LOTTERY TICKET ADAPTATION: MITIGATING DESTRUCTIVE INTERFERENCE IN LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing methods for adapting large language models (LLMs) to new tasks are not suited to multi-task adaptation because they modify all the model weights–causing destructive interference between tasks. The resulting effects, such as catastrophic forgetting of earlier tasks or performance drop due to model merging, make it challenging to obtain good performance on multiple tasks at the same time. To mitigate this, we propose Lottery Ticket Adaptation (LoTA), a sparse adaptation method that identifies and optimizes only a sparse subnetwork of the model. We evaluate LoTA on a wide range of challenging tasks such as instruction following, reasoning, math, and summarization. LoTA obtains better performance than full fine-tuning and low-rank adaptation (LoRA), and maintains good performance even after training on other tasks – thus, avoiding catastrophic forgetting. By extracting and fine-tuning over *lottery tickets* (or *sparse task vectors*), LoTA also enables model merging over highly dissimilar tasks.

1 INTRODUCTION

Large language models (LLMs) (Brown et al., 2020) have seen an explosion of applications to real-world problems (OpenAI, 2023; Team et al., 2023) via adaptation (Ouyang et al., 2022) to new tasks. Three major *multi-task adaptation* paradigms have emerged: storing and loading task-specific *adapters* (Hu et al., 2022; Beck et al., 2021), continuing to train instruction-tuned models on new tasks in serial via *sequential training* (Ouyang et al., 2022), and combining the adaptations to tasks learned in parallel via *model merging* (Ilharco et al., 2022). Each paradigm has its own associated challenges, such as catastrophic forgetting during sequential training (McCloskey & Cohen, 1989; Dong et al., 2023; Ramasesh et al., 2022; Luo et al., 2023; Wang et al., 2024), and methods that have been proposed to mitigate these challenges (Crawshaw, 2020; Zhang & Yang, 2021). In this work, we propose a new LLM adaptation method, called **Lottery Ticket Adaptation (LoTA)**, that (1) provides sparse adaptation by freezing a majority of the parameters and updating only a sparse subnetwork of the base model and (2) resolves the challenges in common *multi-task adaptation* paradigms. (More details in Section 3.) We summarize our contributions:

- We train *lottery tickets* (or *sparse task vectors*) that can be stored efficiently and obtain performance similar to full fine-tuning (FFT) and higher than LoRA across a range of tasks spanning reasoning, math, code generation, and instruction following. When adapting Mistral for instruction following, FFT and LoTA both get a length-controlled AlpacaEval 2 winrate (Dubois et al., 2024) (how often GPT-4 prefers the outputs of our model over its own) of 19.0%, but LoRA only gets a winrate of 15.3% after extensive hyperparameter tuning.
- We apply LoTA to mitigate **catastrophic forgetting** (McCloskey & Cohen, 1989) of earlier tasks, enabling sequential adaptation to new tasks. If we train models with FFT or LoRA on first GSM8k and then Commonsense Reasoning tasks, they forget GSM8k completely ($< 5\%$) accuracy, but by using LoTA we maintain 57% accuracy on GSM8k and get 85% on Reasoning.
- We can use LoTA to **merge models in parallel** (Wortsman et al., 2022; Jin et al., 2022; Zhang et al., 2023a) across dramatically different tasks. LoTA achieves better performance than existing merging methods that rely on post hoc sparsification (Yadav et al., 2023) (which degrades performance for FFT models), because LoTA naturally trains *sparse task vectors*. When we use LoTA to merge multiple models trained on heterogeneous tasks of instruction following, math and summarization, we get a task-average performance of 21.9% where a merge of FFT models obtains 19.1%.

2 Background: Multi-Task Adaptation

In this section, we go over common multi-task adaptation paradigms and discuss the challenges existing fine-tuning methods, such as FFT and LoRA, bring in each paradigm.

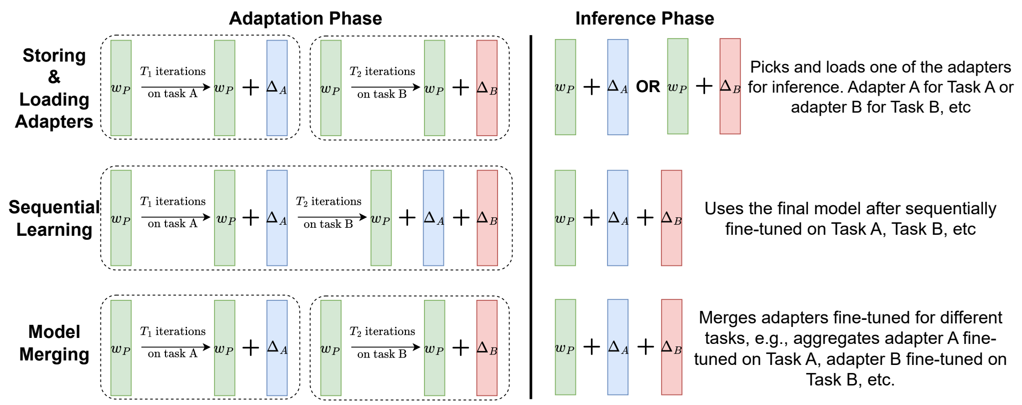
Adaptation Phase
Inference Phase
Storing & Loading Adapters
Sequential Learning
Model Merging
T_1 iterations on task A
T_2 iterations on task B
w_P
Δ_A
Δ_B
OR
Picks and loads one of the adapters for inference. Adapter A for Task A or adapter B for Task B, etc
Uses the final model after sequentially fine-tuned on Task A, Task B, etc
Merges adapters fine-tuned for different tasks, e.g., aggregates adapter A fine-tuned on Task A, adapter B fine-tuned on Task B, etc.

Figure 1: Multi-task adaptation: storing and loading adapters, sequential training, model merging.

Storing and Loading Adapters. The first row of Figure 1 shows training task-specific adapters that are loaded during inference (Ostapenko et al., 2024; Beck et al., 2021; Mangrulkar et al., 2022), depending on the need (Houlsby et al., 2019). Notably, Apple stores and loads adapters to enable their on-device models (Apple, 2024). While using adapters avoids any interference between tasks, it increases memory and compute costs as it requires storing and loading task-specific adapters. To mitigate these costs, a number of parameter-efficient fine-tuning (PEFT) methods have been developed, most popularly LoRA (Hu et al., 2022) (training adapters in the low-rank space). Services such as Punica and S-LoRA allow developers to use this approach to serve large numbers of task-specific adapters for specific requests (Chen et al., 2023; Sheng et al., 2023). However, a persistent gap in capacity between PEFT methods and FFT has presented a tradeoff between adapter overhead and performance (Hu et al., 2022; Liu et al., 2024b; Kopiczko et al., 2023; Biderman et al., 2024; Nikdan et al., 2024).

Sequential Training. When we want one model with multi-task abilities, we can fine-tune the model on different tasks sequentially (Ruder, 2017), e.g., first fine-tune on task A, then fine-tune on task B, as in the second row of Figure 1. We can also train the model on a mixture of all tasks, which may require optimizing the data mixture. Fine-tuning the LLM for new tasks with FFT or existing PEFT methods leads to catastrophic forgetting of earlier tasks. This is problematic, especially for *safety* alignment, since we can fine-tune an LLM to be safe but later get this feature erased during future fine-tuning on new tasks (Lermen et al., 2023). In fact, a number of works have aimed to mitigate this vulnerability (McCloskey & Cohen, 1989; Dong et al., 2023; Ramasesh et al., 2022; Luo et al., 2023; Wang et al., 2024), leaving an open research question: *Can model developers allow users to finetune their aligned models on custom datasets while retaining safety?*

Model Merging. The third row of Figure 1 shows how we can merge multiple models adapted to different individual tasks to have a single model adapted to multiple tasks, by aggregating *task vectors* (or adapters) of different tasks. Existing model merging techniques either require post-processing the task vectors through sparsification (Yu et al., 2023; Davari & Belilovsky, 2023; Yadav et al., 2023), degrading the performance on the task, and/or require extensive hyperparameter tuning for a weighted aggregation of task vectors (Matena & Raffel, 2022; Xiao et al., 2023b).

