MoPe: Model Perturbation-based Privacy Attacks on Language Models

Marvin Li[*]
Harvard College
marvinli@college.harvard.edu

Jason Wang [*]
Harvard College
jasonwang1@college.harvard.edu

Jeffrey Wang [*]
Harvard College
jgwang@college.harvard.edu

Seth Neel
Harvard Business School
sneel@hbs.edu

Abstract

Recent work has shown that Large Language Models (LLMs) can unintentionally leak sensitive information present in their training data. In this paper, we present $\texttt{MoPe}_\theta$ (**Mo**del **Pe**rturbations), a new method to identify with high confidence if a given text is in the training data of a pre-trained language model, given white-box access to the models parameters. $\texttt{MoPe}_\theta$ adds noise to the model in parameter space and measures the drop in the log-likelihood for a given point x, a statistic we show approximates the trace of the Hessian matrix with respect to model parameters. We compare $\texttt{MoPe}_\theta$ to existing state-of-the-art loss-based attacks and other attacks based on second-order curvature information (such as the trace of the Hessian with respect to the model input). Across language models ranging from size 70M to 12B parameters, we show that $\texttt{MoPe}_\theta$ is more effective than existing loss-based attacks. We also find that the loss of a point alone is insufficient to determine extractability—there are training points we can recover using our methods that have average loss. This casts some doubt on prior work that uses the loss of a point as evidence of memorization or "unlearning."

1 Introduction

Over the last few years, Large Language Models or LLMs have set new standards in performance across a range of tasks in natural language understanding and generation, often with very limited supervision on the task at hand [Brown et al. (2020)]. As a result, opportunities to use these models in real-world applications proliferate, and companies are rushing to deploy them in applications as diverse as A.I. assisted clinical diagnoses [Sharma et al. (2023)], NLP tasks in finance [Wu et al. (2023)], or an A.I. "love coach" [Soper (2023)]. While early state of the art LLMs have been largely trained on public web data [Radford et al. (2019); Gao et al. (2021); Biderman et al. (2023b); Black et al. (2021)], increasingly models are being fine-tuned on data more relevant to their intended domain, or even trained from scratch on this domain specific data. In addition to increased performance on a range of tasks, training models from scratch is attractive to companies because early work has shown it can mitigate some of the undesirable behavior associated with LLMs such as hallucination [Ji et al. (2022)], toxicity [Gehman et al. (2020)], as well as copyright issues that may arise from mimicking the training data [Franceschelli and Musolesi (2021); Vyas et al. (2023)]. For example, `BloombergGPT` [Wu et al. (2023)] is a 50-billion parameter auto-regressive language model that was trained from scratch on financial data sources, and exhibits superior performance on tasks in financial NLP.

[*] Alphabetical order; equal contribution.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

While all of this progress seems set to usher in an era where companies deploy custom LLMs trained on their proprietary customer data, one of the main technical barriers that remains is *privacy*. While LLMs don't overfit to training data to the same extent as other over-parameterized models, training points do in general have lower loss than non-train points. This allows adversaries to perform what is called a membership inference attack [Shokri et al. (2017)]: given access to the model and a candidate sample x', the adversary can determine with high-accuracy if x' is in the training set. While prior work shows privacy is a real concern when deploying language models, the loss-based membership inference attacks used can extract specific points but perform quite poorly on average, leaving significant room for improvement [Yu et al. (2023); Carlini et al. (2021)]. Other recent studies on data deletion [Jang et al. (2022)] and memorization [Carlini et al. (2023a)] in pre-trained LLMs use the loss of a point to determine whether that point has been "deleted" from the model, or conversely is "memorized." Given that the loss of a point is a relatively poor indicator of training set membership for LLMs, however, this raises a series of tantalizing questions we address here: (i) *If a point has average "loss" with respect to a language model θ, does that imply it cannot be detected as a training point?* (ii) *Can we develop stronger MIA attacks against LLMs than existing loss-based attacks*?

Contributions We present several contributions:

- We develop a new membership inference attack we call $\texttt{MoPe}_\theta$, based on the idea that when the model loss is localized around a training point, θ is likely to lie in a "sharper" local minima than if the point was a test point. Concretely, we do this by perturbing the model's parameters and measuring the change in loss (see Figure 1), a statistic we show approximates the trace of the Hessian with respect to model parameters, $\text{Tr}(\mathbf{H}_\theta)$.
- We are the first to evaluate the `DetectGPT` statistic of [Mitchell et al. (2023)] for the task of membership inference against pre-trained LLMs, a related perturbation method first developed for LLM content detection; we show it outperforms attacks based on the loss.
- Using the Pythia suite from EleutherAI [Biderman et al. (2023b)], we benchmark $\texttt{MoPe}_\theta$ and other attacks on models that range in size from 70M to 12B parameters.
- We demonstrate that $\texttt{MoPe}_\theta$ significantly outperforms other attacks on models smaller than 1.4B parameters, with comparable results on larger models.

These results establish our techniques as state of the art for MIA against pre-trained LLMs in the white-box setting where we can access the model weights. They also challenge conventional wisdom that the loss of a point is in isolation a sufficently good proxy for training set membership.

