Memorization in NLP Fine-tuning Methods

Anonymous Authors[1]

Abstract

Large language models are shown to present privacy risks through memorization of training data, and several recent works have studied such risks for the pre-training phase. Little attention, however, has been given to the fine-tuning phase and it is not well understood how different fine-tuning methods (such as fine-tuning the full model, the model head, and adapter) compare in terms of memorization risk. This presents increasing concern as the "pre-train and fine-tune" paradigm proliferates. In this paper, 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.

1. Introduction

Transformer-based language models have become the models of choice for a multitude of 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 it to a specific domain (Ramponi & Plank, 2020; Li & Liang, 2021; Houlsby et al., 2019).

There is abundant work that shows 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 (Carlini et al., 2019; 2021b; 2022; Nakamura et al., 2021). More specifically, Carlini et al. (2021b); Mireshghallah et al. (2022) have mounted such attacks on pre-trained language models and shown the gravity of this issue by extracting complete training sequences and inferring membership of most of the training samples.

Although memorization has been studied during pre-training, scant attention has been given to fine-tuning, even though fine-tuning data is actually of higher concern than pre-training data. Most pre-training sets are large public corpora (Raffel et al., 2019; Dodge et al., 2021), with less privacy concerns (Brown et al., 2022). But fine-tuning sets are small, targeted, diverse, potentially very private (Basu et al., 2021; Li et al., 2021), thus, actually deserving of more attention than pre-training sets. Further, pre-training generally happens only a few times (as it needs resources that are usually only available to large companies, e.g. the pre-training of the GPT models (Brown et al., 2020)) while fine-tuning is increasingly the dominant way that end-users fit models. In this work, we focus on different fine-tuning methods and their propensity for memorization of training samples.

Given the size of these large language models, fine-tuning all the model parameters can be compute and memory intensive (Radford et al., 2019; Lewis et al., 2019; Brown et al., 2020; Fedus et al., 2021). As a result, recent work has proposed new parameter efficient fine-tuning methods that update only a subset of the model's parameters (Houlsby et al., 2019; Li & Liang, 2021; He et al., 2022). In this paper, we focus on studying memorization of the following three 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., which are small bottleneck modules inserted within transformer blocks. For measuring memorization, we use two proxy metrics for memorization: (a) recall of a reference-based membership inference attack (Mireshghallah et al., 2022) and (b) exposure metric, which measures how susceptible the model is to a sample extraction attack which tries to find a secret in the training data. We run our experiments on the Wikipedia (Merity et al., 2016), Penn Treebank (Marcus et al., 1993) and Enron Emails (Klimt & Yang, 2004) datasets, for the task of autoregressive language modeling (next word prediction). We have selected Wikipedia and Penn Treebank as they are most commonly used for fine-tuning, and selected

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the First Workshop of Pre-training: Perspectives, Pitfalls, and Paths Forward at ICML 2022. Do not distribute.

Enron since it is a dataset of emails and is private by nature.

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.

2. Related Work

An membership inference attack (MIA) tries to determine if a target sample was used to train a target model or not (Shokri et al., 2017; Yeom et al., 2018; Jayaraman et al., 2021), whereas, a training data extraction attack quantifies the risk of memorization in language models by probing the trained model (Salem et al., 2020; Carlini et al., 2019; Zanella-Béguelin et al., 2020; Carlini et al., 2021b; 2022; Nakamura et al., 2021). The presence of unintended memorization in neural language models can lead to major privacy breaches, which is distinctly shown by (Carlini et al., 2019) and can be quantified through their exposure metric. Additionally, some progress has been made in examining the perturbation in memorization on the exclusion of particular samples from the training data (Zhang et al., 2021). In this work, we use a recent MIA (Mireshghallah et al., 2022), which makes use of a reference model and a likelihood ratio test to provide a stronger measurement of training data leakage in masked language models, helps us analyze the various fine-tuning methods and the level of data leakage they bring about in language models. In terms of fine-tuning, the work most relevant to ours is adapter tuning (Houlsby et al., 2019), as we experiment with it as a fine-tuning method. Adapters are generally used to fine-tune a model for a new task without revisiting the previous tasks. This helps curb the retraining of the model for every new task.

