An Empirical Analysis of Memorization in Fine-tuned Autoregressive Language Models

Fatemehsadat Mireshghallah[1*], **Archit Uniyal**[2], **Tianhao Wang**[2], **David Evans**[2], **Taylor Berg-Kirkpatrick**[1]
[1] University of California San Diego, [2] University of Virginia
[fatemeh, tberg]@ucsd.edu,
[a.uniyal,tianhao,evans]@virginia.edu

Abstract

Several recent works have shown that large language models present privacy risks through memorization of training data. Little attention, however, has been given to the fine-tuning phase and it is not well understood how memorization risk varies across different fine-tuning methods (such as fine-tuning the full model, the model head, and adapter). This presents increasing concern as the "pre-train and fine-tune" paradigm proliferates. We empirically study memorization of fine-tuning methods using membership inference and extraction attacks, and show that their susceptibility to attacks is very different. We observe that fine-tuning the head of the model has the highest susceptibility to attacks, whereas fine-tuning smaller adapters appears to be less vulnerable to known extraction attacks.

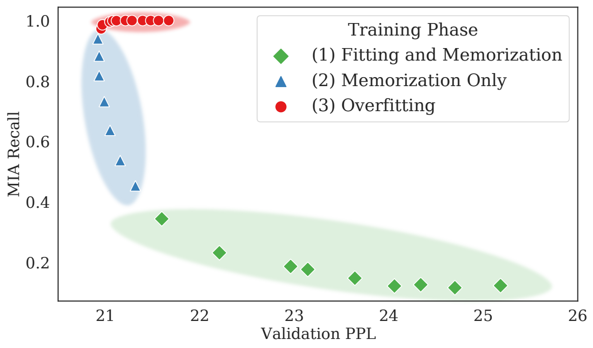


Figure 1: Each point in the graph shows the given metric values at the end of each training epoch. The rightmost lower points show the beginning, and as we move to left and upwards training progresses. We identify three separate phases within the learning process, distinguished by their memorization and generalization trends.

1 Introduction

Transformer-based language models have become the models of choice for many NLP tasks, such as email, text and code auto-completion, question answering and sentiment analysis (Chen et al., 2021, 2019). These models are commonly trained using the *pre-train and fine-tune paradigm*, where they are first trained (*pre-trained*) on a large, general domain dataset (in the order of hundreds of Gigabytes), and then *fine-tuned* on smaller, task-specific datasets to adapt the model to a specific domain (Ramponi and Plank, 2020; Li and Liang, 2021; Houlsby et al., 2019).

Several works have demonstrated that such large models have a high capacity for memorizing training samples during pre-training and are therefore highly susceptible to membership inference and data extraction attacks (Zanella-Béguelin et al., 2020; Carlini et al., 2021b; Nakamura et al., 2021). More specifically, Carlini et al. (2021b) and Mireshghallah et al. (2022) have mounted such attacks on pre-trained language models and shown the severity of this issue by extracting complete training sequences and inferring membership of a large fraction of the training samples.

These works focused on memorization during pre-training, but scant attention has been given to fine-tuning. In this work, we focus on different fine-tuning methods and their propensity for memorization of training samples. Fine-tuning data is actually of higher concern than pre-training data, since most pre-training datasets are large public corpora (Raffel et al., 2019; Dodge et al., 2021) with limited privacy concerns (Brown et al., 2022), while fine-tuning sets are small, targeted, and potentially very private (Basu et al., 2021; Li et al., 2021). Further, pre-training generally happens only a few times (as it needs resources that are usually only available to large companies (Brown et al., 2020)) while fine-tuning is increasingly the dominant way that end-users fit models.

Given the size of these large language models, fine-tuning all the model parameters can be compute and memory-intensive (Lewis et al., 2019; Brown et al., 2020; Fedus et al., 2021). As a result, recent works have proposed new parameter efficient fine-tuning methods that update only a

[*] Corresponding author email: fatemeh@ucsd.edu

subset of the model's parameters (Houlsby et al., 2019; Li and Liang, 2021; He et al., 2022). In this paper, we focus on studying memorization of three popular fine-tuning methods: (1) fine-tuning all model parameters (2) fine-tuning the head, which is commonly used by practitioners and involves updating only the last layer of the model which produces the logits, and (3) fine-tuning adapters (Houlsby et al., 2019), which are small bottleneck modules inserted within transformer blocks. For measuring memorization, we use two proxy metrics: (a) recall of a reference-based membership inference attack (MIA) (Mireshghallah et al., 2022) and (b) exposure (Carlini et al., 2019), which measures how susceptible the model is to a sample extraction attack which tries to reconstruct samples from training data. We run our experiments on the Wikipedia (Merity et al., 2016), Penn Treebank (Marcus et al., 1993) and Enron Emails (Klimt and Yang, 2004) datasets, for the task of autoregressive language modeling. We selected Wikipedia and Penn Treebank as they are most commonly used for fine-tuning, and Enron since it is a dataset of emails representing private tuning data.