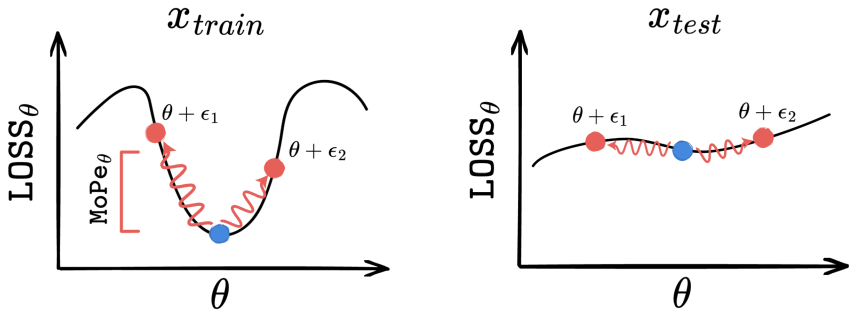


Figure 1: $\texttt{MoPe}_\theta$ detects training point membership by comparing the increase in loss on the point when the model is perturbed, relative to the increase in loss of the perturbed model on a test point.

2 Membership Inference Attacks on LLMs

Given a training corpus $\mathcal{C}$ sampled i.i.d from a distribution $\mathcal{D}$ over sequences of tokens, autoregressive language models optimize their parameters θ to minimize the average loss, $\frac{1}{|\mathcal{C}|} \min_\theta \sum_{x \in \mathcal{C}} \texttt{LOSS}_\theta(x)$, where $\texttt{LOSS}_\theta$ is defined as the negative log-likelihood l of the sequence with respect to θ. This is

typically referred to as "pre-training" on the task of next token prediction, to differentiate it from "fine-tuning" for other downstream tasks which occurs subsequent to pre-training.

A membership inference attack (MIA) against a model θ assigns a score $\mathcal{M}(x, \theta)$ to a data point x indicating its confidence in whether x is in the training set of θ. To test a canonical membership inference attack, we sample a random point $x \in \mathcal{C}$ with probability $\frac{1}{2}$; otherwise, we sample $x \sim \mathcal{D}$. We then compute $\mathcal{M}(x, \theta)$ and threshold over some value τ, where we say x is a train point if $\mathcal{M}(x, \theta) > \tau$ and not otherwise. Throughout the paper, as is common, we overload notation and refer to $\mathcal{M}$ as a membership inference attack rather than specifying a specific threshold. By construction, if $\mathcal{M}$ has accuracy above $\frac{1}{2}$, θ must be leaking information about x.

The most commonly used membership inference attack, introduced in [Yeom et al. (2018)], takes $\mathcal{M}(x, \theta) = -\ell(x, \theta)$; it predicts points are training points if their loss is less than $-\tau$, or equivalently the confidence is greater than τ. We refer to this attack throughout as the loss attack, or $\texttt{LOSS}_\theta$. Different MIAs exist for language models in the fine-tuned and pre-trained settings. In the fine-tuned setting, MIAs have been proposed [Ye et al. (2021); Carlini et al. (2022); Sablayrolles et al. (2019)] that calibrate the threshold at which a point x is declared a training point to account for the component of the loss that is specific to the example x. These approaches rely on training "shadow models", an approach that assumes an adversary can train additional language models from scratch on fresh data, or have access to a reference model that has been trained without the candidate point x (e.g. attacks against fine-tuned language models). In the pre-trained setting, however, such attacks are computationally impractical and so the state-of-the-art is loss-based thresholding. As such, we focus our attention on MIAs against pre-trained models in this paper.

3 Perturbation Privacy Attacks

In this section, we introduce the formal underpinnings of perturbation-based privacy attacks.

$\texttt{MoPe}_\theta$. $\texttt{MoPe}_\theta$ is based on the intuition that the loss landscape with respect to the weights should be different around training versus testing points. In particular, we expect that since $\theta \approx \text{argmin}_\Theta \sum_{x \in \mathcal{C}} \ell(x, \theta)$, the loss around $x' \in \mathcal{C}$ should be sharper than around a random point. Formally, given a candidate point $x \sim \mathcal{D}$, we define:

$$\texttt{MoPe}_\theta(x) = \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \sigma^2 I_{n_{\text{params}}})}[\ell(x, \theta + \epsilon) - \ell(x, \theta)], \tag{1}$$

where σ^2 is a variance hyperparameter that we specify beforehand, and $\theta \in \mathbb{R}^{n_{\text{params}}}$. In practice, rather than computing this expectation, we sample n noise values $\epsilon_i \sim \mathcal{N}(0, \sigma^2 I_{n_{\text{params}}})$ and compute the empirical $\texttt{MoPe}^n_\theta(x) = \frac{1}{n} \sum_{i=1}^n [\ell(x, \theta + \epsilon_i) - \ell(x, \theta)]$.

This gives rise to the natural $\texttt{MoPe}_\theta$ thresholding attack, with our MIA statistic defined as $\mathcal{M}(x, \theta) = \texttt{MoPe}_\theta(x)$. Note that the time complexity of computing the $\texttt{MoPe}_\theta$ statistic scales linearly with the number of perturbed models and so we typically use ≤ 20 perturbed models for computational reasons. We now provide some theoretical grounding for our method.

