NAVIGATING THE DESIGNS OF PRIVACY-PRESERVING FINE-TUNING FOR LARGE LANGUAGE MODELS

Haonan Shi, Tu Ouyang & An Wang
Case Western Reserve University
{haonan.shi3, tu.ouyang, an.wang}@case.edu

ABSTRACT

Instruction tuning has proven effective in enhancing Large Language Models' (LLMs) performance on downstream tasks. However, real-world fine-tuning faces inherent conflicts between model providers' intellectual property protection, clients' data privacy requirements, and tuning costs. While recent approaches like split learning and offsite tuning demonstrate promising architectures for privacy-preserving fine-tuning, there is a gap in systematically addressing the multidimensional trade-offs required for diverse real-world deployments. We propose several indicative evaluation metrics to guide design trade-offs for privacy-preserving fine-tuning and a series of example designs, collectively named *GuardedTuning*; they result from novel combinations of system architectures with adapted privacy-enhancement methods and emerging computation techniques. Each design represents distinct trade-offs across model utility, privacy guarantees, and costs. Experimental results demonstrate that these designs protect against data reconstruction attacks while maintaining competitive fine-tuning performance.

1 INTRODUCTION

Large Language Models (LLMs) (Zhang et al., 2022; Touvron et al., 2023) have demonstrated exceptional performance across a wide range of natural language processing tasks. Through pretraining on massive text corpora, these models learn rich language representations and knowledge that exhibit strong generalization capabilities across various downstream tasks. Although LLMs typically demonstrate reasonable zero-shot learning abilities, research shows that fine-tuning on specific tasks can significantly improve model performance on these tasks (Wei et al., 2021). Fine-tuning the LLMs with domain-specific data has emerged as an important paradigm to adapt pre-trained LLMs to better suit some domain-specific tasks.

Due to substantial computational investments, model providers consider these LLMs proprietary assets and want to guard against leakage. Conversely, even though some model providers offer fine-tuning services through APIs (e.g., OpenAI fine-tuning APIs [1]), it requires the client to upload their private data to the model provider, fine-tuning clients might want to safeguard the fine-tuning data even against model providers due to the sensitivity of the data or regulation requirements. Overall, the requirements of real-world fine-tuning solutions are diverse and subject to sometimes conflicting participating parties' concerns, particularly regarding privacy and budget.

To address these privacy conflicts between fine-tuning participants, Recent privacy-preserving approaches like split learning (Vepakomma et al., 2018; Wang et al., 2023) and offsite tuning (Xiao et al., 2023) propose new system architectures that partition the training process between providers and users, where only model intermediate values or compressed model components are exchanged rather than raw input data. Different system architectures enable distinct trade-offs that fit some fine-tuning requirements but not others. In addition, privacy-enhancement methods, some of which choose to add noise with respect to different optimization objectives to either input or intermediate data, further broaden the privacy-preserving fine-tuning solution design space.

We advocate evaluating these emerging privacy-preserving fine-tuning designs using multiple metrics but not any single one. We propose using metrics in three categories, utility, privacy (of each

[1] https://platform.openai.com/docs/guides/text-generation

participating party), and tuning cost, for both design evaluation and to help navigate the design trade-offs necessary to meet diverse real-world requirements. We acknowledge that different system architectures, combined with different privacy-enhancement methods, could result in various solutions with distinct utility, privacy, and cost characteristics.

To showcase how different design trade-offs impact the metrics and hence help navigate the design space, we produce a series of example designs for privacy-preserving fine-tuning, collectively named *GuardedTuning*, from novel combinations of system architectures, adapted privacy-enhancement methods and emerging computation techniques. We articulate the trade-offs underneath these designs and their differences from existing ones in the literature in section 3 and present some preliminary quantitative results on the evaluation metrics these designs achieve in section 4. Our experiment results demonstrate that *GuardedTuning* effectively protects both server's model weights and client's data privacy, reducing data reconstruction attacks effectiveness to below 50% and decreasing communication cost by up to 73.7%, while the degrade of fine-tuned model utility is $< 1.5\%$ in almost all of the cases.