3. Model Fine-tuning

Model fine-tuning (FT) is commonly applied to large language models to help improve performance on new data domains (domain adaptation). The fine-tuning objective can be different from the original pre-training objective, such as in Bert-based models (Devlin et al., 2018) where the pre-training objective is masked language modeling (MLM) and fine-tuning objective is usually classification. In this paper, we focus on generative language models, such as the GPT family, where both the pre-training and fine-tuning tasks are autoregressive language modeling (next word prediction).

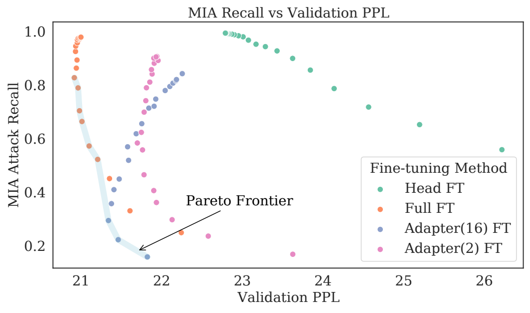
Given the size of transformer-based language models, fine-tuning the entire model can be compute-intensive and inconvenient, therefore there have been a multitude of 'parameter-efficient' fine-tuning methods that enable domain adaptation by training only a small set of model parameters (Houlsby et al., 2019; Hu et al., 2021; Li & Liang, 2021; He et al., 2022). We focus on two main FT methods: (1) fine-tuning the model head, i.e., the prediction layer, as it is the most common FT method used in practice. (2) fine-tuning adapters (Houlsby et al., 2019). Adapters are small bottle-neck 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 fine-tuning, the rest of the model remains frozen, therefore the number of trainable parameters is low (around 1% of the full model parameters). As a baseline, we also experiment with (3), full model fine-tuning which updates all 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. The number of parameters for each of these methods, along with the number of parameters for each transformer block are all shown in the last row of Table 1.

4. Measuring Memorization

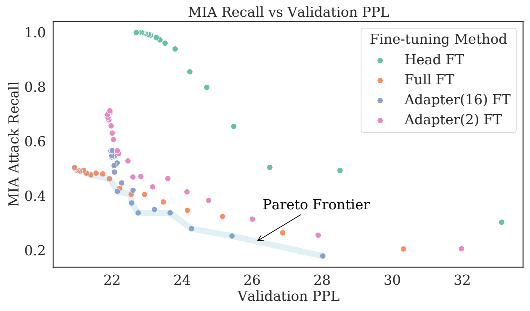
To measure memorization of fine-tuning training samples by different fine-tuning methods, we use two metrics:

Membership Inference Attack (MIA Recall). We use the percentage of training samples that are correctly classified as training set members by the reference-based attack proposed in (Mireshghallah et al., 2022; Carlini et al., 2021a) as a proxy metric of memorization. This is how the attack operates: 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 such that 10% of the validation samples would be falsely classified as members (i.e., so that the false positive rate would be 10%). The higher the recall of this attack is, the higher the leakage of the model is. In our experiments, we use a pre-trained GPT-2 as the reference model.

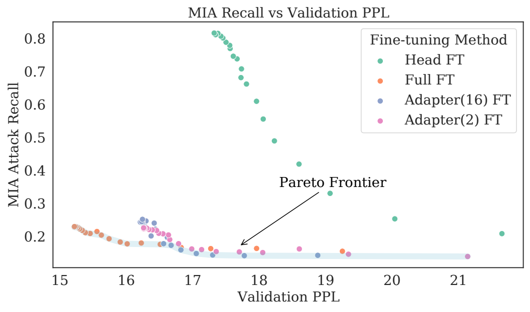
Exposure. As a secondary measure of memorization, we use the exposure metric from Carlini et al. (2019) which inserts a secret (canary) of a certain format into the training data and calculates its vulnerability to extraction. Exposure