In order to derive a more intuitive expression for $\texttt{MoPe}_\theta(x)$ we start with the multivariate Taylor approximation [Königsberger (2000)]:

$$\ell(\theta + \epsilon, x) = \ell(\theta, x) + \nabla_\theta \ell(\theta, x) \cdot \epsilon + \tag{2}$$

$$\frac{1}{2} \epsilon^T \mathbf{H}_\theta \epsilon + O(\epsilon^3) \tag{3}$$

where $\mathbf{H}_\theta = \nabla^2_\theta \ell(\theta, x)$ is the Hessian of log-likelihood with respect to θ evaluated at x. Then assuming σ^2 is sufficiently small that $O(\epsilon^3)$ in Equation 2 is negligible, rearranging terms and taking the expectation with respect to ϵ of both sides of (2) we get:

$$\texttt{MoPe}_\theta(x) = \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \sigma^2 I_{n_{\text{params}}})}[\ell(\theta + \epsilon, x) - \ell(\theta, x)] \approx$$

$$\mathbb{E}_{\epsilon \sim \mathcal{N}(0, \sigma^2 I_{n_{\text{params}}})}[\frac{1}{2} \epsilon^T \mathbf{H}_\theta \epsilon] = \frac{\sigma^2}{2} \text{Tr}(\mathbf{H}_\theta),$$

where the last identity is known as the Hutchinson Trace Estimator [Hutchinson (1989)]. This derivation sheds some light on the importance of picking an appropriate value of σ. We need it to be small enough so that the Taylor approximation holds in Equation 2, but large enough that we

have enough precision to actually distinguish the difference in the log likelihoods in Equation 1. Empirically, we find that $\sigma = 0.005$ works well across all model sizes, and we don't observe a significant trend on the optimal value of σ (see Table 2 in the Appendix).

DetectGPT. The most closely related work to our $\texttt{MoPe}_\theta$ method on a technical level is the recent paper [Mitchell et al. (2023)], who propose a perturbation-based method for detecting LLM-generated text using probability curvature. Their method DetectGPT compares the log probabilities of a candidate passage x and its randomly perturbed versions $m(x)$ using the source model and another generic pre-trained mask-filling model m; large drops in the log probability correspond to training points. They show via a similar derivation that DetectGPT represents second-order curvature, approximating the trace of the Hessian with respect to x : $\text{Tr}(\mathbf{H}_x)$. However, it's impossible to actually perturb input token encodings x with continuous noise and have valid tokens; using a masking model to simulate perturbations in x lacks the theoretical grounding of directly perturbing θ. Nevertheless, while DetectGPT is designed to determine if x was *generated by* θ, an orthogonal concern from privacy, we can still use a threshold based on the DetectGPT statistic for membership inference, which we evaluate in Table 1. We find that it outperforms the random baseline and is comparable to loss-based attacks, but performs much worse than $\texttt{MoPe}_\theta$.

4 Attack Results

We benchmark our MIAs on the Pythia Suite [Biderman et al. (2023b)]. Pythia is trained on The Pile [Gao et al. (2021)], which after deduplication contains 207B tokens from 22 primarily academic sources. Crucially, The Pile has a clean training/validation split, as well as model checkpoints that allow us to evaluate on a model corresponding to one full pass over the dataset. We evaluate attacks using 1000 points sampled randomly from the training data and 1000 points from the test data.

For each MIA we report the AUC as an overall success metric. We also report the TPR at low FPR rates $(.05, .1)$, an approach [Carlini et al. (2022)] advocates as a better metric for MIA performance.

Model	Method	AUC	$\text{TPR}_{.1}$	$\text{TPR}_{.05}$
70M	$\texttt{LOSS}_\theta$	0.5075	0.067	0.025
70M	DetectGPT	0.5207	0.071	0.036
70M	$\textbf{MoPe}_\theta$	**0.6069**	**0.095**	**0.054**
160M	$\texttt{LOSS}_\theta$	0.5118	0.071	0.030
160M	DetectGPT	0.5155	0.076	0.034
160M	$\textbf{MoPe}_\theta$	**0.6478**	**0.126**	**0.058**
410M	$\texttt{LOSS}_\theta$	0.5135	0.070	0.028
410M	DetectGPT	0.5254	0.073	0.033
410M	$\textbf{MoPe}_\theta$	**0.5958**	**0.118**	**0.045**
1B	$\texttt{LOSS}_\theta$	0.5161	0.069	0.029
1B	DetectGPT	0.5322	0.073	0.031
1B	$\textbf{MoPe}_\theta$	**0.5924**	**0.103**	**0.037**
1.4B	$\texttt{LOSS}_\theta$	0.5168	0.067	0.029
1.4B	DetectGPT	0.5336	0.069	0.029
1.4B	$\textbf{MoPe}_\theta$	**0.5656**	**0.091**	**0.046**
2.8B	$\texttt{LOSS}_\theta$	0.5039	**0.059**	0.023
2.8B	DetectGPT	—	—	—
2.8B	$\texttt{MoPe}_\theta$	**0.5320**	0.051	0.023
6.9B	$\textbf{LOSS}_\theta$	**0.5227**	**0.070**	**0.026**
6.9B	DetectGPT	—	—	—
6.9B	$\texttt{MoPe}_\theta$	0.5217	0.069	0.020
12B	$\textbf{LOSS}_\theta$	**0.5252**	**0.068**	0.024
12B	DetectGPT	—	—	—
12B	$\texttt{MoPe}_\theta$	0.5164	0.067	0.024

Table 1: For each model size and attack, we report the AUC, TPR at FPR 0.1, and TPR at FPR $.05$.