Each paradigm of multi-task learning poses different challenges, and different methods have been proposed to address these challenges. We defer the more in-depth analysis of these proposed methods in Appendix B because of the sheer quantity of related work that must be covered. The number of methods itself poses challenges for studying their drawbacks, especially when these methods are adopted in settings orthogonal to those they were originally developed for (i.e., LoRA leading to catastrophic forgetting of safety alignment (Lermen et al., 2023)). *Rather than proposing tailored solutions for each paradigm of multi-task adaptation, we want to propose a simple algorithm that can serve as an effective foundation across all three paradigms and evaluate it on a wide range of challenging tasks.*

3 Lottery Ticket Adaptation (LoTA)

The desiderata for each multi-task adaptation paradigm motivates the design of our method. For *adapters*, we want a representation that can be easily compressed for memory efficiency. For *sequential training*, we want a representation that minimizes destructive interference between the previously learned tasks and tasks to be learned in the future. For *model merging*, we want representations that are mutually sparse with each other in parameter space to again prevent destructive interference. We now propose LoTA, a single method that enjoys all these features. We first describe the workflow of LoTA, then revisit the problems each multi-task paradigm faces and discuss how and why LoTA successfully mitigates them.

Algorithm 1 Lottery Ticket Adaptation (LoTA)

```
Require: Adaptation algorithm $\mathbb{A}$, alignment dataset $\mathbb{D}$, pre-trained weights $w_P$, sparsity ratio $s$, learning rate $\eta$, number of calibration iterations $\mathcal{T}$, number of sparse training iterations $T$.
 1: Mask Calibration:
 2: $w_F \leftarrow w_P$
 3: for $\tau \in 0, \dots, \mathcal{T}$ do
 4:     $\nabla = \mathbb{A}_{\mathbb{D}}(w_F)$ {Compute gradient for weights}
 5:     $w_F = w_F - \eta \cdot \nabla$ {Update the model}
 6: end for
 7: Mask Extraction:
 8: $\Delta = w_F - w_p$ {Find the task vector}
 9: $m = \text{Sparsify}(\Delta, s)$ {Create the sparsity mask by thresholding the task vector based on magnitude}
10: Sparse Adaptation:
11: $w \leftarrow w_P$
12: for $t \in \{0, \dots, T\}$ do
13:     $\nabla = \mathbb{A}_{\mathbb{D}}(w)$ {Compute gradient for weights}
14:     $\hat{\nabla} = \nabla \odot m$ {Apply sparse mask to gradient}
15:     $w = w - \eta \cdot \hat{\nabla}$ {Update the model}
16: end for
17: $\hat{w}_F \leftarrow w$
output $\hat{w}_F$
```

Lottery Ticket Adaptation (LoTA). LoTA works in three phases as summarized in Figure 2: (1) mask calibration, (2) mask extraction, (3) sparse adaptation. In the mask calibration phase of LoTA, a base model with parameters w_P is fine-tuned for $\mathcal{T}$ iterations, yielding a fine-tuned model with parameters w_F. Then, in the mask extraction phase, LoTA extracts a sparsity mask m from the task vector $\Delta = w_F - w_P$ based on the magnitude of the updates in Δ. $\mathcal{T}$ could be as small as one iteration. In the sparse adaptation phase of LoTA, the model is first reset to its original state with weights w_P. Then the subnetwork $w_P \odot m$ is fine-tuned for T iterations, while leaving the remaining parameters $w_P \odot (1 - m)$ frozen at their initial values. We summarize the workflow of LoTA in Figure 2 and Algorithm 1 further.

By confining the adaptation updates within subnetworks (identified by m), **LoTA is able to mitigate destructive interference**, e.g., adaptation loss during fine-tuning on future datasets or model merging, that FFT and LoRA suffer from. We discuss this in more detail under three multi-task adaptation paradigms below and provide empirical comparisons with FFT and LoRA in Section 5.

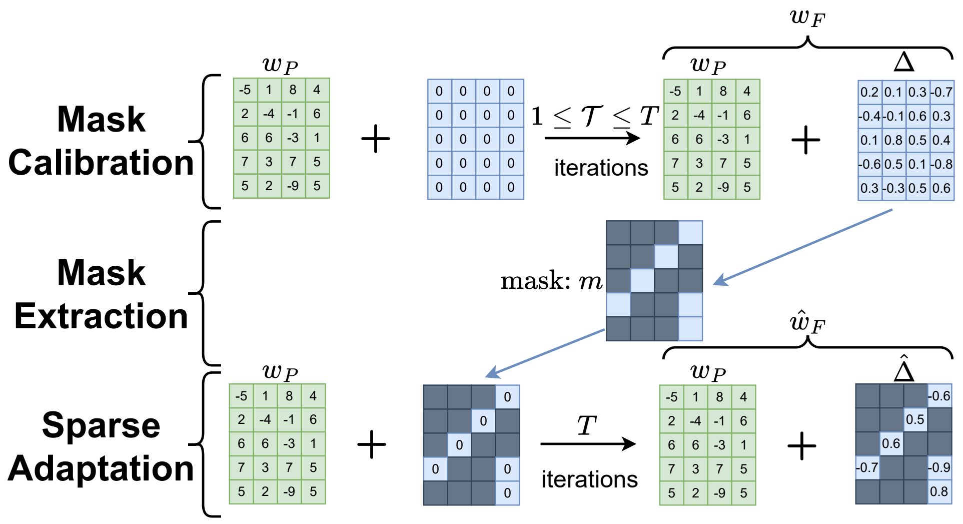

Figure 2: Lottery Ticket Adaptation (LoTA): (1) Mask calibration via FFT for $\mathcal{T}$ iterations, (2) Extracting the sparsity mask m from the task vector Δ, (3) Restarting from the pretrained model and doing sparse fine-tuning with sparsity mask m for T iterations to get the final LoTA model.

(1) Storing & Loading Compressible Adapters Without Compromising Performance. LoRA restricts the adaptation updates to have a low rank, which may be insufficient for complex downstream tasks (Nikdan et al., 2024; Biderman et al., 2024). In parallel, recent work (Isik et al., 2023) on compressing the delta between the fine-tuned and pre-trained model $\Delta = w_F - w_P$ suggests that FFT updates are highly compressible through a simple magnitude-based sparsification. LoTA exploits this underlying sparsity during fine-tuning and obtains better performance than LoRA, while requiring fewer parameters to be trained. As we will show, LoTA adapters can be stored efficiently because of the small number of parameters that need to be updated. LoTA also requires fewer parameters to be updated in the model when the adapter is applied to the model, so loading a LoTA adapter can be faster than loading a LoRA adapter.

(2) Preventing Catastrophic Forgetting During Sequential Training by Mitigating Destructive Interference. Without loss of generality, suppose we have two tasks, Task A and Task B, and that after training on Task A our model performs well on Task A but not on Task B, and after training the same model on Task B our model performs well on Task B but not on Task A. We believe one underlying cause for this *catastrophic forgetting* is that parameters containing important information for a task are updated in other –almost certainly orthogonal– directions during continued training; we refer to this as *destructive interference*. A natural method to mitigate destructive interference is to ensure that the model updates disjoint sets of parameters for Task A and Task B. To enhance the robustness against destructive interference in sequential training, we propose **Lottery Ticket Together Optimization (LoTTO)** which learns mutually sparse (i.e., non-overlapping) masks for sequentially learned tasks.