(a) Wikipedia Dataset

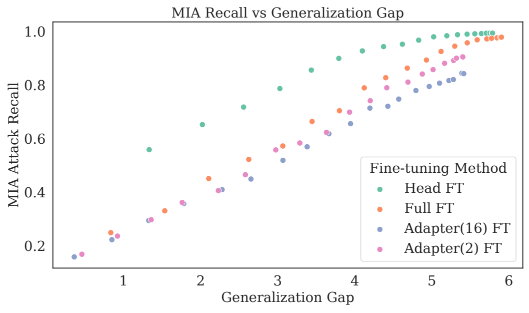


(b) Penn Treebank Dataset

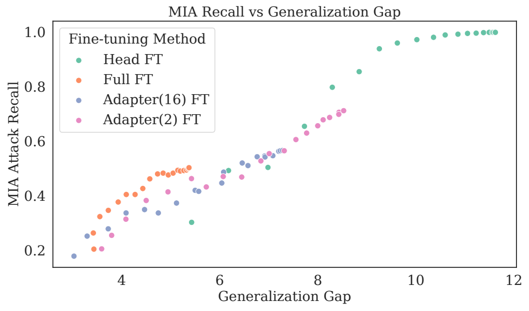


(c) Enron Dataset

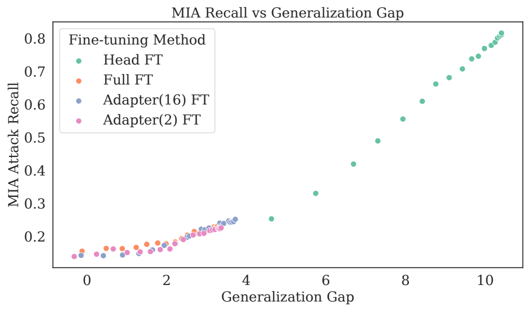
Figure 1: Pareto frontier for utility (validation PPL) Vs. privacy (MIA attack recall). Each dot shows different checkpoints, and the colors show different fine-tuning methods. We desire models that have low PPL and low attack recall.



(a) Wikipedia Dataset



(b) Penn Treebank Dataset



(c) Enron

Figure 2: 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.

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. 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.

5. Results

We run our experiments on the following datasets: (1) Huggingface's Wikipedia wikitext-2-raw-v1 dataset (2) Huggingface's Penn Treebank ptb_text_only and (3) a sub-sampled version of Enron email dataset. We use sequence length of 1024, training batch size of 8, and fine-tune for 20 epochs. We fine-tune the pre-trained GPT-2 model from Huggingface, and use adapter hub's implementation of adapters (Pfeiffer et al., 2020). We evaluate memorization at each epoch using the MIA recall and exposure metrics described in the previous section. All the points shown in Figures 1 and 2 are from evaluations at different training epochs, therefore there are 20 points

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	25.26	24.41
	Exposure	1.42	10.78	0.83	14.54
PTB	Val PPL	29.55	29.03	29.41	29.79
	Exposure	7.03	13.62	4.54	12.40
Enron	Val PPL	12.52	13.51	12.81	13.03
	Exposure	1.32	10.77	0.440	2.02

for each method. For each method, we have chosen the set of hyperparameters (learning rate) that provide the best validation perplexity for the method.

5.1. Memorization of Fine-tuning Methods

Figure 1 compares the fine-tuning methods in terms of privacy leakage, measured by MIA recall; Table 1 shows the exposure results for the three datasets, along with their parameter counts. Results for both the MIA recall and exposure metrics are consistent, showing higher leakage for head fine-tuning and lower for full model fine-tuning and adapters. Each data point in Figure 1 corresponds to one epoch during the 20 epochs of the training process, hence there are 20 points for each method. The blue lines show the Pareto frontier, the desirable trade-off points, given how low recall and low PPL are desirable.

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 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.

Correlation between Generalization and Memorization. Figure 2 shows the correlation between the generalization gap and membership inference attack recall. The generalization gap refers to the subtraction of train 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.

5.2. 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. Discussion

In this paper we study and compare memorization in different fine-tuning methods, using the recall of a membership inference attack and the exposure metric on three text datasets: Wikipedia, Penn treebank, and Enron email dataset. We find that the common practice of fine-tuning only the head of a model has the highest memorization (by a large margin) among different fine-tuning methods, for the same levels of validation perplexity (utility). We 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. Adapters are overall less susceptible to state-of-the-art inference attacks, but with slightly lower perplexity than full fine-tuning. Full fine-tuning on the other hand has better validation perplexity with slightly higher attack recall. This work is a preliminary study on memorization in fine-tuning. We argue 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 results are a first step to understanding those differences, but further work is needed to understand how inference risks vary with fine-tuning methods and to develop methods that provide better privacy-utility trade-offs.