Inspecting Table 1 we see that $\texttt{MoPe}_\theta$ achives the highest AUCs at model sizes up to 2.8B, and is tied at 6.9B, with similar relative performance to other methods at low FPRs. Interestingly, DetectGPT also outperforms the random baseline and $\texttt{LOSS}_\theta$ at all model sizes. Note that we could not run DetectGPT for models with 2.8B parameters and above due to the large computational burden of performing mask-filling perturbations. See Appendix G for combined ROC curves for each method across model sizes.

Model Size. Recent work [Carlini et al. (2023a)] on the GPT-Neo [Black et al. (2021)] models evaluated on the Pile has shown that as model size increases so does memorization. Consisent with this finding, we see that with increasing model size there is a gradual increase in the AUC achieved by the $\texttt{LOSS}_\theta$ attack. However, $\texttt{MoPe}_\theta$ and DetectGPT exhibit no such trend, with the highest $\texttt{MoPe}_\theta$ AUC values actually coming at the two smallest model sizes! While this doesn't directly contradict previous results we find it very surprising. One potential explanation could be that the attack success of $\texttt{MoPe}_\theta$ is actually inversely correlated with model size, due to variance of the Hutchinson Trace Estimator in Equation 2 increasing for larger models, and may not reflect the fact that larger Pythia suite models are actually more private. That is, we conjecture an attack that could *actually compute* $\text{Tr}(\mathbf{H}_\theta)$ might still enjoy increasing attack success with increasing model size.

$\texttt{MoPe}_\theta$ vs $\texttt{LOSS}_\theta$ comparison. The disparities in MIA performance between $\texttt{MoPe}_\theta$ and the other attacks above (particularly $\texttt{LOSS}_\theta$) imply that there *must exist* a number of training points where the statistics take very different values. We're particularly interested in $\texttt{MoPe}_\theta$ vs $\texttt{LOSS}_\theta$, since much prior work uses (some function of) $\texttt{LOSS}_\theta$ as a proxy for memorization and membership. Since $\texttt{MoPe}_\theta$ outperforms $\texttt{LOSS}_\theta$, particularly at smaller model sizes, there must be points with average loss values but outlier $\texttt{MoPe}_\theta$ values. We also investigated change in $\texttt{LOSS}_\theta$ vs $\texttt{MoPe}_\theta$ across training data order and saw $\texttt{LOSS}_\theta$ uniformly decreases whereas $\texttt{MoPe}_\theta$ does not. See Appendix F for more details.

This also raises the obvious question of if the attacks can be combined to yield stronger attacks than either attack in isolation. We find that while for smaller model sizes, we don't get a significant improvement over $\texttt{MoPe}_\theta$ in AUC by ensembling, for the two largest model sizes, we do get a significant improvement by thresholding on a weighted sum of the two statistics (after z-scoring). See Appendix E for an investigation into combining $\texttt{LOSS}_\theta$ and $\texttt{MoPe}_\theta$ attack statistics.

5 Social Impact Statement

As language models are ubiquitously deployed with few safeguards for protecting the privacy of their data, understanding their vulnerabilities is of utmost importance. In this paper, we explore the white-box setting of MIAs—an important setting due both to the prevalence of models with publicly published weights (e.g., Llama 2, BLOOM, MPT, etc.) as well as risks from weights being leaked from cybersecurity attack vectors. Our work demonstrates the concerning ability to infer the membership of training data points, and challenges previous work using just loss as evidence for memorization. MIAs can be used to check if a model has used training data without consent but we also emphasize that they can be naturally extended for actual data extraction. Given these dangers, we call on the research community to take greater action to study and remedy this privacy problem.

6 Future Directions

In this paper we develop an effective new MIA against pre-trained LLMs across a wide range of model sizes, opening up several enticing avenues for future research. We show that detection methods like DetectGPT can also function as privacy attacks, an observation that could extend to other types of detection methods and generative models (e.g. GANs, Diffusion models) which are currently under-explored. Given our MIA is successful at low FPRs, future work could use $\texttt{MoPe}_\theta$ as part of a training data extraction attack in the style of [Carlini et al. (2021); Yu et al. (2023); Carlini et al. (2023b)]. Another major unresolved question in this work is why $\texttt{MoPe}_\theta$ success actually scales *inversely* with model size, which we conjecture is actually a property of the method, namely error in the Hutchinson trace estimator, rather than the model. Finally, since our white-box $\texttt{MoPe}_\theta$ attack outperforms black-box attacks like $\texttt{LOSS}_\theta$ and DetectGPT, it raises the question of whether other white-box attacks could improve attack success even further. For example, attacks based on the parameter gradient have been shown to outperform loss-based attacks in other settings, and would be a natural candidate to try against LLMs [Nasr et al. (2019)].

References

Rohan Anil, Badih Ghazi, Vineet Gupta, Ravi Kumar, and Pasin Manurangsi. 2021a. Large-scale differentially private bert.

Rohan Anil, Badih Ghazi, Vineet Gupta, Ravi Kumar, and Pasin Manurangsi. 2021b. Large-scale differentially private BERT. *CoRR*, abs/2108.01624.

Stella Biderman, USVSN Sai Prashanth, Lintang Sutawika, Hailey Schoelkopf, Quentin Anthony, Shivanshu Purohit, and Edward Raf. 2023a. Emergent and predictable memorization in large language models.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023b. Pythia: A suite for analyzing large language models across training and scaling. *arXiv preprint arXiv:2304.01373*.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large scale autoregressive language modeling with meshtensorflow.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, page 1877–1901. Curran Associates, Inc.