Figure 1 shows how we conceptually identify three distinct phases in the fine-tuning process, based on validation perplexity (generalization) and membership inference attack recall (memorization). Each point shows these metrics at the end of a training epoch. For all fine-tuning methods, we observe that in a *memorization only* phase, the model memorizes more and more, without overfitting or generalizing better (Figure 2). In terms of different fine-tuning methods, we find that the common practice of fine-tuning only the head of a model has the highest memorization (by a large margin) for the same level of perplexity, among different fine-tuning methods – even full fine-tuning, which updates more parameters. This result is surprising and potentially indicates that only tuning parameters higher in the model architecture (closer to the output) exacerbates the memorization and increases the leakage based on our metrics. We also show that fine-tuning the full model and small adapters are on the Pareto-frontier in terms of the attack recall vs. validation perplexity graph. Code and instructions to reproduce our results are available at `https://github.com/mireshghallah/ft-memorization/`.

2 Model Fine-tuning

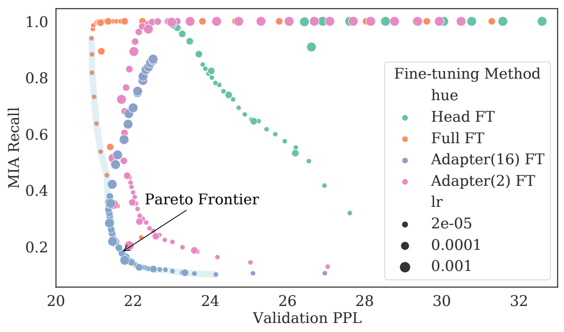
We focus on two main fine tuning methods, for fine-tuning GPT-2 with next word prediction objective: (1) fine-tuning the model head, i.e., the prediction layer, as it is the most common method used in practice, and (2) fine-tuning adapters (Houlsby et al., 2019). Adapters are small rank-restricted modules that are inserted inside transformer blocks, as added parameters and are fine-tuned for different tasks or datasets. The shape and size of the adapter module is controlled by the *reduction factor*, which determines the ratio of the size of the bottleneck to its input. During adapter tuning, the rest of the model remains frozen, therefore the number of trainable parameters is low (around 1% of the full model parameters). In our experiments, we choose reduction factors of 16 and 2, for adapters, as the former is the default used by (Pfeiffer et al., 2020; Houlsby et al., 2019), and the latter is the largest factor.

3 Measuring Memorization

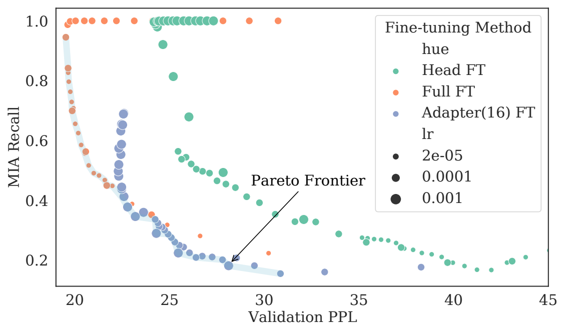
To measure memorization, we use two metrics: membership inference attack recall and exposure.

Membership Inference (MIA Recall). We use the percentage of training samples that are correctly classified as training members (out of a pool of training and validation samples) by the reference-based attack proposed in Mireshghallah et al. (2022) and Carlini et al. (2021a) as a proxy metric of memorization. For each sample x whose membership in the training set we want to determine, we feed it to the fine-tuned model, M, and get its likelihood, $\mathbf{Pr}^M(x)$. We also feed it to a reference model, R, a pre-trained model that is not fine-tuned, and get the probability $\mathbf{Pr}^R(x)$. We then use $LR(x) = \frac{\mathbf{Pr}^R(x)}{\mathbf{Pr}^M(x)}$, the likelihood ratio, to determine if x is a training sample. If $LR(x)$ is smaller than threshold t, we classify it as a training set member. Otherwise, we classify it as a non-member. We determine the threshold t by calculating $LR(s)$ for all s in the validation set, and then choose the threshold to be the highest threshold such that the false positive rate (over training and validation members) would not exceed 10%. The higher the recall of this attack is, the higher the leakage of the model.

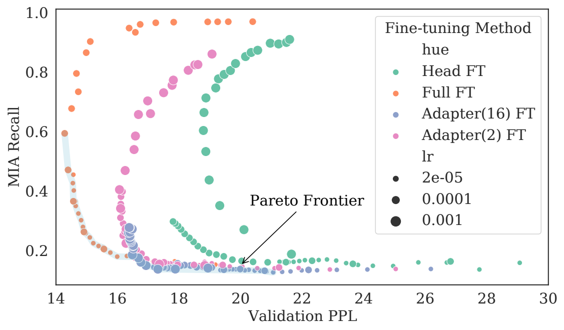
Exposure. As a second measure of memorization, we use the exposure metric from Carlini et al. (2019) which inserts a secret (canary) of a certain format into the training data and calculates its vulnerability to extraction. Exposure is defined as the negative log-rank of the inserted secret in terms of model probability, among all other possible sequences of the same length. This quantity is then added to a constant to ensure the exposure is always positive.



(a) Wikipedia Dataset



(b) Penn Treebank Dataset



(c) Enron Dataset

Figure 2: Pareto frontier for utility (validation PPL) Vs. privacy (MIA recall). Each dot shows different checkpoints, and the colors show different fine-tuning methods. We desire models that have low PPL and low attack recall.