Algorithm 2 Lottery Ticket Together Optimization (LoTTO)

```
Require: Alignment dataset sequence $\mathbb{D}_{i=1}^N$, pre-trained weights $w_P$, LoTA hyperparameter set $\mathbb{H}$.
1: $\mathbb{C} = 0$ {Initialize constraint set}
2: $w_i = w_P$
3: for $\mathbb{D} \in \mathbb{D}_{i=1}^N$ do
4:     $m_i^C = C$ {Create mask from constraint set}
5:     $w_i^C = w_i - \nabla \odot m_i^C$ {Train model with constraint mask}
6:     $m_i^S = \text{Sparsify}(w_i^C - w_i)$ {Extract sparse mask from trained model}
7:     $w_i^F = w_i - \nabla \odot m_i^S$ {Train model with sparse mask}
8:     $\mathbb{C} = \mathbb{C} \cup m_i^S$ {Update constraints}
9: end for
output $\hat{w}_N^F$
```

Mitigating Destructive Interference via Lottery Ticket Together Optimization (LoTTO). Suppose that we have already learned Task A with LoTA. LoTTO calibrates a sparsity mask for Task B by first training a model where the only weights that can be updated are those that are *not* updated when running LoTA on Task A, and then using a sparse set of those weights to train the final model. In doing so, LoTTO ensures that the weights updated for Task A are *disjoint* from the weights updated for Task B. LoTTO enables learning new tasks without forgetting old ones, by restricting weight adaptations to not interfere with the weights that were important for previous adaptations, therefore mitigating destructive interference. This procedure can be applied inductively to enable sequential adaptation to multiple tasks, so that a model developer seeking to adapt a model adapted with LoTA (potentially on several tasks), just needs to ensure that they do not update the task vector with respect to the base model. LoTTO can be applied to capabilities one at a time; first the model learns math, then coding, then reasoning, then planning, etc. because each of these capabilities benefits from prior knowledge. We train on up to 4 tasks sequentially, but optimistically if we believe we can update 1% of the weights for each capability, then LoTTO can fit at most 100 capabilities. When the subnetworks for different tasks overlap, we do not expect that LoTTO will result in any worse forgetting than LoTA, which we will show itself has better robustness to forgetting than FFT. We will also show that LoTTO on sequential tasks outperforms the common practice of training on a mixture of tasks.

(3) Model Merging Across Heterogeneous Tasks. Existing model merging methods have been shown to merge models trained on relatively similar language datasets (Wortsman et al., 2022; Jin et al., 2022; Zhang et al., 2023a), typically via a post hoc sparsification (Yadav et al., 2023; Yu et al., 2023; Davari & Belilovsky, 2023) of the task vectors. This ensures the task vectors are disjoint – hence limiting destructive interference – but also degrades the performance of each individual task. LoTA enforces sparsity during fine-tuning and directly trains *sparse task vectors*, obviating the need for post hoc sparsification. LoTA's advantage over methods such as TIES (Yadav et al., 2023) is similar to the advantage of training a model with an intelligently chosen sparsity constraint throughout training (as in the original lottery ticket hypothesis (Frankle & Carbin, 2019)) over pruning parameters after training.

4 Experimental Setup

In this section, we provide details of the experimental setup, including the baselines, model, dataset, and metric selection. We present the results later in Section 5. We share our anonymized code. We are limited to an academic computing budget, and all results are conducted with a single A100 GPU. We typically do $1-3$ epochs of training for each dataset as this is a standard choice in LLM fine-tuning. We use the RMSProp optimizer with default hyperparameters.

Baselines & Hyperparameters. Across all three multi-task adaptation paradigms, we compare LoTA against FFT and LoRA. We extensively tune the hyperparameters for FFT and LoRA to ensure that we are comparing against strong baselines. We do not tune hyperparameters for LoTA and directly transfer the hyperparameters from FFT. We fix the sparsity ratio hyperparameter in LoTA to 90%.

Models. We use two commonly-used open-weights model families, Mistral Jiang et al. (2023) and Llama 3 (AI@Meta, 2024; Touvron et al., 2023), specifically Mistral-7B and Llama-3-8B – the largest models we can adapt with FFT on a single GPU.

Tasks. We consider six capabilities: instruction following, safety, math, coding, summarization, and reasoning. We now briefly discuss each capability, the datasets we use to fine-tune and evaluate the presented methods, and the motivation behind the choices.

Instruction Following. The most widely-used instruction-tuned LLMs are the "Instruct" or "chat" versions of base models, such as Llama-3-8B-Instruct (AI@Meta, 2024). This is because the process of tuning models on human instructions aligns models to human preferences across a range of tasks (Ouyang et al., 2022). For this, we adapt models to data from UltraFeedback (Cui et al., 2023), which contains a mixture of datasets covering truthfulness, honesty, and helpfulness in addition to instruction-following. We measure the instruction following ability by length-controlled AlpacaEval2 Win Rate (Li et al., 2023), which we refer to as "winrate". A high winrate means that GPT-4 (OpenAI, 2023) prefers the responses of our model on a set of representative prompts over its own responses. Winrate is the metric most closely correlated with human rating preference (Dubois et al., 2024). Another common benchmark for "chat" models is MT-Bench (Zheng et al., 2023), but there is a significant degree of data contamination between MT-Bench and other task-specific training datasets (Yu et al., 2024)–hence, we do not evaluate on MT-Bench.

Reasoning. We train on the standard set of 8 commonsense reasoning tasks (Christopher et al., 2019; Bisk et al., 2019; Sap et al., 2019; Zellers et al., 2019; ai2, 2019; Clark et al., 2018; Mihaylov et al., 2018) (Boolq, PIQA, SocialIQA, Hellaswag, Winograde, ARC-easy, ARC-challenge, OpenBookQA) and report the exact-match accuracy on the test set. As a representative task, we use ARC-easy.

Math. We use the set of 9 math instruction datasets from (Yue et al., 2023) for fine-tuning and report performance on the test set of GSM8k (Cobbe et al., 2021). When only considering a single task, we choose GSM8k as the representative task as it is commonly used as a single training and test task by other papers.

Code Generation. We use data that instructs the model to write SQL queries given some context (b mc2, 2023)(SQL-create-context) and report the ROUGE-1 F1 score (Lin, 2004) on the test set.

Summarization. We use Samsum (Gliwa et al., 2019), reporting ROUGE-1 F1 score on the test set.

Safety. A recent concern in AI policy is that, while frontier models such as GPT-3.5/GPT-4 are aligned, they can also be fine-tuned and this presents an opportunity to misalign them. Recently, Qi et al. (2023) show that by fine-tuning GPT-3.5 on just 100 harmful examples for a few epochs, they can ask it to answer harmful queries that it ordinarily would refuse, and Zhan et al. (2024) show the same for GPT-4. Lermen et al. (2023) show that this can be done with LoRA rather than the fine-tuning method OpenAI are using in their fine-tuning API (presumably FFT). We evaluate the safety of our models on HEx-Phi (Qi et al., 2023), a dataset of 330 questions spanning multiple categories such as malware, fraud, etc. The safety score is the percentage of harmful queries that the model refuses to respond. Aligned models (Llama-3-Instruct) will refuse 100% of harmful queries, but because we are aligning base models ourselves, our baseline Instruct model only gets 93%. Because we are interested in measuring the *forgetting* of safety alignment, we do not see this as a major limitation.

