0.094	0.624	0.91
	$p>0.05$	0.008	0.114	0.646	0.912
	$p>0.01$	0.008	0.154	0.652	0.914
	$p>0.001$	0.01	0.166	0.654	0.914
Llama-7B	Precision	0.006	0.098	0.602	0.902
	01-Hamming	0.082	0.396	0.792	0.949
	$p>0.1$	0.006	0.134	0.686	0.914
	$p>0.05$	0.008	0.156	0.696	0.916
	$p>0.01$	0.008	0.19	0.708	0.918
	$p>0.001$	0.012	0.204	0.712	0.92
Llama-7B-Chat	Precision	0.004	0.144	0.622	0.892
	01-Hamming	0.103	0.478	0.812	0.954
	$p>0.1$	0.008	0.186	0.658	0.92
	$p>0.05$	0.008	0.216	0.668	0.92
	$p>0.01$	0.008	0.25	0.678	0.92
	$p>0.001$	0.01	0.266	0.684	0.92

Table 21. Non-Discoverable Extraction. In the following table, we provide the proportion of suffixes with a probability above a certain threshold, for all Pythia model sizes across 1-3 epochs of fine-tuning, in all non-discoverable extraction settings. For instance, the column $k = 2, m = 25$ corresponds to the probability of the 25-length suffix after a 2-token prefix of a fine-tune sample.

Model	Threshold	$(2, 25)$	$(2, 50)$	$(4, 25)$	$(4, 50)$	$(8, 25)$	$(8, 50)$
Pythia-1B Epoch 1	$p > 0.1$	0.004	0	0.008	0	0.01	0
	$p > 0.05$	0.004	0	0.008	0	0.01	0
	$p > 0.01$	0.01	0	0.016	0	0.016	0
	$p > 0.001$	0.014	0	0.02	0	0.016	0
1B Epoch 2	$p > 0.1$	0.074	0.008	0.08	0.014	0.11	0.012
	$p > 0.05$	0.098	0.016	0.118	0.02	0.144	0.028
	$p > 0.01$	0.172	0.03	0.208	0.04	0.236	0.064
	$p > 0.001$	0.244	0.062	0.294	0.078	0.312	0.088
1B Epoch 3	$p > 0.1$	0.57	0.432	0.722	0.536	0.76	0.57
	$p > 0.05$	0.6	0.486	0.76	0.588	0.802	0.62
	$p > 0.01$	0.65	0.54	0.786	0.654	0.834	0.68
	$p > 0.001$	0.668	0.576	0.806	0.69	0.844	0.708
Pythia-1.4B Epoch 1	$p > 0.1$	0.004	0	0.004	0.004	0.01	0.02
	$p > 0.05$	0.004	0.004	0.004	0.004	0.01	0.006
	$p > 0.01$	0.004	0.004	0.006	0.004	0.012	0.006
	$p > 0.001$	0.008	0.004	0.008	0.004	0.014	0.008
1.4B Epoch 2	$p > 0.1$	0.07	0.006	0.08	0.01	0.11	0.014
	$p > 0.05$	0.116	0.008	0.13	0.012	0.158	0.018
	$p > 0.01$	0.186	0.018	0.226	0.032	0.236	0.044
	$p > 0.001$	0.258	0.054	0.312	0.072	0.332	0.084
1.4B Epoch 3	$p > 0.1$	0.528	0.378	0.644	0.458	0.695	0.51
	$p > 0.05$	0.572	0.418	0.68	0.502	0.738	0.558
	$p > 0.01$	0.618	0.472	0.732	0.566	0.778	0.63
	$p > 0.001$	0.618	0.502	0.74	0.596	0.788	0.656
Pythia-2.8B Epoch 1	$p > 0.1$	0	0	0.006	0	0.004	0
	$p > 0.05$	0.004	0	0.006	0	0.008	0
	$p > 0.01$	0.006	0	0.01	0	0.008	0
	$p > 0.001$	0.008	0	0.01	0	0.014	0
2.8B Epoch 2	$p > 0.1$	0.11	0.014	0.126	0.018	0.13	0.022
	$p > 0.05$	0.134	0.02	0.162	0.028	0.166	0.03
	$p > 0.01$	0.21	0.048	0.248	0.06	0.254	0.052
	$p > 0.001$	0.274	0.082	0.332	0.096	0.346	0.086
2.8B Epoch 3	$p > 0.1$	0.512	0.352	0.614	0.45	0.714	0.518
	$p > 0.05$	0.552	0.386	0.652	0.482	0.75	0.568
	$p > 0.01$	0.58	0.44	0.678	0.538	0.77	0.61
	$p > 0.001$	0.594	0.46	0.686	0.554	0.774	0.622
Pythia-6.9B Epoch 1	$p > 0.1$	0.01	0.004	0.018	0.004	0.024	0.008
	$p > 0.05$	0.012	0.004	0.02	4	0.026	0.008
	$p > 0.01$	0.014	0.006	0.024	0.01	0.034	0.01
	$p > 0.001$	0.024	0.006	0.032	0.01	0.046	0.012
6.9B Epoch 2	$p > 0.1$	0.28	0.094	0.306	0.116	0.324	0.114
	$p > 0.05$	332	0.122	0.37	0.154	0.384	0.154
	$p > 0.01$	0.414	0.19	0.47	0.218	0.47	0.228
	$p > 0.001$	0.45	0.234	0.516	0.272	0.548	0.29
6.9B Epoch 3	$p > 0.1$	0.612	0.524	0.75	0.64	0.816	0.708
	$p > 0.05$	0.628	0.55	0.774	0.678	0.844	0.74
	$p > 0.01$	0.662	0.584	0.8	0.708	0.856	0.766
	$p > 0.001$	0.67	0.596	0.804	0.714	0.862	0.77