The lower the exposure is, the harder it is to extract the secret. In our experiments, we insert 50 copies of the phrase "the secret number is 940955" into the training data to accentuate the differences between the fine-tuning methods. For a six-digit secret, an exposure of around $\log_2(10^6) \approx 20$ means the canary can be reliably extracted from the model.

4 Experimental Setup

Datasets. (1) Huggingface's Wikipedia wikitext-2-raw-v1 dataset, consisting of 36718 training samples (2) Huggingface's Penn Treebank ptb_text_only, consisting of 42068 training samples and (3) a sub-sampled version of Enron email dataset consisting of 7180 emails. We use a sequence length of 1024, training batch size of 8, and fine-tune for 20 epochs.

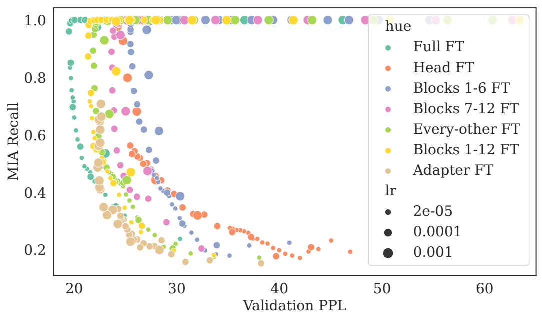


Figure 3: Ablating how the location and number of trainable parameters effects memorization on the Penn Treebank dataset. Each dot shows different checkpoints, and the colors show different fine-tuning methods. We desire models that have low PPL and low attack recall.

Models. We study memorization in fine-tuning Huggingface's pre-trained GPT-2 on the datasets mentioned above. We use a pre-trained but not fine-tuned GPT-2 as the reference model for our membership inference attack. We use the adapter hub's implementation of the Pfeiffer architecture, with reduction factors 2 and 16 (Pfeiffer et al., 2020).

Metrics. We use *Validation Perplexity* as a metric for the performance of the model, where lower perplexity is better. We evaluate memorization at each epoch using the *MIA recall* and *exposure* metrics described in Section 3. The experiments in the paper are all repeated 3 times and we report the average values for each metric.

Hyperparameters and result presentation. We run optimization for each fine-tuning method for 20 epochs, and perform evaluation of the mentioned metrics at the end of each epoch. We experiment with the three learning rates 2×10^{-5},10^{-4},10^{-3}, and present the results for all of them. Therefore, each graph would have an overall of 20×3 points, for each fine-tuning method, unless the point is outside the plot range. For the reported exposure numbers, we selected points close to the pareto frontier to present in Table 1, to summarize results.

5 Results

In this section we discuss our experimental results comparing the privacy-utility trends for different fine-tuning methods. We refer to the naming convention shown in Figure 1 and provide extended graphs for each experiment in Appendix A.3. We also present additional experiments where we train the model from scratch (instead of fine-tuning

Table 1: Exposure metric. Higher exposure indicates more leakage, and exposure above 20 means the secrets (canaries) are reliably extractable. The perplexity numbers here are different from the ones in other experiments since the training data is diluted with the artificially inserted secrets.

		Full FT	Head FT	Adapters (2)	Adapters (16)
Parameters (Millions)		124.440	38.590	7.092	0.895
Wiki	Val PPL	**24.82**	28.76	24.41	25.26
	Exposure	1.42	10.78	14.54	**0.83**
PTB	Val PPL	29.55	31.24	29.79	**29.41**
	Exposure	7.03	12.0	12.40	**4.54**
Enron	Val PPL	**12.52**	13.51	13.03	12.81
	Exposure	1.32	10.77	2.02	**0.440**

pre-trained models), fine-tune different model architectures, and study the generalization gap in Appendix A.1.

5.1 Memorization of Fine-tuning Methods

Figures 2a, 2b, 2c compare the fine-tuning methods in terms of privacy leakage, measured by MIA recall and Table 1 shows the exposure results for the three datasets, along with their parameter counts. The blue lines show the Pareto frontier, marking the desirable trade-off points, with low recall and PPL.

5.1.1 Shared Trends

The "memorization only" phase in training, where validation perplexity (generalization) is stable and the model has not yet overfit, is also observed by Tänzer et al. (2022) in pre-trained BERT-based classifiers. However, it is named the "settling phase" there, and it is suggested that as *validation perplexity* is rather stable, early stopping is not important and training can stop at any point before overfitting. We, however, show that *memorization* is actually increasing during that phase. Therefore, if we are optimizing for privacy as well, it is best to stop training earlier. Appendix A.1.2 shows generalization gap vs. validation perplexity graphs demonstrating that the gap remains stable during the "memorization only" phase. For all the methods, across all datasets, in the "fitting+memorization" and the "memorization only" phases, we see an increase in memorization, without any overfitting. This shows that we can have high memorization/learning, and still not overfit. This is also observed for training large language models from scratch in Tirumala et al. (2022), which focuses on analyzing the effect that text type (e.g., part of speech, numbers), data size and model size have on memorization when training from scratch.