2 THREAT MODEL AND DESIGN TRADE-OFFS

In our privacy-preserving fine-tuning setting, we consider an honest-but-curious model provider (also referred to as *server*) that has access to the pre-trained model weights. Fine-tuning client provides data for fine-tuning training and inferences. The fine-tuning training and inference could be split between these two parties. While faithfully following the fine-tuning protocol, the server can not be fully trusted and might try to look into the client's data by performing data reconstruction attacks(DRAs) during either training or inference phases using the activations or gradients transmitted from client. Conversely, the model provider also needs to protect their intellectual property. So they cannot hand over the pre-trained model to the client.

There is a vast design space to produce a specific fine-tuning design that satisfies the privacy concerns of both the server and the client, as well as other requirements arising from, for instance, the utility of the tuned model and operation budgets. We propose to use the following metrics to evaluate different solutions and leverage the metrics as guidance to arrive at the right balance when making concrete design choices for a specific set of requirements: 1. The utility of the fine-tuned model; 2. The privacy of clients' data for fine-tuning training; 3. The privacy of clients' data for inference with fine-tuned model; 4. The server's model privacy; 5. Communication cost associated with fine-tuning.

Our concrete designs below show how to navigate the trade-offs among these metrics.

3 GUARDEDTUNING ARCHITECTURES

3.1 ONLINE GUARDEDTUNING

Inspired by the split learning, we propose *Online GuardedTuning*, as shown in Figure 1. In this design, the server distributes several bottom and top transformer layers of a pre-trained LLM to the client for collaborative fine-tuning. We call the layers clients hold adapter layers, and those servers kept as backbone layers. During fine-tuning training, the client and server exchange activations for forward pass and gradients for backward propagation. Client data privacy is enhanced by avoiding plain text transfers to the server, though this introduces additional communication cost during both fine tuning and inference.

This design has a utility-cost trade-off: while adapter layers in the client always require fine-tuning, the server can choose to fine-tune the backbone layers or not—the second choice results in lower overall cost but with a reduced model utility. Prior research (Wu et al., 2024; Xiao et al., 2023) suggests that fine-tuning a few selected layers of the model can still yield reasonable performance, slightly lower than fully fine-tuning.

Through the activations uploaded by clients, the server can reconstruct clients' data via DRAs (Song & Raghunathan, 2020; Chen et al., 2024) with $> 90\%$ attack effectiveness, rendering the fine-tuning design useless as client data is close to having no protection. Existing solutions like Pri-

vateLoRA(Wang et al., 2023), despite retaining embedding layer, head layer and low-rank adapters locally, remain vulnerable to server-side DRAs due to unprotected activation transmission.

To defend against such attacks and realize better client data privacy, we introduce two novel privacy-enhancing methods that can be composed into several *GuardedTuning* designs. These methods help partially dwarf such DRAs at the cost of a small amount of extra computation for data perturbation.

First, We apply a *distance decorrelation* technique, adapted from the Nopeek (Vepakomma et al., 2020), to transform activations output from the input adapter layers. Vepakomma *et al.* identify that the distance correlation $dCor(x, \theta(x))$ between input data x and activations $\theta(x)$ indicates the risk of input data being inverted from intermediary representations, they incorporate $dCor(x, \theta(x))$ as a regularization term in the training loss function. By minimizing $dCor(x, \theta(x))$ during training, they could mitigate DRAs. Unlike Nopeek, which uses Euclidean distance in computing distance correlation between input data x and activations $\theta(x)$, we instead calculate the distance correlation between input embeddings $emb(x)$ and activations $\theta(x)$ using cosine distance, which achieves better performance-privacy trade-off for LLMs. Cosine distance provides a more appropriate representation of distances in the high-dimensional embedding space in LLMs. After incorporating the decorrelation operation, the fine-tuning loss function becomes:

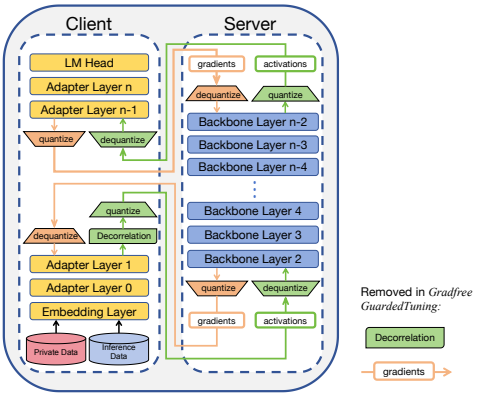

Figure 1: Architectures for *Online GuardedTuning* and *Gradfree GuardedTuning*.

$$L = L(f(x), y) + \lambda dCor(emb(x), \theta(x)) \tag{1}$$

where f represent the LLM, θ denotes the input adapter layers, $dCor()$ denotes the distance correlation function(Vepakomma et al., 2020), and λ represents the regularization weight.