Roadmap of the Evaluation. We vary hyperparameters (Section 5), consider single-task adapters (Section 5.1), sequential training (Section 5.2), and model merging (Section 5.3).

5 Experimental Results

We first evaluate the impact of varying the sparsity and mask calibration.

Sparsity. We vary the sparsity parameter in LoTA in Table 1. Multiple sparsity thresholds work well; some values with less sparsity (30%) seem to negatively impact performance and extreme sparsity seems to improve performance 99% but both deviations are within 2% of FFT.

Table 1: The impact of varying the sparsity of LoTA. GSM8k finetuning with Mistral-7B.

Sparsity	0% (FFT)	10%	20%	30%	40%	50%	60%	70%	80%	90%	95%	99%
Performance	59.8	60.5	60.2	57.9	57.8	59.4	59.1	59.2	60.7	60.4	60.4	$62.0_{0.3}$

Table 2: LoTA performance on GSM8k degrades gracefully as a smaller fraction of the data is used to calibrate a 90% sparse mask. 0 data usage corresponds to creating a random mask.

Data Used	100%	10%	1%	0.1%	0
Accuracy (GSM8k)	$60.4_{1.3}$	$60.1_{1.1}$	$60.0_{1.7}$	$59.2_{1.3}$	54.0

Calibrating the LoTA Mask Efficiently. In Table 2, we consider how the performance drops if we calibrate a 90% sparse mask on a fraction of the overall GSM8k dataset, including the baseline where we use a random mask which corresponds to 0% data used. We find that a 90%-sparse mask can achieve good performance on GSM8k with just a single step of calibration (calibration dataset size= 0.1% of the full dataset). Furthermore, instruction tuning datasets are generally quite small (finetuning on GSM8k takes less than an hour on a single GPU). Given that the mask can be efficiently calibrated, transferred from other datasets (as we will show), or in the worst case, calibrated on the entire dataset in at most a few hours, we do not anticipate that the compute overhead of LoTA will be a major limitation of the method.

Next, we compare LoTA with baselines, FFT and LoRA, across all three paradigms of multi-task adaptation. We run all the methods for the same number of steps, i.e., the calibration time for LoTA is factored in. Unless stated otherwise, LoTA has 90% sparsity where the mask is calibrated by training for 1% of the adaptation dataset. We choose this sparsity level rather than the 99% sparsity level that sometimes outperforms FFT because we can compare the improvements of LoTA that are due to the method's mitigation of destructive interference, rather than any inherent advantage sparsity may have.

5.1 Adapting to a Single Task

We first fit an adapter to each dataset starting from a pre-trained base model, i.e., one adapter per task. In Table 3, we find that LoTA outperforms LoRA and performs similarly to FFT. Although LoRA is able to achieve similar performance to LoTA on the easier tasks, such as SQL, Samsum, there is a clear gap in performance on the more challenging tasks, such as Instruction Following and GSM8k. LoTA consistently recovers the performance of FFT.

Table 3: Performance comparison of Full Finetuning (FFT), LoRA and LoTA on single-task datasets for 3 epochs. We report the winrate on instruction following, the accuracy of exact match on the reasoning and math tasks, and the ROUGE-1 score on the SQL generation and summarization tasks. LoTA outperforms LoRA on the challenging tasks of instruction following, reasoning, and math, obtaining comparable performance to FFT. **bold**: best method, underline: second best method.

Model	Method	Instruction Following	Reasoning	GSM8k	SQL	Summarization
Mistral	FFT	$\underline{19.0_{0.98}}$	85.8	$\underline{59.8_{1.0}}$	$98.9_{0.1}$	$52.0_{0.2}$
	LoRA	$15.3_{0.8}$	$\underline{86.8}$	$58.3_{1.1}$	$98.9_{0.1}$	$\underline{52.9_{0.3}}$
	LoTA (ours)	$\mathbf{19.0_{0.7}}$	$\mathbf{87.5}$	$\mathbf{60.4_{1.1}}$	$\mathbf{98.9_{0.1}}$	$\mathbf{52.9_{0.2}}$
Llama 3	FFT	$\underline{17.61_{0.8}}$	$\mathbf{84.8}$	$\underline{63.4_{0.1}}$	$\mathbf{99.4_{0.1}}$	$\mathbf{53.6_{1.9}}$
	LoRA	$14.2_{0.8}$	84.1	$62.3_{0.4}$	$98.7_{0.1}$	$\underline{52.3_{0.2}}$
	LoTA (ours)	$\mathbf{18.0_{0.7}}$	$\underline{84.4}$	$\mathbf{64.2_{0.7}}$	$\underline{99.0_{0.1}}$	$\underline{52.3_{0.3}}$

5.2 Sequential Training

During sequential training, a model is first adapted to one capability (Task A) and then to another capability (Task B) (such as when creating a specialized model for math) or a set of capabilities (the most common setting for open-sourced fine-tunes of frontier models). The main challenges we seek to mitigate are (1) catastrophic forgetting of Task A and (2) the inability to adapt to Task B.

Table 4: Comparison of FFT and LoRA baselines with our LoTTO method, which calibrates masks on Task A (GSM8k) and Task B (Commonsense), where "Mask B" is disjoint from Mask A. Alternating freeze applies to odd and even layers, respectively. The model is Mistral7B.

Method	Task A	Task A Params	Task B	Task B Params
FFT (baseline)	59.8	-	-	-
LoRA (baseline)	58.3	-	-	-
FFT	2.3	All	86.4	All
LoRA	4.2	All	86.3	All
FFT	2.4	Odd Layers	86.15	Even Layers
LoRA	47.26	Odd Layers	84.9	Even Layers
LoTTO (ours)	57.81	Mask A	85.12	Mask B

LoTTO Mitigates Destructive Interference. Table 4 compares LoTTO to FFT and LoRA when training first on GSM8k and then on Reasoning; LoTTO obtains the best performance across both tasks because it mitigates the destructive interference that leads to catastrophic forgetting. We now do a detailed comparison to LoRA, proposing a new method in our effort to compare LoTTO to the strongest possible baselines.

LoRA Does Not Significantly Reduce Catastrophic Forgetting of Capabilities Learned During Finetuning. First, note that LoRA loses nearly all performance on Task A when training on Task B. This observation may seem contrary to the results in (Biderman et al., 2024). However, they measure forgetting as the degradation in performance on standard benchmark tasks between the pretrained model and the finetuned model, whereas we measure forgetting as the degradation on Task A between the model finetuned on Task A and the model finetuned on first Task A and then Task B. We speculate that the discrepancy is that knowledge learned through the pretraining process may be far more persistent than knowledge learned through finetuning. Indeed, LoRA *does* forget less than FFT –just not enough to make a difference when training with LoRA on both tasks.

LoRA Can Also Mitigate Destructive Interference, But Not As Well as LoTTO. A key insight into the design of LoTTO is that we can mitigate destructive interference between different tasks by forcing the tasks to update disjoint sets of parameters. LoTTO carefully calibrates the disjoint subsets by pruning the weights that were updated the most after a small amount of finetuning on the dataset. We can also extend this insight to try and mitigate destructive interference in FFT and LoRA. We do this by proposing an "alternating freeze" strategy, where the odd layers can only be trained on Task A and the even layers can only be trained on Task B. LoRA already struggles to fit complex tasks such as math (Task A is GSM8k) and the alternating freeze further restricts LoRA's representative capacity. Therefore, while the alternating freeze does forget less, and on easy tasks such as commonsense reasoning, there is no significant gap between LoRA and LoTA, the gap on the challenging Task A is hard to overcome because it is partially due to the inherent gap in performance between LoRA and LoTA and partially due to LoTA's limited forgetting as compared to LoRA. We note that LoTA can also be combined with LoRA, by fixing the A matrix in LoRA as a random projection (Zhang et al., 2023b) and tuning the B matrix with LoTA; this could be an interesting piece of future work.