5.1.2 Comparison of Fine-tuning Methods

Results for both the MIA recall and exposure metrics (Figure 2 and Table 1) are consistent, showing higher leakage for head fine-tuning and lower for full model fine-tuning and adapters. The first observation here is that head fine-tuning is an outlier, with extremely high leakage, on all three datasets. We can also see that the validation perplexity achieved by this method is consistently lower than the other methods. We hypothesize that the high leakage of fine-tuning the head is due to both the high number of parameters (38 million) and the location of the parameters, right at the last layer of the model where the next word prediction happens. While full fine-tuning actually touches more parameters than head fine-tuning, it leads to less leakage under the attacks we investigate. This result is somewhat surprising and potentially indicates that tuning parameters lower in the model architecture mitigates some of the explicit memorization performed by the head. We further study this phenomenon and ablate it in Section 5.2.

We also observe that for a low-perplexity regime (without considering the cost), full fine-tuning is the best choice as it offers utility superior to adapters. However, if we have tolerance for higher perplexity, to get lower leakage, opting for adapters with a reduction factor of 16 appears better as it has lower MIA recall and a lower propensity for overfitting, compared to the other methods. One final observation is that full-finetuning has the shortest "fitting+memorization" phase, whereas head fine-tuning has the longest.

5.2 Parameter Count, Location and Tying

To further test our hypothesis that the privacy-utility trade-off has to do with *both* trainable parameter count and location/distribution within the model architecture (Section 5.1.2), we run experiments with the following set of trainable parameters: (1) first half: blocks 1–6 of the 12 transformer blocks of the GPT2 model (42M trainable params), (2) second half: blocks 7–12 (42M), (3) every other block (42M) and (4) entire body: all the 12 blocks (84M). In all these scenarios we freeze the head and fine-tune only the blocks. As shown in Figure 3, we find that Full FT $>$ Adapters $>$ all 12 blocks=every other block $>$ blocks 7 to 12 $>$ blocks 1 to 6 $>$ Head FT, in terms of privacy-utility trade-off desirability. Based on this, we argue that how the trainable parameters are scattered in the network affects how well the model makes progress in the first phase (the training and fitting phase), which affects the validation perplexity when it enters the second phase (memorization-only phase). As Figure 2 also shows, full fine-tuning and adapter tuning make faster progress and end up in a lower perplexity.

Figure 4 shows an ablation study of how

Table 2: Comparison of fine-tuning different transformer blocks on the Wikipedia dataset.

	Block 1	Block 5	Block 8	Block 12	Full FT	Head FT	Adapters (2)	Adapters (16)
Validation PPL	24.39	23.35	23.36	24.05	23.05	23.93	23.62	**21.75**
MIA Recall	22.2	22.6	20.8	21.3	19.2	81.6	16.8	**15.2**
#Params (in Millions)	7.088	7.088	7.088	7.088	124.440	38.590	7.092	0.895

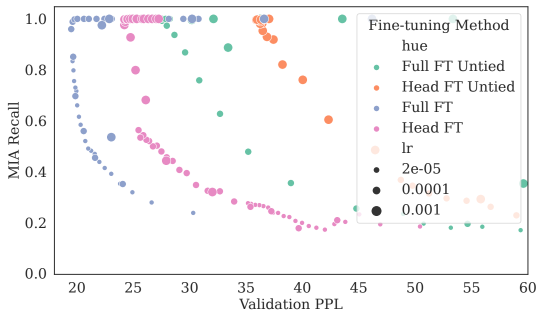


Figure 4: Ablating how the untying of the trainable parameters effects memorization on the Penn Treebank dataset. Each dot shows different checkpoints, and the colors show different fine-tuning methods. We desire models that have low PPL and low attack recall.

untying model parameters affects the privacy-utility trade-off. By untying parameters, we mean creating a separate set of parameters for the head of the model and the input embeddings, as by default these two parameter sets are tied in GPT2, meaning the same set of 38.59 Million parameters are used for both these components. However, in the untied scenario, we first duplicate them, and then create separate trainable parameters, adding an extra set of 38.59 Million trainable parameters to the model. As the figure shows, tying the parameters improves the progress in training and puts the model at an advantage, compared to untying them, creating a better overall privacy-utility trade-off.

5.3 Fine-tuning Single Transformer Blocks

To have a full analysis of fine-tuning leakage, we also look at fine-tuning individual adapter blocks and freezing the rest of the model. The GPT-2 model has 12 blocks, and we experiment with fine-tuning the first, 5th, 8th, and 12th block, to cover different positions within the model. Table 2 shows the results for this experiment. We have selected the numbers such that the validation PPLs are as similar as possible. There does not seem to be any significant difference between fine-tuning different blocks, as they all manifest similar attack recalls. Block 8's recall, however, is lower than other blocks, with lower PPL, which would make it the most desirable block for fine-tuning in terms of the PPL-leakage trade-off. With respect to privacy-utility tradeoffs, fine-tuning full blocks seems less desirable than using adapters or fine-tuning the entire model.

6 Conclusion

When fine-tuning is done using sensitive training data, it is important to not just consider the cost and utility of fine-tuning methods but to also be aware that they may have different risks in terms of privacy. Our experiments show that the common practice of fine-tuning only the head of a model has the highest memorization (by a large margin). Full model fine-tuning and adapter tuning, however, are both on the Pareto-frontier in terms of attack recall vs. validation perplexity, suggesting that they are more suitable when privacy is a concern.