References

Basu, P., Roy, T. S., Naidu, R., Muftuoglu, Z., Singh, S., and Mireshghallah, F. Benchmarking differential privacy

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 Attack 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

and federated learning for bert models. *arXiv preprint arXiv:2106.13973*, 2021.

Brown, H., Lee, K., Mireshghallah, F., Shokri, R., and Tramèr, F. What does it mean for a language model to preserve privacy? *arXiv preprint arXiv:2202.05520*, 2022.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 2020.

Carlini, N., Liu, C., Erlingsson, U., Kos, J., and Song, D. The Secret Sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security Symposium*, 2019.

Carlini, N., Chien, S., Nasr, M., Song, S., Terzis, A., and Tramer, F. Membership inference attacks from first principles. *arXiv preprint arXiv:2112.03570*, 2021a.

Carlini, N., Tramer, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, U., Oprea, A., and Raffel, C. Extracting training data from large language models. In *USENIX Security Symposium*, 2021b.

Carlini, N., Ippolito, D., Jagielski, M., Lee, K., Tramer, F., and Zhang, C. Quantifying memorization across neural language models, 2022. URL https://arxiv.org/abs/2202.07646.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Chen, M. X., Lee, B. N., Bansal, G., Cao, Y., Zhang, S., Lu, J., Tsay, J., Wang, Y., Dai, A. M., Chen, Z., et al. Gmail smart compose: Real-time assisted writing. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dodge, J., Sap, M., Marasović, A., Agnew, W., Ilharco, G., Groeneveld, D., and Gardner, M. Documenting the english colossal clean crawled corpus. *ArXiv*, abs/2104.08758, 2021.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*, 2021.

He, J., Zhou, C., Ma, X., Berg-Kirkpatrick, T., and Neubig, G. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*, 2022.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pp. 2790–2799. PMLR, 2019.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Jayaraman, B., Wang, L., Knipmeyer, K., Gu, Q., and Evans, D. Revisiting membership inference under realistic assumptions. In *Privacy Enhancing Technologies Symposium*, 2021.

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

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

Li, X., Tramer, F., Liang, P., and Hashimoto, T. Large language models can be strong differentially private learners. *arXiv preprint arXiv:2110.05679*, 2021.

Li, X. L. and Liang, P. 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)*, 2021.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. URL https://www.aclweb.org/anthology/J93-2004.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models, 2016.

Mireshghallah, F., Goyal, K., Uniyal, A., Berg-Kirkpatrick, T., and Shokri, R. Quantifying privacy risks of masked language models using membership inference attacks, 2022.

Nakamura, Y., Hanaoka, S., Nomura, Y., Hayashi, N., Abe, O., Yada, S., Wakamiya, S., and Aramaki, E. KART: Parameterization of privacy leakage scenarios from pre-trained language models, 2021.

Pfeiffer, J., Rücklé, A., Poth, C., Kamath, A., Vulić, I., Ruder, S., Cho, K., and Gurevych, I. Adapterhub: A framework for adapting transformers. In *Conference on Empirical Methods in Natural Language Processing (Systems Demonstrations)*, 2020.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

Ramponi, A. and Plank, B. Neural unsupervised domain adaptation in NLP—a survey. *arXiv preprint arXiv:2006.00632*, 2020.

Salem, A., Bhattacharya, A., Backes, M., Fritz, M., and Zhang, Y. Updates-Leak: Data set inference and reconstruction attacks in online learning. In *USENIX Security Symposium*, 2020.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. Membership inference attacks against machine learning models. In *IEEE Symposium on Security and Privacy*, 2017.

Yeom, S., Giacomelli, I., Fredrikson, M., and Jha, S. Privacy risk in machine learning: Analyzing the connection to overfitting. In *IEEE Computer Security Foundations Symposium*, 2018.

Zanella-Béguelin, S., Wutschitz, L., Tople, S., Rühle, V., Paverd, A., Ohrimenko, O., Köpf, B., and Brockschmidt, M. Analyzing information leakage of updates to natural language models. In *ACM SIGSAC Conference on Computer and Communications Security*, 2020.

Zhang, C., Ippolito, D., Lee, K., Jagielski, M., Tramèr, F., and Carlini, N. Counterfactual memorization in neural language models. *arXiv preprint arXiv:2112.12938*, 2021.