Sequential Training Under LoTTO Outperforms Mixing Datasets Together. We compare our LoTTO method, that trains disjoint LoTA adapters sequentially on each dataset, to FFT on a mixture of Instruction Following and the dataset noted in Task B. LoTTO under sequential training consistently outperforms the common practice of mixing all datasets together and doing FFT. We note that LoTA can also be used in the setting of mixing all datasets together, by routing each sample of a given dataset to a GPU –specific to that dataset– that applies the corresponding LoTA mask for that dataset.

Table 5: We compare our LoTTO method, that trains disjoint LoTA adapters sequentially on each dataset, to FFT on a mixture of Instruction Following and the dataset noted in Task B. LoTTO under sequential training consistently outperforms the common practice of mixing all datasets together and doing FFT.

Task B	Method	Utility of Task A (Drop)	Utility of Task B (Drop)
GSM8k	FFT-Mixing (baseline)	15.2 (3.8)	54.5 (5.3)
	LoTTO (ours)	17.8 (1.2)	59.1 (0.7)
Reasoning	FFT-Mixing (baseline)	10.7 (8.3)	81.3 (4.5)
	LoTTO (ours)	16.5 (2.5)	83.7 (2.1)
GSM8k+Arc+SQL	FFT-Mixing (baseline)	13.8 (5.2)	73.6 (4.4)
	LoTTO (ours)	15.9 (3.1)	73.8 (3.2)

Mitigating Catastrophic Forgetting While Enabling Adaptation to Downstream Datasets. In Table 6, we consider a range of method combinations for a simplified setting where we seek to adapt an Instruct model to Math data without catastrophic forgetting. We will go row-by-row through the table and analyze each set of results. Even when training on just a single, relatively small dataset, FFT in both phases suffers a significant drop in winrate. An easy way to mitigate this is to simply train the initial Instruct model with LoTA. Following this, FFT on GSM8k does not significantly reduce winrate. However, it does present a potentially unwelcome tradeoff in task accuracy. For this, we turn to LoTTO, which achieves the best performance across both tasks.

Table 6: Sequential learning first on Task A (Instruction Following) then on Task B (varied). Fine-tuning on Tasks A and B is performed using both FFT and LoTA. The utility of each task is computed after fine-tuning on Task B is completed. Note that there is no utility when we fine-tune on harmful data to evaluate the catastrophic forgetting of Safety, and the baseline is the safety score of the Instruct model. FT=Fine-tuning. We reproduce the baseline of doing FFT on each method independently as reported in Table 3 for convenience; note that on the reasoning task LoTA outperforms FFT. **bold**: best method, underline second-best method; we do not report second-best when only two methods are presented. All results are with Mistral. We reuse the same mask for LoTTO calibrated on GSM8k for MathInstruct, Reasoning, GSM8k+Arc+SQL and Safety.

Task B	Method on Task A	Method on Task B	Utility of Task A (Drop)	Utility of Task B (Drop)
Instruction Following	Baseline	-	19.0 (-)	-
GSM8k	-	Baseline	-	59.8 (-)
	FFT	FFT	15.2 (3.8)	58.3 (1.5)
	LoTA (ours)	FFT	17.7 (1.3)	58.7 (1.1)
	FFT	LoTA (ours)	15.9 (3.1)	54.2 (5.6)
	LoTA (ours)	LoTTO (ours)	**17.8 (1.2)**	**59.1 (0.7)**
	FFT	LoRA	14.1 (4.2)	55.5 (4.9)
	LoRA	LoRA	13.7 (5.3)	58.8 (1.0)
	FFT	FFT (Replay)	16.3 (2.7)	55.5 (4.3)
MathInstruct	-	Baseline	-	56.7 (-)
	FFT	FFT	$14.2_{0.8}$ (4.8)	$51.3_{0.2}$ (5.4)
	LoTA (ours)	LoTA (ours)	**16.0**$_{0.7}$ (-3.0)	**55.5**$_{0.1}$ (1.2)
Reasoning	-	Baseline	-	85.8 (-)
	FFT	FFT	$0.2_{0.1}$ (18.8)	82.3 (3.5)
	LoTA (ours)	LoTTO (ours)	**16.5**$_{0.9}$ (2.5)	**83.7** (2.1)
GSM8k+Arc+SQL	-	Baseline	-	77.0
	FFT	FFT	$0.5_{0.2}$ (18.6)	75.0 (2.0)
	LoTA (ours)	FFT	$11.5_{0.7}$ (7.5)	**75.4 (1.6)**
	LoTA (ours)	LoTTO (ours)	**15.9**$_{0.9}$ (3.1)	73.8 (3.2)
Safety	Baseline	-	93.1 (-)	-
	FFT	FFT	$19.1_{3.5}$ (73.9)	-
	LoTA (ours)	LoTTO (ours)	**63.4**$_{2.2}$ (29.7)	-

How Much Does Data Reuse Help? The simplest and arguably most performant method from prior work that we found for mitigating catastrophic forgetting is simply to replay in some in-distribution data from Task A. This is the line marked with "FFT (Replay)", and it does mitigate forgetting on Task A, but at the cost of performance on Task B. We ablate the amount of data from Task A to be used between $1 - 100\%$ of the dataset size of the data from Task B, and this is the best result in terms of mitigating forgetting on Task A. As we mix in less and less data from Task A, this method approaches just doing FFT sequentially in performance, so we omit those results for brevity.

Adapting Sequentially to Multiple Tasks with LoTTO. We now consider the more challenging setting when we need to adapt to *multiple* tasks without catastrophic forgetting while still obtaining good performance on those tasks. In Table 6, we adapt an Instruct model to a mix of reasoning, math, and SQL data and find a surprising result; the FFT Instruct model collapses almost completely to a winrate of less than half a percent. As we showed in Table 6, this can be mitigated by mixing in more instruction following data, albeit at a cost. The LoTA Instruct model still degrades in performance ($19 \rightarrow 11$) but nowhere near as much. In the line marked "LoTTO", we instead adapt the LoTA Instruct model to the downstream tasks using the mask calibrated from GSM8k with LoTTO. Applying LoTTO to make the downstream adaptation be mutually sparse with the initial LoTA Instruct model increases performance significantly on instruction following and math. Performance on Arc and SQL suffers somewhat because our mask does not consider those tasks, but this means that our mask is much cheaper to calibrate and is, in fact, generalizable not only across tasks within a domain (as shown in Table 6) but also *across* domains.

Mitigating Catastrophic Forgetting of Safety Alignment. In the "Safety" row of Table 6, we consider fine-tuning the Mistral Instruct model we trained ourselves on the 100 harmful instructions from (Qi et al., 2023). The baseline model gets a score of 93% and training with FFT quickly degrades safety. Adapting the LoTA Instruct with LoTTO (again, with the LoTTO mask calibrated from GSM8k) mitigates this safety drop significantly, even though our LoTTO mask was calibrated on an extremely different dataset. Therefore, a potential mitigation would be for an entity providing a fine-tuning API such as OpenAI to do the safety training with LoTA, calibrate the LoTTO mask on a utility dataset, and then do fine-tuning on their client's dataset with LoTTO. We do not intend to present our method as an active defense against fine-tuning attacks; although finetuning with LoTTO improves safety, given sufficient data and access to the model weights, any attacker can of course undo safety tuning entirely. However, catastrophic forgetting of safety alignment is an important problem with real-world applications, and we find it compelling that our method can mitigate this.