Acknowledgements

This project is funded in part by the NSF under grant 2200333. The authors would like to thank the anonymous reviewers and meta-reviewers for their helpful feedback. We also thank Nikolai Vogler, Nikita Srivatsan, and Kazem Taram for insightful discussions. Additionally, we thank our colleagues at the UCSD Berg Lab and UVA Security Research Group for their helpful comments and feedback.

Limitations and Ethics Statement

In our study we focus on autoregressive language models – specifically GPT-2, as it has been shown to be more prone to memorizing samples than pre-trained masked language models (MLM) (Carlini et al., 2021c; Lehman et al., 2021) Also, in this paper we loosely refer to the recall of the membership inference attack on the training set as memoirzation. However, we need to keep in mind that a low attack recall does not necessarily mean low memorization, and there might be stronger attacks (of other types, such as reconstruction) that can better uncover memorization in language models.

In this work we have used publicly available datasets and have not collected any sensitive/private data. The ultimate goal of our study is to contribute to analyzing memorization under different fine-tuning paradigms, thereby advancing our intuition of how we can better deploy private, fair and safe language models.

References

Priyam Basu, Tiasa Singha Roy, Rakshit Naidu, Zumrut Muftuoglu, Sahib Singh, and Fatemehsadat Mireshghallah. 2021. Benchmarking differential privacy and federated learning for Bert models. *arXiv preprint arXiv:2106.13973*.

Hannah Brown, Katherine Lee, Fatemehsadat Mireshghallah, Reza Shokri, and Florian Tramèr. 2022. What does it mean for a language model to preserve privacy? *arXiv preprint arXiv:2202.05520*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. 2021a. Membership inference attacks from first principles. *arXiv preprint arXiv:2112.03570*.

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

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

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. 2021c. Extracting training data from large language models.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Mia Xu Chen, Benjamin N Lee, Gagan Bansal, Yuan Cao, Shuyuan Zhang, Justin Lu, Jackie Tsay, Yinan Wang, Andrew M Dai, Zhifeng Chen, et al. 2019. Gmail smart compose: Real-time assisted writing. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, and Matt Gardner. 2021. Documenting the english colossal clean crawled corpus. *ArXiv*, abs/2104.08758.

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Bryan Klimt and Yiming Yang. 2004. The Enron corpus: A new dataset for email classification research. In *European conference on machine learning*, pages 217–226. Springer.

Eric Lehman, Sarthak Jain, Karl Pichotta, Yoav Goldberg, and Byron Wallace. 2021. Does BERT pretrained on clinical notes reveal sensitive data? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 946–959, Online. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. 2021. Large language models can be strong differentially private learners. *arXiv preprint arXiv:2110.05679*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models.

Fatemehsadat Mireshghallah, Kartik Goyal, Archit Uniyal, Taylor Berg-Kirkpatrick, and Reza Shokri. 2022. Quantifying privacy risks of masked language models using membership inference attacks.

Yuta Nakamura, Shouhei Hanaoka, Yukihiro Nomura, Naoto Hayashi, Osamu Abe, Shuntaro Yada, Shoko Wakamiya, and Eiji Aramaki. 2021. KART: Parameterization of privacy leakage scenarios from pre-trained language models.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020. Adapterhub: A framework for adapting transformers. In *Conference on Empirical Methods in Natural Language Processing (Systems Demonstrations)*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Alan Ramponi and Barbara Plank. 2020. Neural unsupervised domain adaptation in NLP—a survey. *arXiv preprint arXiv:2006.00632*.

Michael Tänzer, Sebastian Ruder, and Marek Rei. 2022. Memorisation versus generalisation in pre-trained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7564–7578, Dublin, Ireland. Association for Computational Linguistics.

Kushal Tirumala, Aram H Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. 2022. Memorization without overfitting: Analyzing the training dynamics of large language models. *arXiv preprint arXiv:2205.10770*.

Santiago Zanella-Béguelin, Lukas Wutschitz, Shruti Tople, Victor Rühle, Andrew Paverd, Olga Ohrimenko, Boris Köpf, and Marc Brockschmidt. 2020. Analyzing information leakage of updates to natural language models. In *ACM SIGSAC Conference on Computer and Communications Security*.

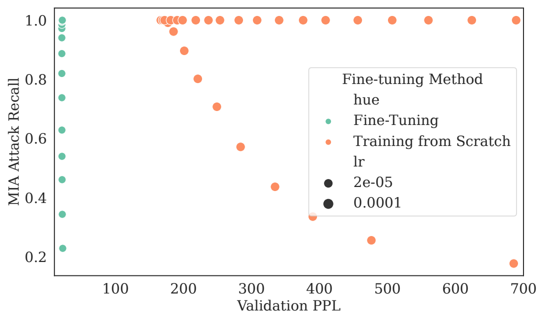


Figure 5: Ablating how training the model from scratch affects the privacy-utility trade-off, compared to fine-tuning a pre-trained model, on the Wikipedia dataset. Each dot shows different checkpoints, and the colors show different fine-tuning methods. We desire models that have low PPL and low attack recall.

A Appendix

A.1 Additional Experiments

A.1.1 Correlation between Generalization and Memorization

Figure 8 shows the correlation between the generalization gap and membership inference attack recall. The generalization gap refers to the subtraction of training perplexity from validation perplexity, and a larger gap means more overfitting. We can see that there is a direct relation between the generalization gap and attack recall, for all fine-tuning methods. We can also see that for Penn Treebank and Enron, head fine-tuning has a consistently higher generalization gap, which could explain why the membership inference attack is more successful on it.