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

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2023a. Quantifying memorization across neural language models.

Nicholas Carlini, Utkarsh Kandpal, Jacob Lehman, Nicolas Papernot, and Florian Tramèr. 2023b. Lm-extraction: A benchmark for training data extraction from language models. `https://github.com/google-research/lm-extraction-benchmark`.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *USENIX Security Symposium*, volume 6.

Christophe Dupuy, Radhika Arava, Rahul Gupta, and Anna Rumshisky. 2022. An efficient dp-sgd mechanism for large scale nlp models.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. 2016. Calibrating noise to sensitivity in private data analysis. *J. Priv. Confidentiality*, 7(3):17–51.

Giorgio Franceschelli and Mirco Musolesi. 2021. Copyright in generative deep learning.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2021. The pile: An 800gb dataset of diverse text for language modeling. *CoRR*, abs/2101.00027.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. *CoRR*, abs/2009.11462.

Michael F. Hutchinson. 1989. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics - Simulation and Computation*, 18:1059–1076.

Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. 2022. Knowledge unlearning for mitigating privacy risks in language models.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2022. Survey of hallucination in natural language generation. *CoRR*, abs/2202.03629.

Konrad Königsberger. 2000. *Analysis 2*. Springer Berlin Heidelberg.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2021. Deduplicating training data makes language models better. *CoRR*, abs/2107.06499.

Xuechen Li, Florian Tramèr, Percy Liang, and Tatsunori Hashimoto. 2022. Large language models can be strong differentially private learners.

Jimit Majmudar, Christophe Dupuy, Charith Peris, Sami Smaili, Rahul Gupta, and Richard Zemel. 2022. Differentially private decoding in large language models.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature.

Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. 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.

Alec Radford, Jeff Wu, Rewon Child, D. Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.

Alexandre Sablayrolles, Matthijs Douze, Yann Ollivier, Cordelia Schmid, and Hervé Jégou. 2019. White-box vs black-box: Bayes optimal strategies for membership inference.

Brihat Sharma, Yanjun Gao, Timothy Miller, Matthew M Churpek, Majid Afshar, and Dmitriy Dligach. 2023. Multi-task training with in-domain language models for diagnostic reasoning. *arXiv preprint arXiv:2306.02077*.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models.

Rebecca Soper. 2023. Can ai help you build relationships? amorai thinks so. `https://techcrunch.com/2023/05/13/ai-relationship-building-amorai/`.

Nikhil Vyas, Sham Kakade, and Boaz Barak. 2023. Provable copyright protection for generative models.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*.

Jiayuan Ye, Aadyaa Maddi, Sasi Kumar Murakonda, and Reza Shokri. 2021. Enhanced membership inference attacks against machine learning models. *CoRR*, abs/2111.09679.

Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting.

Weichen Yu, Tianyu Pang, Qian Liu, Chao Du, Bingyi Kang, Yan Huang, Min Lin, and Shuicheng Yan. 2023. Bag of tricks for training data extraction from language models.

A Additional Related Work on LLM Privacy.

[Carlini et al. (2021); Yu et al. (2023)] focus on the problem of *training data extraction* from LLMs by generating samples from the model, using loss-based MIAs to determine if the generated point is actually a member of the training set that has been memorized. Both papers focus more on extraction than explicitly evaluating membership inference attack success, and acknowledge existing MIAs against LLMs are relatively weak. [Jang et al. (2022)] studies the problem of *unlearning* a training point from a trained model via taking gradient ascent steps. One metric they use to determine if a point has been unlearned is if the loss on a point x that has been unlearned is close to the expected loss for a test point. Our work has implications for this kind of definition of unlearning, as our results show that an average $\texttt{LOSS}_\theta$ value does not mean the point cannot be easily detected as a training point. Differentially private training Dwork et al. (2016) is a canonical defense against MIAs, and there has been a flurry of recent work on private model training in NLP [Anil et al. (2021a); Majmudar et al. (2022); Dupuy et al. (2022)]. While [Li et al. (2022)] report success in *fine-tuning* language models with differential privacy, it is know that privacy during pre-training comes at a great cost to accuracy [Anil et al. (2021b)]. Since pre-training with differential privacy remains a challenge, existing work provides theoretical mitigation guarantees against our attacks on pre-trained models.

B Pythia Suite.

We identified EleutherAI's *Pythia* [Biderman et al. (2023b)] suite of models as the prime candidate for studying membership inference attacks. Models in the Pythia suite are trained on the Pile [Gao et al. (2021)] dataset, which is an 825GB dataset of about 300B tokens, consisting of 22 primarily academic sources. All our experiments are using models trained on a version of the Pile that was de-duplicated using MinHashLSH with a threshold of 0.87, which reduces the size to 207B tokens. We perform our experiments in the de-duplicated regime as it has been shown that the presence of duplicated data greatly increases the likelihood of training data memorization [Lee et al. (2021)], and so attacks in the de-duplicated setting are significantly more compelling. We use a model checkpoint corresponding to one full pass over the de-duplicated Pile. The data is tokenized using a BPE tokenizer developed specifically on the Pile. Training examples are 2048 tokens, and the batch size used during training is 1024. In order to maintain an apples-to-apples comparison between train and test examples, we batch test examples identically when evaluating our MIA attacks. Importantly, the Pile contains train vs. test splits which allow us to evaluate our MIA attacks, and is also annotated with the order of points during the training of all models allowing us to study the implications of training order for privacy.