5.3 LoTA for Model Merging

We now consider the setting of model merging, where we train models on disjoint datasets fully in parallel and then merge together the task vectors with the goal of producing a model with good performance on multiple tasks. Prior work in model merging mostly considers merging similar datasets, such as the commonsense reasoning datasets, but it is relatively easy to merge models when the datasets are similar and becomes increasingly hard as the datasets become more heterogeneous due to the gradient mismatch (Daheim et al., 2024). We consider merging models trained on heterogeneous datasets.

Merging Models. We use TIES-Merging (Yadav et al., 2023) as a baseline to merge models trained on heterogeneous tasks, because it outperformed other merging methods in our testing. TIES performs post-hoc sparsification on each task vector and requires a 2-D hyperparameter search for this quantity, which we perform for the merge of FFT models. Naturally, we could optimize the performance of Task A by fully sparsifying Task B, and vice versa; we report the result that achieves good performance on Task B while maintaining some performance on Task A, and report the full range of hyperparameters in Appendix A. LoTA is inherently sparse, so when we merge a LoTA model with an FFT model we do not need to perform hyperparameter search on the LoTA model, and we use the same level of sparsity for the FFT model that we obtained when merging together two FFT models. When merging two LoTA models together, no hyperparameter search is required at all as both models are inherently sparse. We could in theory sparsify the LoTA models beyond their existing levels with post-hoc sparsification, but we do not tune this hyperparameter.

Challenge of Overlapping Sparsity in Model Merging. In the sequential training paradigm, we exploited the fact that masks for different tasks have a significant overlap in order to generalize our

LoTTO mask calibrated on GSM8k to provide robustness to forgetting across a range of other tasks. However, this same phenomenon of overlapping sparsity presents a challenge in the model merging setting. The challenge is that because the merging is parallel, we cannot use LoTTO to calibrate the masks to be disjointly sparse as we did in the sequential training setting. One alternative is to enforce random disjoint masks for each task, corresponding to "0% Data Used" in Table 2.

Table 7: Model merging using TIES of Task A, Instruction Following, and Task B, (varied; GSM8k+Samsum means we are merging 3 models). Fine-tuning on Tasks A and B is performed using both full fine-tuning (FFT) and LoTA. The utility of each task is computed after merging the two task vectors. **bold**: best result, $\underline{\text{underline}}$: second best result. We reproduce the baseline for each task on FFT from Table 3 for convenience; note again that for some tasks the LoTA baseline outperforms the FFT baseline. All results are with Mistral.

Task B	Method on Task A	Method on Task B	Utility of Task A (Drop)	Utility of Task B (Drop)
Instruction Following	Baseline	-	19.0 (-)	-
GSM8k+Samsum	-	Baseline	-	55.9 (-)
	FFT	FFT	$7.6_{0.6}$ (11.4)	$49.7_{1.1}$ (6.2)
	LoRA	LoRA	$8.5_{0.5}$ (10.5)	$44.3_{1.0}$ (11.6)
	LoTA (ours)	FFT	$8.9_{0.6}$ (10.1)	$\underline{50.7}_{1.1}$ (5.2)
	FFT	LoTA (ours)	$\underline{10.7}_{0.7}$ (8.3)	$\mathbf{55.0}_{1.1}$ (0.9)
	LoTA (ours)	LoTA (ours)	$\mathbf{13.1}_{0.8}$ (5.9)	$46.6_{1.0}$ (9.3)
GSM8k	-	Baseline	-	59.8 (-)
	FFT	FFT	$6.9_{0.5}$ (12.1)	$59.1_{0.1}$ (0.7)
	LoTA (ours)	FFT	$\mathbf{16.1}_{0.8}$ (2.9)	$\mathbf{60.5}_{1.1}$ (+0.7)
	FFT	LoTA (ours)	$14.0_{0.8}$ (5.0)	$\underline{59.3}_{1.0}$ (0.5)
	LoTA (ours)	LoTA (ours)	$\underline{15.1}_{0.8}$ (3.9)	$58.4_{1.1}$ (1.5)
Samsum	-	Baseline	-	52.0 (-)
	FFT	FFT	$8.1_{0.7}$ (10.9)	$47.0_{0.0}$ (5.0)
	LoTA (ours)	FFT	$\underline{13.8}_{0.8}$ (5.2)	$51.8_{0.2}$ (0.2)
	FFT	LoTA (ours)	$10.4_{0.7}$ (8.6)	$\mathbf{53.3}_{0.1}$ (+1.3)
	LoTA (ours)	LoTA (ours)	$\mathbf{15.4}_{0.9}$ (3.6)	$\underline{52.7}_{0.1}$ (0.1)

In Table 7 we merge together models trained on GSM8k and Samsum with the Mistral model that we trained for instruction following. We consider the full combination of merging FFT and LoTA models. The merge of two FFT models performs poorly in all settings on both tasks, indicating that post-hoc sparsification does not perform well for heterogeneous tasks, which is in line with recent model merging theory (Daheim et al., 2024). Merging LoRA models together does not particularly improve performance. The merges that contain LoTA models have better performance across all tasks, but there is no combination that is pareto-optimal across all tasks. One avenue to improve LoTA for model merging may be to assume that each participant has a small sample from other participants' datasets to perform a joint LoTA calibration procedure.

6 Discussion

We propose Lottery Ticket Adaptation (LoTA), a sparse alignment framework that fine-tunes only a sparse subnetwork of the base model, leaving the rest of the parameters frozen. LoTA successfully mitigates destructive interference (a problem with existing fine-tuning methods including FFT and LoRA) in many multi-task adaptation paradigms, prevents catastrophic forgetting of earlier tasks, including safety, and allows for successful model merging of even dramatically different tasks.

Limitations. As mentioned in Section 5, LoTA does not provide the compute efficiency of LoRA. If the adapter needs to be compressed by more than $100\times$, LoTA may not provide sufficient compression. We evaluate on instruction following, reasoning, math, SQL generation, and summarization, yet even more tasks exist such as Python code generation, classification, or long-context question answering. We compare LoTA to baselines (LoRA, TIES) but other PEFT and merging methods exist. In a future revision of this paper, we plan to provide comparisons to a broader range of PEFT (Liu et al., 2024b) and merging (Yu et al., 2023) methods. All LoTA masks are unstructured, so we do not reap the runtime benefits of structured sparsity. We only explore magnitude pruning for LoTA, but many other methods have been proposed in the literature, as we discuss in Appendix B, and may perform better.

Reproducibility Statement. As requested by ICLR, we are providing a brief statement on reproducibility. We provide an anonymous link to our codebase in Appendix A, where all the results in the paper can be reproduced. We made efforts to create strong baselines for ourselves by tuning hyperparameters, whose ranges we report in Appendix A, and by creating what we believe is a new method in the form of the "alternating freeze" baseline. The script "continual learning", for instance, reproduces Table 4 end to end on publicly available datasets. In the main text, we provided details on the datasets we used, all of which are publicly available and in fact are provided in our repository along with the prompts and code we used to train and evaluate on that data, and our reasoning for choosing those datasets. We fine-tuned models that are publicly available on Huggingface, but because we have academic compute restrictions we had to create code that fine-tunes Mistral-7B and Llama-3-8B on a single GPU, which we provided in the repository so that other researchers can more easily reproduce our results (typically, instruction tuning these models requires 8 GPUs, as in the popular Alignment Handbook repository's scripts). We plan to open source our code and upstream our LoTA method to Huggingface to further aid in reproducibility.

References

Winogrande: An adversarial winograd schema challenge at scale. 2019.