However, adding a regularization term alone cannot protect fine-tuning data in early training stages. To address this limitation, we additionally apply a quantization-based technique that protects the beginning stage of the training. During transmission, the sender quantizes the inlier values of activations and gradients while preserving outliers:

$$Q(A) = \begin{cases} \text{round}\left(\frac{A - \min(A)}{P_p(A) - \min(A)} \cdot (2^b - 1)\right) & \text{if } A \leq P_p(A) \\ A & \text{if } A > P_p(A) \end{cases} \tag{2}$$

$$\hat{A} = \begin{cases} Q(A) \cdot s + \min(A) & \text{if bit-width}(Q(A)) = b \\ Q(A) & \text{if bit-width}(Q(A)) = 32 \end{cases} \tag{3}$$

where $s = \frac{P_p(A) - \min(A)}{2^b - 1}$ is the scaling factor transmitted with $\min(A)$ and quantized values $Q(A)$. The quantization bits b and the p-th percentile threshold are determined locally by the sender. Our experiments show that quantization introduces controlled noise that helps realize better privacy protection without leading to significant model performance degradation after fine-tuning.

We note that quantization also helps reduce communication costs since the transmitted data is smaller in size than split learning designs. Additional techniques could be included in this design to reduce communication costs further. For instance, by leveraging emerging trusted computing technologies, like hardware-assisted secure GPU sharing (Corporation, 2023), or confidential computing techniques that enable two parties sharing the same host machine with privacy attestation by cloud vendors (Microsoft, 2024), it is possible to run separated components of the fine-tuning design on the same machine, even on the same GPU, with similar privacy protection as running them on different machines. While the communication volume is the same, the time of achieving a similar amount of communication significantly improves if employing these techniques, such as NVLink.

3.2 GRADFREE GUARDEDTUNING

Although *Online GuardedTuning* offers substantial decreases in the DRAs attack effectiveness as shown in our evaluations in section 4, we note that privacy attacks like gradient matching-based DRAs (Chen et al., 2024; Deng et al., 2021) can still marginally improve DRA's performance. We introduce a design called *Gradfree GuardedTuning*, which favors better client data privacy than *Online GuardedTuning* at the cost of a slight decrease in model utility.

As shown in Figure 1, in *Gradfree GuardedTuning*, we choose to fine-tune only the output adapter; by doing so, it removes the gradient backpropagation between the client and server required in *online GuardedTuning*. Removing the gradient backpropagation provides full defense against gradient matching-based DRAs. Although the absence of gradient transmission disallows some other privacy-enhancing methods like decorrelation operations, we still can apply method like quantization on activations in this design, this method provides substantive defense to several DRAs.

3.3 OFFLINE GUARDEDTUNING

To realize better client data privacy, we design *offline GuardedTuning*, an enhancement to offsite-tuning (Xiao et al., 2023). This design is as shown in Figure 2, Similar to offsite tuning, in our design, during fine-tuning training, the server provides clients with adapter layers and an emulator constructed through uniform layers dropping from the backbone. This emulator enables clients to simulate the backbone layers and run fine-tuning training entirely within the client, thereby protecting fine-tuning data from server-side data reconstruction attacks during training. The design trade-off is to sacrifice the sever model privacy to a certain degree (because the emulator leaks information about the base model) in exchange for complete client training data privacy.

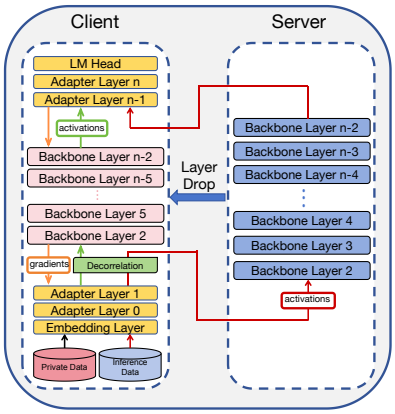

Figure 2: Architecture for *Offline GuardedTuning*.