A.1.2 Generalization Gap vs Val PPL

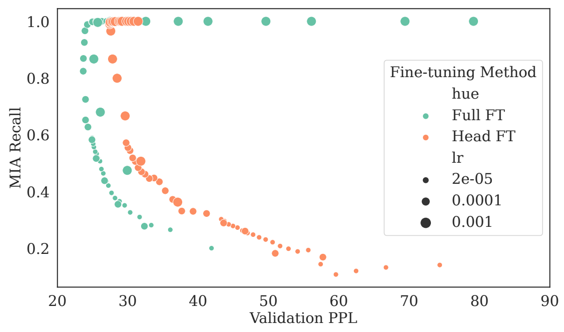
Figure 8 shows generalization gap (validation−train perplexity) versus validation perplexity. We plot this to show how this differes from MIA recall (memorization) versus perplexity (Figure 2), and to emphasize how in the *memorization only* phase, memorization is increasing (the long vertical stretch in Figure 2), however the validation perplexity and generalization gap remain almost the same (the sharp turning point in Figure 8).

A.2 Training From Scratch

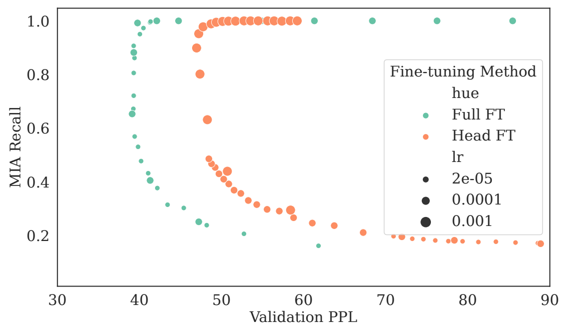
Figure 5 shows how pre-training a finetuned mdoel is different from training the model from scratch, in terms of validation perplexity and attack recall. We can see that fine-tuning a pre-trained model leaks less information, than fine-tuning from scratch.

A.2.1 Other Models

To further test how our findings generalize to other models, we repeat our experiments on the Huggingface `distilgpt2` and `openai-gpt` as well, and show the results in Figure 6. As we see, the results are commensurate with those of GPT2. We cannot run experiments with adapters here as these models are not supported by the adapter library yet.



(a) DistilGPT2



(b) OpenAI-GPT

Figure 6: Utility Vs. privacy (MIA recall) on the Penn Treebank dataset for DistilGPT2 and OpenAI-GPT models. Each dot shows different checkpoints, and the colors show different fine-tuning methods. We desire models that have low PPL and low attack recall.

A.3 Separate Plots

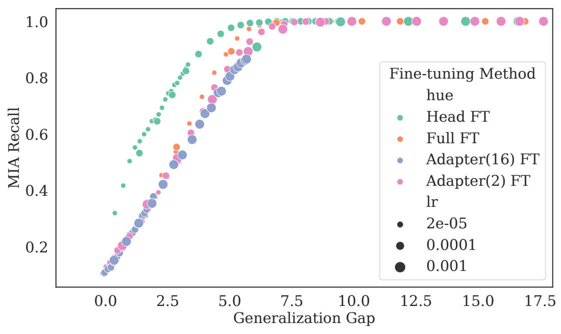
Figures 9, 10, and 11 show the MIA recall vs validation PPL for each fine-tuning method on each dataset separately, to provide better visibility. These Figures correspond to the subfigures in Figure 2.

A.4 Breaking Fine-tuning Into Phases

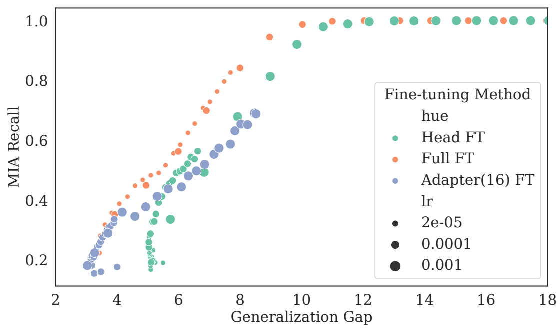
Although there is no ground truth rule on how the phases are defined, we use the following heuristic: break the training between phases 1 and 2 at points where the slope of the lines on the graph starts increasing drastically. For breaking between phases 2 and 3 we choose the point where the validation perplexity starts increasing again.