AI@Meta. Llama 3 model card. 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 139–154, 2018.

Apple. Apple adapters, 2024. https://machinelearning.apple.com/research/introducing-apple-foundation-models.

b mc2. sql-create-context dataset, 2023. URL https://huggingface.co/datasets/b-mc2/sql-create-context. This dataset was created by modifying data from the following sources: Zhong et al. (2017); Yu et al. (2018).

Tilman Beck, Bela Bohlender, Christina Viehmann, Vincent Hane, Yanik Adamson, Jaber Khuri, Jonas Brossmann, Jonas Pfeiffer, and Iryna Gurevych. Adapterhub playground: Simple and flexible few-shot learning with adapters. *arXiv preprint arXiv:2108.08103*, 2021.

Dan Biderman, Jose Gonzalez Ortiz, Jacob Portes, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. Lora learns less and forgets less, 2024.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Lequn Chen, Zihao Ye, Yongji Wu, Danyang Zhuo, Luis Ceze, and Arvind Krishnamurthy. Punica: Multi-tenant lora serving, 2023.

Clark Christopher, Lee Kenton, Chang Ming-Wei, Kwiatkowski Tom, Collins Michael, and Toutanova Kristina. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *NAACL*, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Michael Crawshaw. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. Ultrafeedback: Boosting language models with high-quality feedback, 2023.

Nico Daheim, Thomas Möllenhoff, Edoardo Ponti, Iryna Gurevych, and Mohammad Emtiyaz Khan. Model merging by uncertainty-based gradient matching. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=D7KJmfEDQP.

MohammadReza Davari and Eugene Belilovsky. Model breadcrumbs: Scaling multi-task model merging with sparse masks. *arXiv preprint arXiv:2312.06795*, 2023.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35: 30318–30332, 2022.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2023a.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*, 2023b.

Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. How abilities in large language models are affected by supervised fine-tuning data composition. *arXiv preprint arXiv:2310.05492*, 2023.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, et al. Loramoe: Revolutionizing mixture of experts for maintaining world knowledge in language model alignment. *arXiv preprint arXiv:2312.09979*, 2023.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024.

Ali Edalati, Marzieh Tahaei, Ivan Kobyzev, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. Krona: Parameter efficient tuning with kronecker adapter. *arXiv preprint arXiv:2212.10650*, 2022.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Pruning neural networks at initialization: Why are we missing the mark? In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ig-VyQc-MLK.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pp. 10323–10337. PMLR, 2023.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. OPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tcbBPnfwxS.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pp. 70–79, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5409. URL https://www.aclweb.org/anthology/D19-5409.

Demi Guo, Alexander M Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. *arXiv preprint arXiv:2012.07463*, 2020.

Han Guo, Philip Greengard, Eric Xing, and Yoon Kim. LQ-loRA: Low-rank plus quantized matrix decomposition for efficient language model finetuning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=xw29VvOMmU.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

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

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5254–5276, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.319. URL https://aclanthology.org/2023.emnlp-main.319.

Yufan Huang, Yanzhe Zhang, Jiaao Chen, Xuezhi Wang, and Diyi Yang. Continual learning for text classification with information disentanglement based regularization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2736–2746, 2021.

Tingfeng Hui, Zhenyu Zhang, Shuohuan Wang, Weiran Xu, Yu Sun, and Hua Wu. Hft: Half fine-tuning for large language models. *arXiv preprint arXiv:2404.18466*, 2024.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Berivan Isik, Hermann Kumbong, Wanyi Ning, Xiaozhe Yao, Sanmi Koyejo, and Ce Zhang. Gpt-zip: Deep compression of finetuned large language models. In *Workshop on Efficient Systems for Foundation Models@ ICML2023*, 2023.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. In *The Eleventh International Conference on Learning Representations*, 2022.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning, 2021.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W Mahoney, and Kurt Keutzer. Squeezellm: Dense-and-sparse quantization. *arXiv preprint arXiv:2306.07629*, 2023.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki Markus Asano. Vera: Vector-based random matrix adaptation. *arXiv preprint arXiv:2310.11454*, 2023.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M Asano. VeRA: Vector-based random matrix adaptation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NjNfLdxr3A.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1VZqjAcYX.

Simon Lermen, Charlie Rogers-Smith, and Jeffrey Ladish. Lora fine-tuning efficiently undoes safety training in llama 2-chat 70b. *arXiv preprint arXiv:2310.20624*, 2023.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.243. URL https://aclanthology.org/2021.emnlp-main.243.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 2023.

Yixiao Li, Yifan Yu, Chen Liang, Nikos Karampatziakis, Pengcheng He, Weizhu Chen, and Tuo Zhao. Loftq: LoRA-fine-tuning-aware quantization for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=LzPWWPAdY4.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965, 2022a.

James Liu, Guangxuan Xiao, Kai Li, Jason D Lee, Song Han, Tri Dao, and Tianle Cai. Bitdelta: Your fine-tune may only be worth one bit. *arXiv preprint arXiv:2402.10193*, 2024a.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*, 2024b.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 61–68, Dublin, Ireland, May 2022b. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.8. URL https://aclanthology.org/2022.acl-short.8.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*, 2023.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft, 2022.

Michael S Matena and Colin A Raffel. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

Mahdi Nikdan, Soroush Tabesh, and Dan Alistarh. Rosa: Accurate parameter-efficient fine-tuning via robust adaptation. *arXiv preprint arXiv:2401.04679*, 2024.

OpenAI. Gpt-4 technical report, 2023.

Oleksiy Ostapenko, Zhan Su, Edoardo Maria Ponti, Laurent Charlin, Nicolas Le Roux, Matheus Pereira, Lucas Caccia, and Alessandro Sordoni. Towards modular llms by building and reusing a library of loras, 2024.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to!, 2023.

Vinay Venkatesh Ramasesh, Aitor Lewkowycz, and Ethan Dyer. Effect of scale on catastrophic forgetting in neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=GhVS8_yPeEa.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. Progressive prompts: Continual learning for language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=UJTgQBc91_.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Alexey Romanov, Anna Rumshisky, Anna Rogers, and David Donahue. Adversarial decomposition of text representation. *arXiv preprint arXiv:1808.09042*, 2018.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions, 2019.

Archit Sharma, Sedrick Keh, Eric Mitchell, Chelsea Finn, Kushal Arora, and Thomas Kollar. A critical evaluation of ai feedback for aligning large language models, 2024.

Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, Joseph E. Gonzalez, and Ion Stoica. S-lora: Serving thousands of concurrent lora adapters, 2023.

Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in neural information processing systems*, 33:6377–6389, 2020.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkgsACVKPH.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 5085–5109, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.340. URL https://aclanthology.org/2022.emnlp-main.340.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. PMLR, 2022.

Chengyue Wu, Yukang Gan, Yixiao Ge, Zeyu Lu, Jiahao Wang, Ye Feng, Ping Luo, and Ying Shan. Llama pro: Progressive llama with block expansion. *arXiv preprint arXiv:2401.02415*, 2024.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pp. 38087–38099. PMLR, 2023a.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Xingrun Xing. Lm-cocktail: Resilient tuning of language models via model merging. *arXiv preprint arXiv:2311.13534*, 2023b.

Jing Xu and Jingzhao Zhang. Random masking finds winning tickets for parameter efficient fine-tuning. *arXiv preprint arXiv:2405.02596*, 2024.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=xtaX3WyCj1.

Xiaozhe Yao and Ana Klimovic. Deltazip: Multi-tenant language model serving via delta compression. *arXiv preprint arXiv:2312.05215*, 2023.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. *arXiv preprint arXiv:2311.03099*, 2023.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*, 2018.