In offsite tuning, the client is supposed to transfer their fine-tuned input and output layers to the server and run the entire inference there, leaving the inference-time client data vulnerable. To address this privacy limitation, in *offline guardedtuning*, we choose to let the client keep the input and output adapter layers, only transferring activations to the server for inference. Furthermore, we co-train decorrelation operators during the fine-tuning training phase and use the decorrlation-tuned input adapter layers during inference for activation values, to protect inference data against DRAs. By adding decorrelation operation, our design enhances client inference data privacy with a small amount of data perturbation cost.

Xiao *et al.* (Xiao et al., 2023) observe that the emulator, coupled with fine-tuned adapters, their inference performance may be significantly weaker than the base model's zero-shot. Therefore, in our design, we choose to use the backbone layers in the server instead of the emulator for inference, a similar design choice in offsite learning. This low emulator inference performance might indicate a lower knowledge capacity in the emulator and, thus, a lower risk of allowing the client to reverse-engineer the base model using an emulator or leverage it for other downstream tasks.

4 EVALUATIONS

In this section, we evaluate the three proposed *GuardedTuning* designs and show how their trade-offs affect the utility, privacy, and cost metrics.

4.1 EXPERIMENT SETUP

Datasets. We evaluate *GuardedTuning* on four reasoning task datasets: ARC-Easy (Clark et al., 2018) and OpenBookQA (Mihaylov et al., 2018) for multiple-choice scientific reasoning, PIQA (Bisk et al., 2020) for binary physical commonsense reasoning, and WebQuestions (Berant et al., 2013) for open-domain question answering.

Evaluated Models. We conducted experiments using OPT-1.3B and OPT-6.7B models (Zhang et al., 2022).

Experiment Configurations. In experiments using OPT-1.3B, the client holds the first and last two layers as adapters with the server keeping 20 backbone layers (8 layers for the backbone emulator), while in experiments involving OPT-6.7B, 3 adapter layers for both input and output, and 26 backbone layers (14 layers for the emulator). During fine-tuning, we set the decorrelation weight λ to 5 in Equation 1, employ 8-bit quantization for transmission, and use the 99th percentile as the quantization threshold in Equation 2.

Metrics. We measure ***client data privacy*** through state-of-the-art DRAs (BiSR(b) and BiSR(f)) (Chen et al., 2024) during both fine-tuning and inference phases. Attack effectiveness is quantified using ROUGE-L F1 scores, with higher scores indicating greater privacy leakage. DRAs are conducted via three stages: inverter training with distributed adapters, initial reconstruction using server activations, and optimization through activations/gradients. We conduct evaluations every 100 training steps, averaging results across 5 independent batches. For the designs that involve gradient exchange (split learning-based fine-tuning, *online GuardedTuning*), we apply gradient-based BiSR(b) during fine-tuning. Otherwise, we employ BiSR(f) to attack the activations from forward propagation during fine-tuning and inference. For ***server model privacy***, we use the number of shared model layers as a quantitative proxy metric. ***Model utility*** is measured as task-specific accuracy using lm-evaluation-harness (EleutherAI, 2024). To evaluate ***communication cost***, we quantify the data transmission volume between client and server during the fine-tuning process.

Baseline. We use traditional split learning-based fine-tuning(SL tuning) (Vepakomma et al., 2018) and offsite tuning(Xiao et al., 2023) as our baseline.

4.2 EXPERIMENT RESULTS

We evaluate all the *GuardedTuning* designs on OPT-1.3B and OPT-6.7B models. Table 1 shows OPT-1.3B results on the utility and privacy trade-off, that all designs reach reasonable utility-privacy balance: all surpass OPT-1.3B's zero-shot task accuracy, achieve slightly lower accuracy than SL tuning baseline that has no defense to state-or-the-art DRAs while significantly reducing the server's DRA attack effectiveness to less than 50% (a decrease of at least 40% from the baseline) during both fine-tuning and inference phases. Additionally, our quantization-dequantization mechanism reduces communication cost by up to 73.7% during fine-tuning.