A.5 Computational Resources

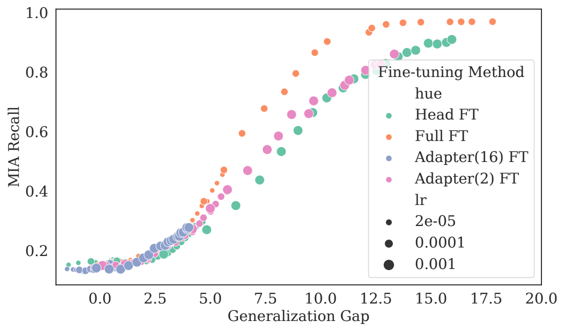
For this paper, we spent an overall 7 days in GPU time for training and evaluation. For that, we used a server with 4×RTX2080 GPU with 11GB of



(a) Wikipedia Dataset



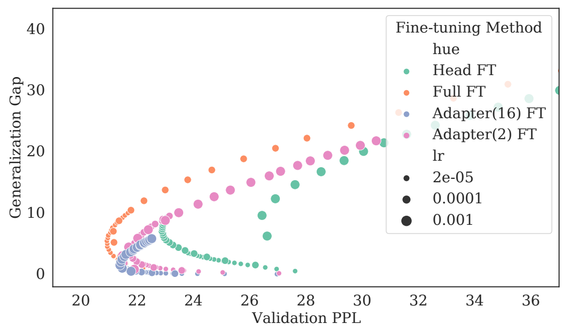
(b) Penn Treebank Dataset



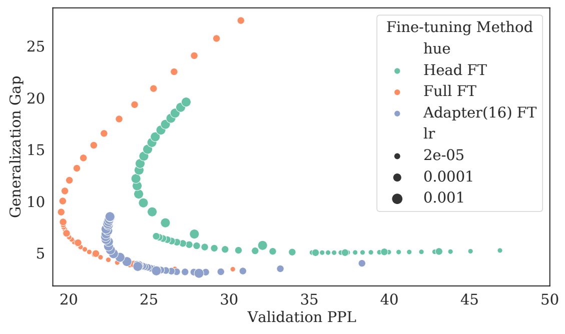
(c) Enron

Figure 7: Attack recall and generalization gap (Validation PPL- Train PPL) correlation. As the generalization gap increases, the attack observes more leakage as expected for all fine-tuning methods on both datasets.

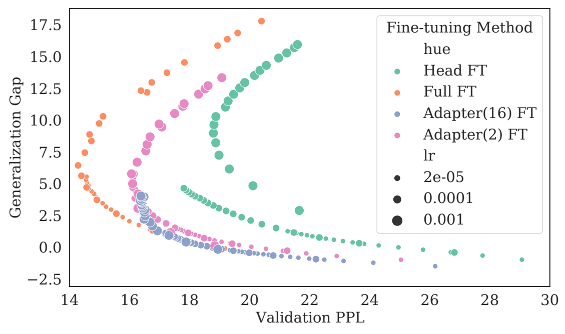
memory.



(a) Wikipedia Dataset

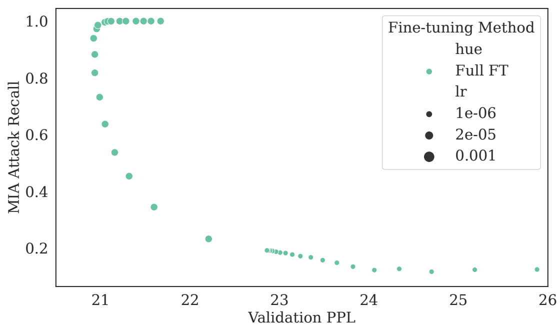


(b) Penn Treebank Dataset

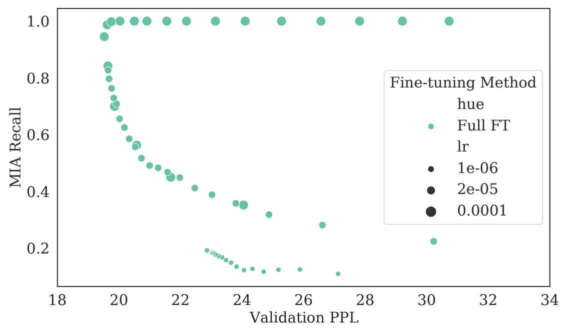


(c) Enron

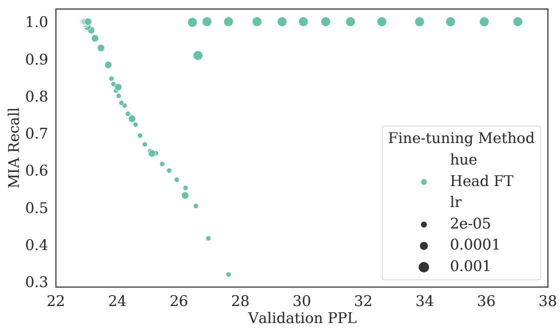
Figure 8: Validation perplexity and generalization gap (Validation PPL- Train PPL) correlation.