Table 22. Non-Discoverable Extraction. In the following table, we provide the proportion of suffixes with a probability above a certain threshold, for **Llama-2 7B and Llama-2 7B-Chat** across 1-3 epochs of fine-tuning, in all non-discoverable extraction settings. For instance, the column $k = 2, m = 25$ corresponds to the probability of the 25-length suffix after a 2-token prefix of a fine-tune sample.

Model	**Threshold**	$(\mathbf{2}, \mathbf{25})$	$(\mathbf{2}, \mathbf{50})$	$(\mathbf{4}, \mathbf{25})$	$(\mathbf{4}, \mathbf{50})$	$(\mathbf{8}, \mathbf{25})$	$(\mathbf{8}, \mathbf{50})$
7B Epoch 1	$p > 0.1$	0.002	0	0	0	0.002	0
	$p > 0.05$	0.002	0	0	0	0.004	0
	$p > 0.01$	0.004	0	0.004	0	0.006	0.002
	$p > 0.001$	0.016	0	0.006	0	0.016	0.002
7B Epoch 2	$p > 0.1$	0.212	0.072	0.22	0.07	0.268	0.086
	$p > 0.05$	0.246	0.106	0.264	0.106	0.308	0.112
	$p > 0.01$	0.308	0.146	0.338	0.146	0.398	0.17
	$p > 0.001$	0.346	0.188	0.374	0.188	0.446	0.226
7B Epoch 3	$p > 0.1$	0.534	0.448	0.584	0.488	0.768	0.626
	$p > 0.05$	0.56	0.474	0.606	0.52	0.796	0.666
	$p > 0.01$	0.602	0.516	0.642	0.562	0.814	0.694
	$p > 0.001$	0.612	0.53	0.652	0.568	0.82	0.704
7B-Chat Epoch 1	$p > 0.1$	0.004	0	0.002	0	0.01	0.004
	$p > 0.05$	0.004	0	0.01	0.004	0.012	0.004
	$p > 0.01$	0.012	0	0.018	0.006	0.018	0.004
	$p > 0.001$	0.032	0.002	0.032	0.006	0.032	0.006
7B-Chat Epoch 2	$p > 0.1$	0.29	0.136	0.328	0.146	0.394	0.16
	$p > 0.05$	0.338	0.168	0.368	0.178	0.45	0.202
	$p > 0.01$	0.402	0.234	0.426	0.246	0.538	0.282
	$p > 0.001$	0.444	0.288	0.474	0.308	0.594	0.35
7B-Chat Epoch 3	$p > 0.1$	0.578	0.488	0.63	0.532	0.814	0.686
	$p > 0.05$	0.598	0.506	0.648	0.554	0.83	0.716
	$p > 0.01$	0.628	0.536	0.678	0.582	0.846	0.726
	$p > 0.001$	0.638	0.552	0.682	0.592	0.85	0.736