Both *online* and *Gradfree GuardedTuning* designs send 4 pre-train model layers to the client in OPT-1.3B model experiment, which is the server's model privacy cost. *Gradfree GuardedTuning* mitigates DRAs by eliminating backpropagation to the server. However, by only training output adapter layers, it incurs a cost of reduced model utility. *Offline GuardedTuning* provides the strongest client-data privacy by keeping client data entirely local during fine-tuning, also protecting inference privacy through keep the fine-tuned adapters locally during the inference phase. Using the emulator for fine-tuning degrades the model utility slightly compared to *online GuardedTuning* but still better than *Gradfree GuardedTuning*, consistent with observations in offsite-tuning (Xiao et al., 2023). There are potentially bigger risks to sever model privacy since the client has the emulator generated from the base model. We leave risk quantification of an emulator for future investigation.

Trade-offs between utility and privacy results on OPT-6.7B are shown in Table 2. For these experiments, we opt for a choice that fine-tunes only the adapter layers in both SL tuning and *online GuardedTuning*. This choice leads to computation cost savings, and even with that reduced resource for fine-tuning, all tuned models still achieve utility higher than zero-shot. We observe similar privacy and utility characteristics to our OPT-1.3B experiments.

Table 1: Utility and privacy trade-offs on OPT-1.3B.

Designs	Metrics	OPT-1.3B			
		OpenbookQA	ARC-Easy	PIQA	WebQs
Zero Shot	Accuracy	23.4%	56.9%	71.6%	4.6%
SL tuning (2-20-2)	Accuracy	31.4%	61.3%	75.2%	31.2%
	Fine-tune privacy	89.4	94.4	86.3	88.2
	Inference privacy	84.6	92.4	83.1	87.5
	Communication cost(GB)	15.2	13.3	326.1	14.1
offsite tuning (2-20(8)-2)	Accuracy	29.1%	59.4%	74.5%	27.2%
	Fine-tune privacy	N/A	N/A	N/A	N/A
	Inference privacy	100	100	100	100
	Communication cost(GB)	3.5	3.5	3.5	3.5
online GT (2-20-2)	Accuracy	30.7%	60.2%	73.9%	32.2%
	Fine-tune privacy	37.7	41.1	28.0	38.7
	Inference privacy	30.1	30.5	24.8	21.2
	Communication cost(GB)	4.6	4.1	85.7	4.3
Gradfree GT (2-20-2)	Accuracy	28.2%	57.8%	73.1%	26.9%
	Fine-tune privacy	34.7	37.1	27.5	24.0
	Inference privacy	32.9	27.1	26.3	21.6
	Communication cost(GB)	2.7	2.5	42.8	2.6
offline GT (2-20(8)-2)	Accuracy	29.0%	59.2%	73.2%	27.4%
	Fine-tune privacy	N/A	N/A	N/A	N/A
	Inference privacy	30.4	30.1	24.6	21.3
	Communication cost(GB)	2.6	2.6	2.6	2.6

[1] "Fine-tune privacy" and "Inference privacy" values represent the DRA attack effectiveness(ROUGE-L F1) achieved during the respective phases; the lower, the better privacy.
[2] "N/A" means the selected privacy attacks are infeasible.
[3] The nums below each design name show num of layers for (input adapter-backbone(emulator))-output adapter). Those in red are layers given to the client.

Table 2: Utility and privacy trade-offs on OPT-6.7B.