The models in the Pythia suite are open source and available through Hugging Face, have publicly available model checkpoints saved during training, and range in size from 70m parameters to 12B. The models follow the transformer-based architecture in [Brown et al. (2020)], with some small modifications [Biderman et al. (2023b)].

C Compute Usage

We performed all experiments on AWS `p3.2xlarge` and `g5.12xlarge` instances. Evaluating $\texttt{MoPe}_\theta$ on the smallest size can be done in a Colab notebook; the largest size takes approximately 200 GPU hours on A10G GPUs (48 hours on 4 cores). In total, we used approximately 1000 GPU hours to perform the experiments in this paper.

D Hyperparameter Tuning

For $\texttt{MoPe}_\theta$ we use $n = 20$ perturbed models; since we are approximating an expectation, we expect our attack to be more accurate if we consider more models, but we also need to balance computational considerations. We found the best noise level σ for the models of sizes up to 2.8B parameters, by conducting a grid search over $\sigma = [0.001, 0.005, 0.01, 0.05]$. The highest AUC was achieved at 0.01 for the 1B parameter model, 0.001 for the 2.8B model and 0.005 for the remaining models. This suggests that there was no relationship between the optimal value of σ and the model size. For the 6.9B and 12B parameter models, we chose a noise level of 0.005. We record the AUCs achieved at different noise levels in Table 2 in the Appendix.

Model	$\sigma = .001$	$\sigma = .005$	$\sigma = .01$	$\sigma = .05$
70M	0.6034	0.6069	0.5708	0.4906
160M	0.6394	0.6478	0.5613	0.5121
410M	0.5915	0.5958	0.5367	0.5190
1B	0.5028	0.5142	0.5924	0.5111
1.4B	0.5652	0.5656	0.5502	0.5136
2.8B	0.5320	0.5086	0.5109	0.5030

Table 2: $\texttt{MoPe}_\theta$ AUC per model size and noise level σ.

E Ensemble Attacks and LOSS/MoPe Scatters

Below, we present ROC curves for the combined MoPE/LOSS attack at the largest model sizes, where ensembling has substantial improvements. Additionally, we show scatter plots of $\texttt{MoPe}_\theta$ and $\texttt{LOSS}_\theta$ values in our tests with little correlation, indicating that $\texttt{MoPe}_\theta$ must identify points that $\texttt{LOSS}_\theta$ does not.