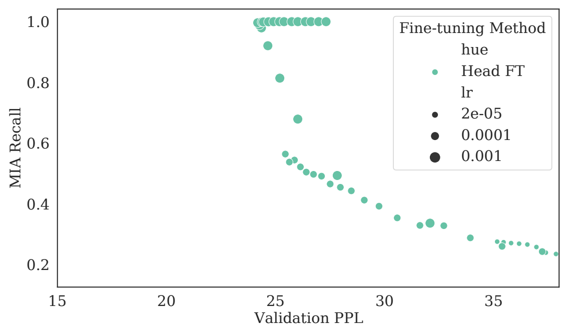
(a) Full FT



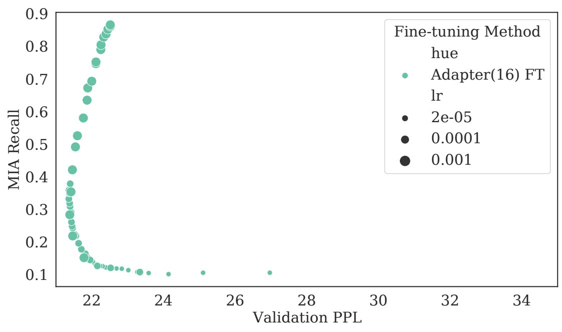
(a) Full FT



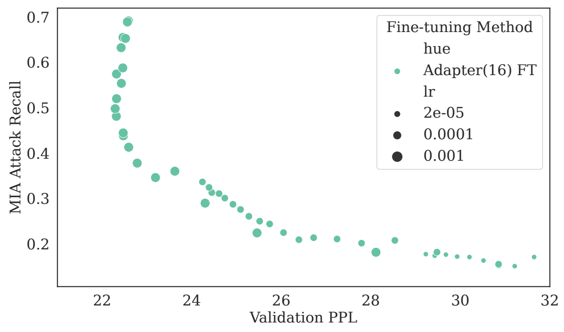
(b) Head FT



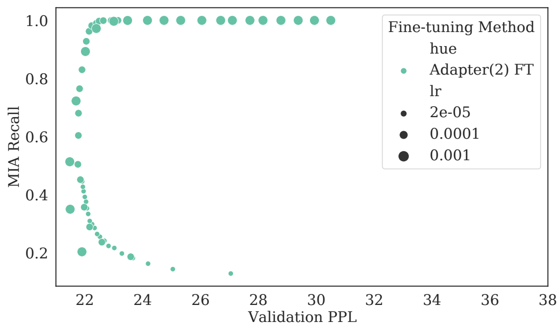
(b) Head FT



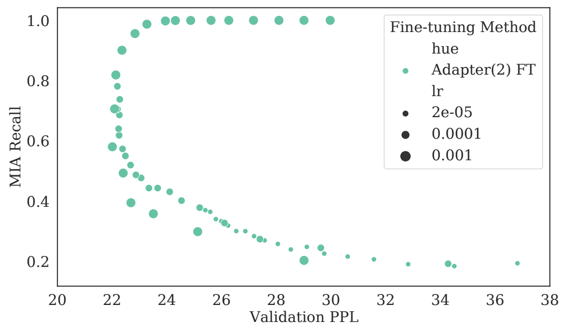
(c) Adapter(16) FT



(c) Adapter(16) FT



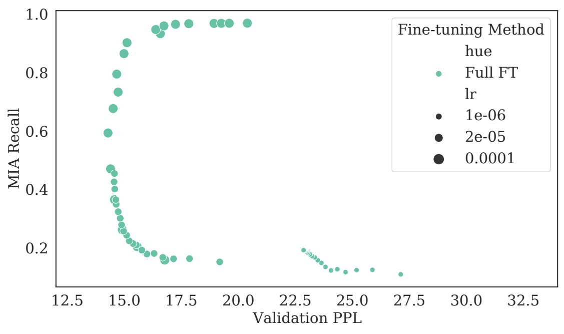
(d) Adapter(2) FT



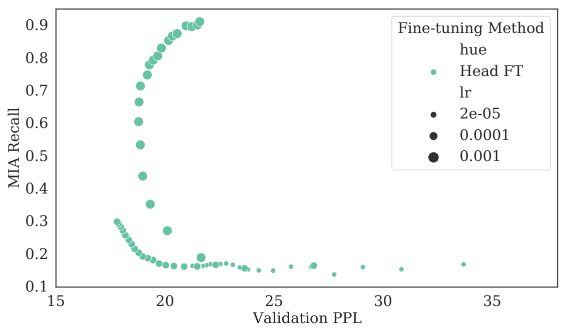
(d) Adapter(2) FT

Figure 9: Wikipedia

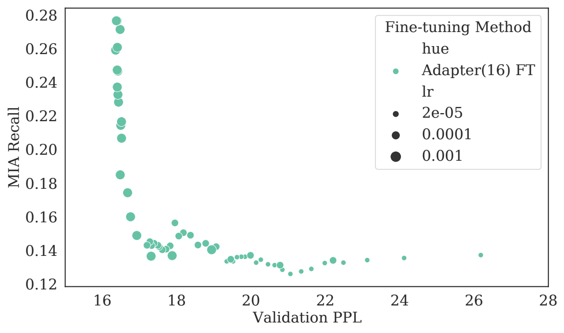
Figure 10: Penn Tree Bank



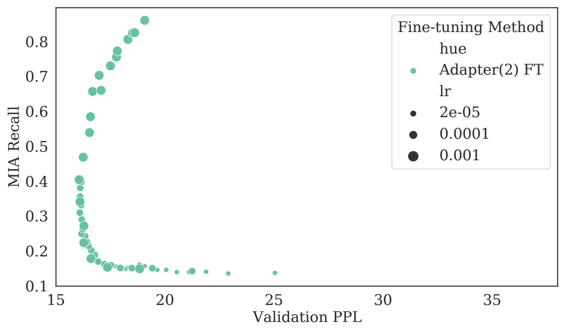
(a) Full FT



(b) Head FT



(c) Adapter(16) FT



(d) Adapter(2) FT

Figure 11: Enron