Designs	Metrics	OPT-6.7B			
		OpenbookQA	ARC-Easy	PIQA	WebQs
Zero Shot	Accuracy	27.6%	65.6%	76.2%	8.8%
SL tuning (3-26-3)	Accuracy	34.9%	68.4%	78.1%	33.9%
	Fine-tune privacy	85.1	93.2	89.4	89.1
	Inference privacy	83.9	92.7	85.2	87.3
	Communication cost(GB)	33.6	29.8	655.5	31.3
offsite tuning (3-26(14)-3)	Accuracy	33.5%	67.2%	77.8%	30.9%
	Fine-tune privacy	N/A	N/A	N/A	N/A
	Inference privacy	100	100	100	100
	Communication cost(GB)	21.8	21.8	21.8	21.8
online GT (3-26-3)	Accuracy	33.6%	67.8%	77.9%	31.7%
	Fine-tune privacy	46.2	46.9	27.6	39.5
	Inference privacy	35.6	41.7	26.3	24.2
	Communication cost(GB)	12.4	11.4	172.5	11.8
Gradfree GT (3-26-3)	Accuracy	32.1%	66.4%	76.9%	30.4%
	Fine-tune privacy	42.6	44.1	27.1	33.9
	Inference privacy	41.5	42.5	27.4	32.5
	Communication cost(GB)	8.7	8.2	88.7	8.4
offline GT (3-26(14)-3)	Accuracy	33.2%	67.1%	77.2%	30.1%
	Fine-tune privacy	N/A	N/A	N/A	N/A
	Inference privacy	34.6	37.5	19.7	29.3
	Communication cost(GB)	16.8	16.8	16.8	16.8

[1] "Fine-tune privacy" and "Inference privacy" values represent the DRA attack effectiveness(ROUGE-L F1) achieved during the respective phases; the lower, the better privacy.
[2] "N/A" means the selected privacy attacks are infeasible.
[3] The nums below each design name show num of layers for (input adapter-backbone(emulator))-output adapter). Those in red are layers given to the client.

5 CONCLUSION

In this paper, we address the challenge of producing privacy-preserving fine-tuning designs that balance protecting the client's data privacy, model provider's intellectual property, and the overall tuning cost. We introduce several novel designs, collectively named *GuardedTuning*. Through these designs, we showcase how to make design tradeoffs to meet various model utility, privacy, and cost requirements for real-world deployments. Through experiments, we demonstrate that each of our designs achieves distinct tradeoffs through selective combinations of architecture and novel privacy-enhancement methods, and they all provide defenses against some state-of-the-art privacy attacks on client data.

REFERENCES

Jonathan Berant, Andrew Chou, Roy Frostig, et al. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP*, pp. 1533–1544, 2013.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of AAAI*, volume 34, pp. 7432–7439, 2020.

Guanzhong Chen, Zhenghan Qin, Mingxin Yang, et al. Unveiling the vulnerability of private fine-tuning in split-based frameworks for large language models: A bidirectionally enhanced attack. *arXiv preprint*, 2024.

Peter Clark, Isaac Cowhey, Oren Etzioni, et al. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint*, 2018.

NVIDIA Corporation. Confidential computing on nvidia h100 gpus for secure and trustworthy ai. `https://nvidia.com/h100-secure-ai`, 2023.

Jieren Deng, Yijue Wang, Ji Li, et al. Tag: Gradient attack on transformer-based language models. *arXiv preprint*, 2021.

EleutherAI. Lm evaluation harness. `https://bit.ly/LM-EvalHarness`, 2024.

Microsoft. Confidential containers on azure container instances. `https://aka.ms/confidential-containers`, 2024.

Todor Mihaylov, Peter Clark, Tushar Khot, et al. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint*, 2018.

Congzheng Song and Ananth Raghunathan. Information leakage in embedding models. In *Proceedings of CCS*, pp. 377–390, 2020.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, et al. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint*, 2018.

Praneeth Vepakomma, Abhishek Singh, Otkrist Gupta, et al. Nopeek: Information leakage reduction to share activations in distributed deep learning. In *Proceedings of ICDMW*, pp. 933–942. IEEE, 2020.

Yiming Wang, Yu Lin, Xiaodong Zeng, et al. Privatelora for efficient privacy preserving llm. *arXiv preprint*, 2023.

Jason Wei, Maarten Bosma, Vincent Y Zhao, et al. Finetuned language models are zero-shot learners. *arXiv preprint*, 2021.

Feijie Wu, Zitao Li, Yaliang Li, et al. Fedbiot: Llm local fine-tuning in federated learning without full model. In *Proceedings of KDD*, pp. 3345–3355, 2024.

Guangxuan Xiao, Ji Lin, and Song Han. Offsite-tuning: Transfer learning without full model. *arXiv preprint*, 2023.

Susan Zhang, Stephen Roller, Naman Goyal, et al. Opt: Open pre-trained transformer language models. *arXiv preprint*, 2022.