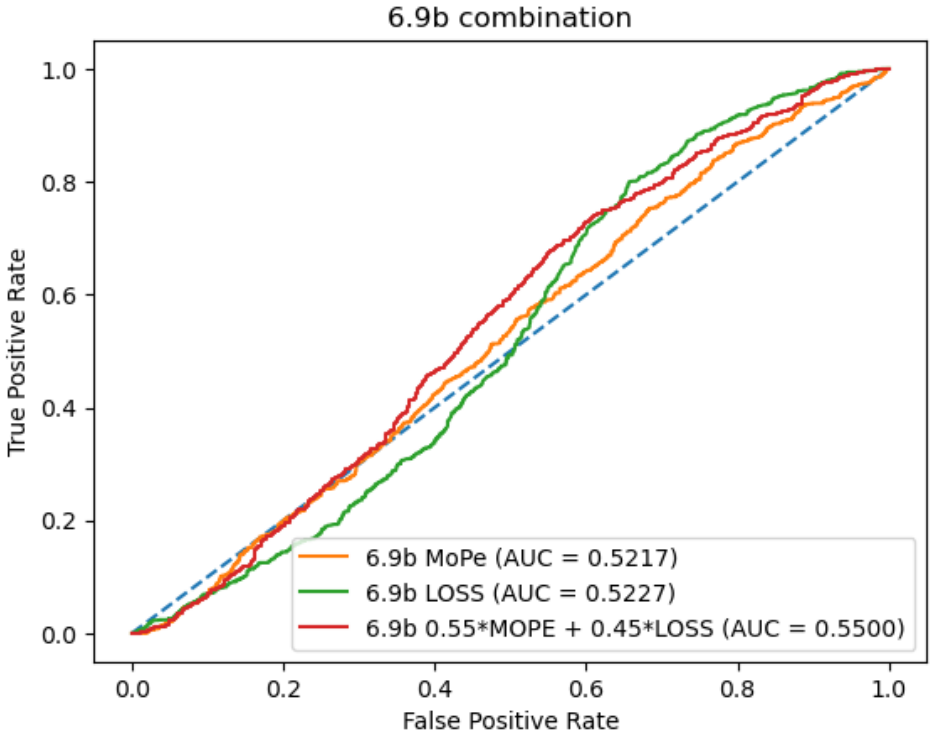


Figure 2: $\texttt{LOSS}_\theta$ and $\texttt{MoPe}_\theta$ Ensemble Attack

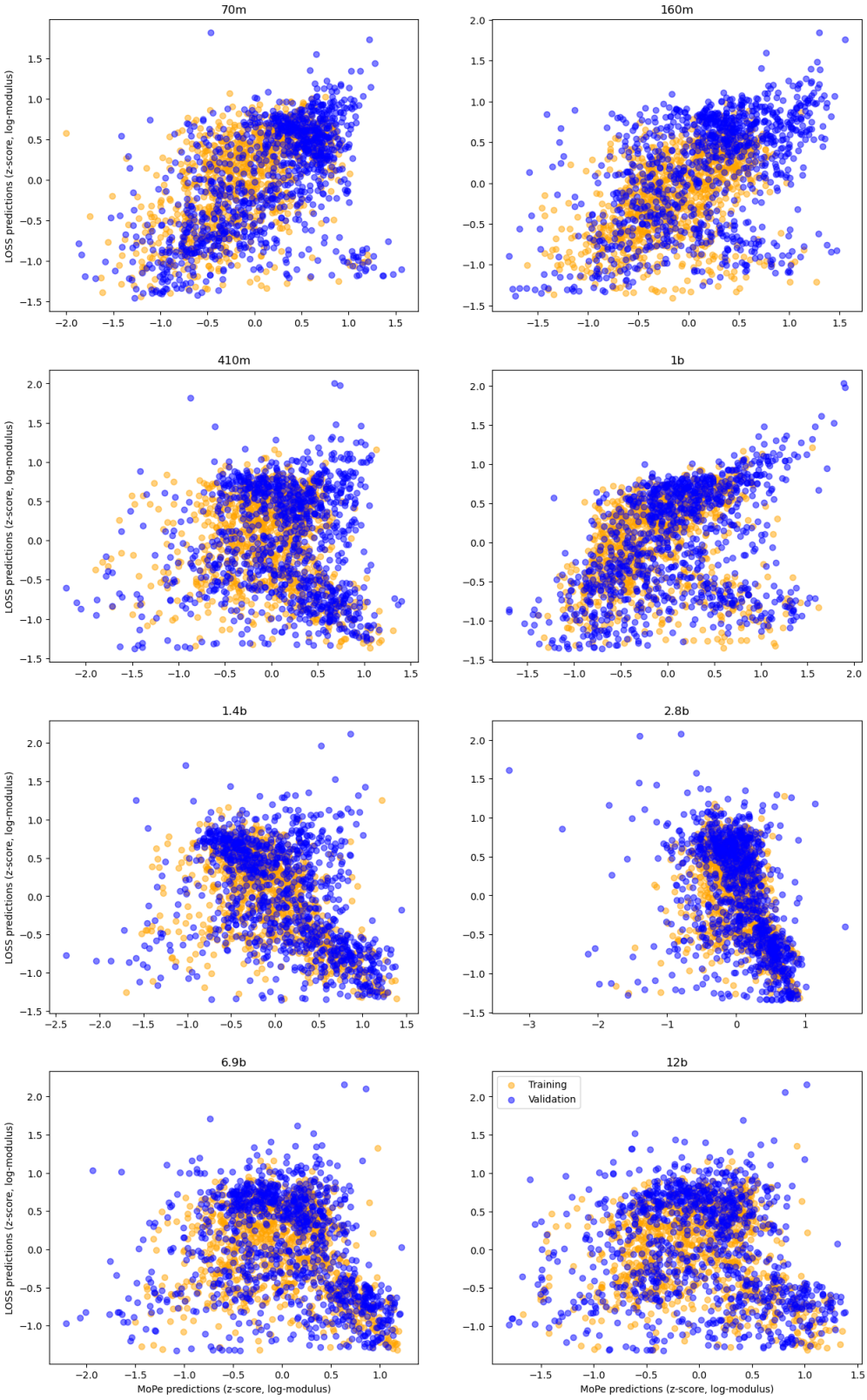


Figure 3: $\texttt{LOSS}_\theta$ vs. $\texttt{MoPe}_\theta$ scatter plots across model sizes.

F Effect of Training Order on MIA Statistics

Below, we investigate how $\texttt{LOSS}_\theta$ and $\texttt{MoPe}_\theta$ statistics vary on average depending on when they are processed during training. Concretely, we sample 2000 points from 10 paired batches $\{0-1, 9999-1\text{e}4, 19999-2\text{e}4, \ldots 89999-9\text{e}4, 97999-9.8\text{e}4\}$, which approximately correspond to the first and last data batches that the Pythia models see during pre-training. For each set of 2000 points, we compute the average $\texttt{LOSS}_\theta$ and $\texttt{MoPe}_\theta$ values with respect to the θ reached at the end of the first epoch. Consistent with previous findings, loss declines for more recent batches; in contrast, there is no such observable pattern at any fixed model size for the $\texttt{MoPe}_\theta$ statistic! This finding is consistent with recent work [Biderman et al. (2023a,b)] that studies memorization in the Pythia suite and find no correlation between order in training and if a point is "memorized" (as defined in terms of extractability).

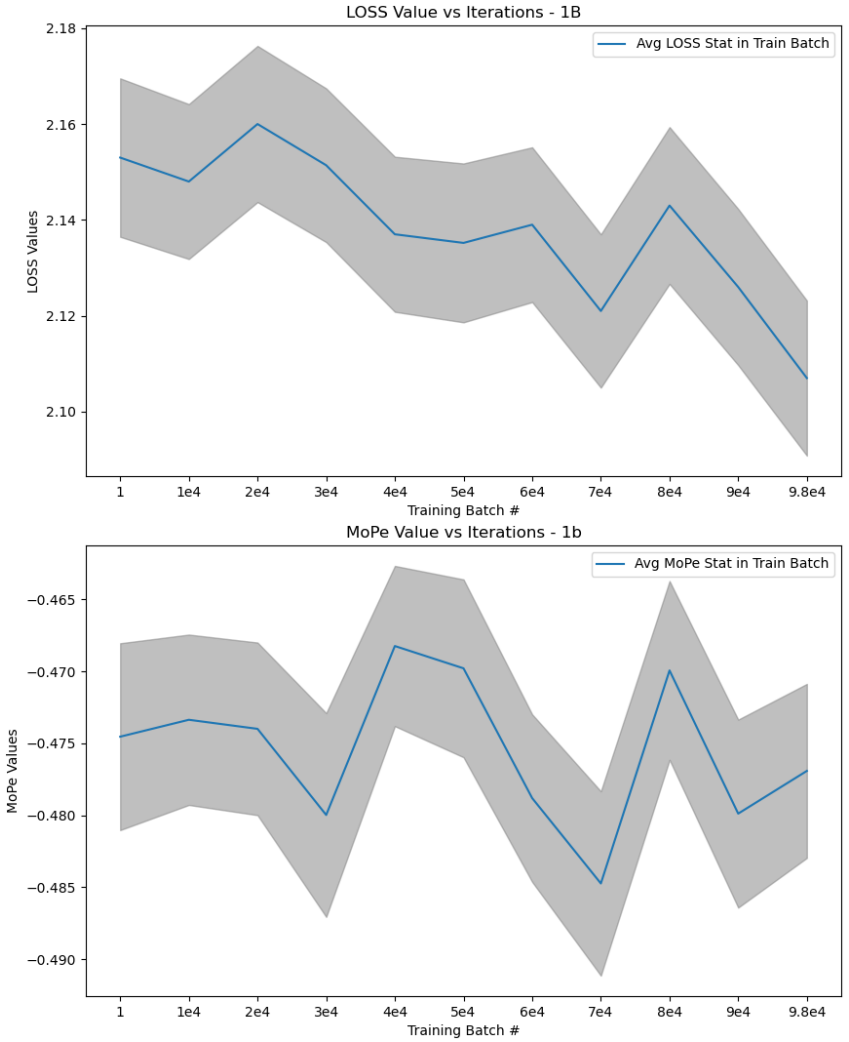
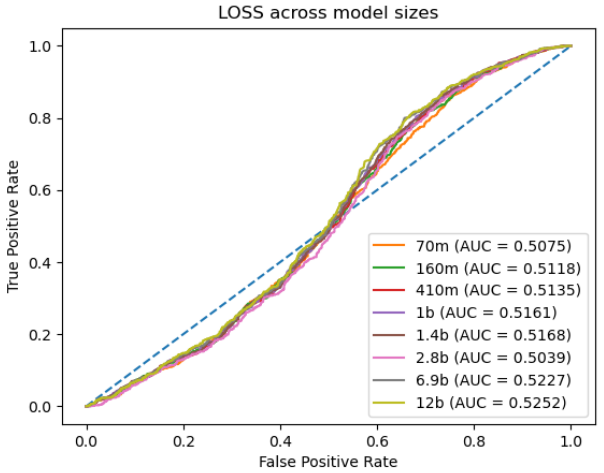
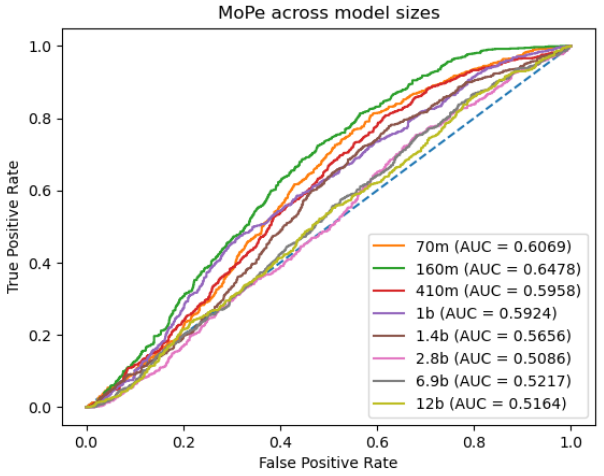


Figure 4: Average $\texttt{LOSS}_\theta$ and $\texttt{MoPe}_\theta$ scores per training batch with 95% CI for 1B model.

G Combined MIA ROC Curves

Below, we show combined ROC curves for $\texttt{LOSS}_\theta$, $\texttt{MoPe}_\theta$, and $\texttt{DetectGPT}$ membership inference attacks across model sizes.





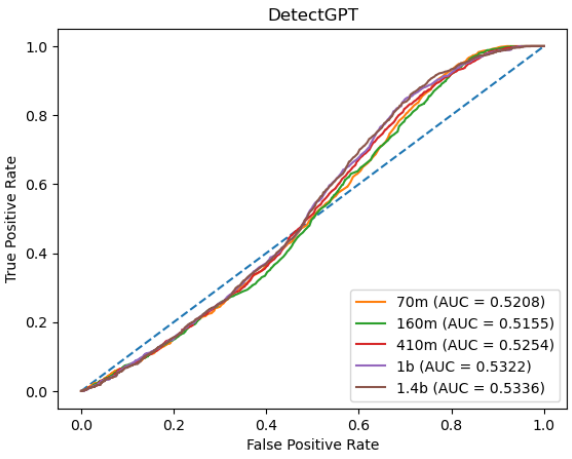


Figure 5: $\texttt{LOSS}_\theta$, $\texttt{MoPe}_\theta$, $\texttt{DetectGPT}$ ROC curves.