Zhuohao Yu, Chang Gao, Wenjin Yao, Yidong Wang, Wei Ye, Jindong Wang, Xing Xie, Yue Zhang, and Shikun Zhang. Kieval: A knowledge-grounded interactive evaluation framework for large language models. *arXiv preprint arXiv:2402.15043*, 2024.

Fei Yuan, Chang Ma, Shuai Yuan, Qiushi Sun, and Lei Li. Ks-lottery: Finding certified lottery tickets for multilingual language models. *arXiv preprint arXiv:2402.02801*, 2024.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, and Wenhu Chen Yu Su. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*, 2023.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Qiusi Zhan, Richard Fang, Rohan Bindu, Akul Gupta, Tatsunori Hashimoto, and Daniel Kang. Removing rlhf protections in gpt-4 via fine-tuning, 2024.

Jinghan Zhang, Junteng Liu, Junxian He, et al. Composing parameter-efficient modules with arithmetic operation. *Advances in Neural Information Processing Systems*, 36:12589–12610, 2023a.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning, 2023b. URL https://arxiv.org/abs/2308.03303.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023c.

Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609, 2021.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103, 2017.

A ADDITIONAL EXPERIMENTAL DETAILS

A.1 CODE

Because we evaluate multiple methods on a wide range of tasks, training on > 20 datasets, we defer all the details on the prompts, exact dataset format, etc. to our open-source code repository.

A.2 HYPERPARAMETER RANGES

FFT. We tune the learning rate in the range $5e-7, 1e-5$ and the best-performing learning rate for the downstream finetuning tasks is $1e-6$, with the exception of instruction finetuning which we detail further below. We use a batch size of 32. We clip the gradient norm of each parameter group to 1. We clip the gradient norm of the parameters as opposed to the full gradient because we fuse the optimizer with the backward pass to enable FFT on a single A100.

LoTA. We do not tune any hyperparameters for LoTA and merely use the same hyperparameters as FFT. Unless stated otherwise, we report the performance of LoTA with a sparsity ratio of 0.9 that was calibrated on the entire dataset.

LoRA. LoRA introduces the additional rank hyperparameter, which we tune **jointly** with the "lora alpha" term and the learning rate. We tune the rank and alpha in the range $(8, 512)$. Common wisdom seems to dictate the use of larger learning rates for LoRA, so we expand the upper edge of the LoRA learning rate range to $1e-4$ and indeed find that LoRA typically benefits from a larger learning rate. We use LoRA on all linear layers (q, k, v, gate, up, down). On the commonsense reasoning tasks, the best hyperparameters are: $r = 512, \alpha = 32, \eta = 1e-4$. On GSM8k, the best hyperparameters are: $r = 64, \alpha = 32, \eta = 1e-4$.

Instruction Tuning. We use the hyperparameters of Sharma et al. (2024) to avoid the expense of hyperparameter tuning on winrates. This means that we use a batch size of 8 and a learning rate of $5e-7$ for both FFT and LoTA. We use the best performing hyperparameters for LoRA transferred from the commonsense reasoning tasks of $r = 64, \alpha = 32, \eta = 1e-4$.

TIES-Merging. We consider post-hoc sparsification factors of $0.1, 0.2, 0.3$; the best performance is at either 0.1 or 0.2.

A.3 MORE COMPARISONS BETWEEN LORA AND LOTA

Compute Overhead. One reason why PEFT methods such as LoRA are commonly used is because they reduce the memory consumption in the backward pass, thus enabling the use of a larger pass. Biderman et al. (2024) recently critically analyzed this and found that to fit tasks such as math and code generation, LoRA needs to use high ranks (≥ 256), which in turn reduces the speedup to at most 15%. We nevertheless acknowledge that any and all PEFT methods will be faster than LoTA during *training* because LoTA requires an initial pass over the dataset to calibrate the sparsity mask.

Memory Overhead. LoTA has a minor overhead when compared to FFT, because a bit-mask must be loaded for each parameter group. We provide an efficient implementation for this in our open-source code. repository. We can finetune Llama-3-8B with LoTA on a single 80GB A100, so we do not anticipate this minor memory overhead being a major limitation of our method. LoRA reduces the memory footprint during finetuning, at the expense of accuracy.

High-Rank Adaptation Beats Low-Rank Adaptation. LoRA has a heavy regularizing effect on training. On the commonsense reasoning task, because the base model can get nontrivial zero-shot performance, LoRA actually improves the performance over that of FFT for Mistral. Note here that we do a grid search over learning rate and rank for LoRA, and the best performance is at $r = 64, \eta = 1e-4$. However, sparsity also has a regularizing effect, and LoTA is even more successful on the reasoning task when using the same learning rate as we searched for the base model ($1e-6$). On GSM8k, the regularization hurts performance significantly; (Nikdan et al., 2024) report a similar-sized gap between LoRA and FFT on this dataset when training LLama-2-7B. If LoRA is underfitting the data, it may be better suited for settings where we only train for a single epoch, or on smaller datasets. In Table 8, we make a side-by-side comparison of LoRA and LoTA when training for a single epoch and find that LoTA outperforms LoRA significantly across all tasks; in fact, the difference is even more pronounced after a single epoch.

Table 8: Performance comparison on single-task datasets for 1 epoch. **bold**: best method

Method	Model	Arc	GSM8k	SQL	Summarization
LoRA	Mistral	$70.4_{0.6}$	$46.3_{1.1}$	$98.6_{0.1}$	$51.8_{0.2}$
LoTA (ours)	Mistral	$\mathbf{73.8_{0.7}}$	$\mathbf{53.5_{1.0}}$	$\mathbf{99.3_{0.1}}$	$\mathbf{54.3_{2.5}}$

Storing LoTA Adapters Efficiently. Although FFT generally performs better than PEFT methods, it is typically infeasible to store a full copy of the model weights for each task. Practitioners, therefore, consider a tradeoff between memory and adaptation performance when loading task-specific adapters. We now discuss the storage memory consumption of LoTA and LoRA. Storing the sparse task vector from LoTA requires 40 bits per parameter; 32 from the parameter, and 8 for the delta encoding of the sparse indices. If we use 90%-sparse LoTA, our task vector is compressed $8\times$; if we use 99%-sparse LoTA, our task vector is compressed $80\times$. The memory-utility tradeoff between LoTA and LoRA can be quantified in terms of each method's performance at a given level of compression, which translates into the sparsity level for LoTA and the rank r for LoRA. Comparing the compression-utility tradeoff between LoRA and LoTA is challenging because the size of the saved LoRA adapter is determined by the rank parameter r, and more is not always better (which is why we have to tune r for every task for LoRA). As a single point of comparison, we can look at the performance on GSM8k, where 99%-sparse LoTA (62.0) outperforms LoRA for any rank (best performance of 58.3 achieved at $r = 512$, increasing rank reduces performance), and they both achieve a similar compression factor.

A.4 Individual Task Results for Averaged Experiments

In the main body we present a number of experiments where we have to report the average performance over a number of tasks for space constraints. We now present the individual task results in Table 9 and Table 10.

Table 9: Individual task results for the "averaged" results in Table 6. **bold**: best method.

FT Method on Task A	FT Method on Task B	Instruction Following	Arc	GSM8k	SQL
FFT	FFT	$0.45_{0.21}$	$74.1_{.09}$	$51.9_{0.09}$	$98.9_{0.01}$
LoTA (ours)	FFT	$\underline{11.48_{0.73}}$	$73.3_{0.02}$	$53.9_{0.09}$	$98.9_{0.01}$
LoTA (ours)	LoTTO (ours)	$\mathbf{15.88_{0.88}}$	$70.3_{0.01}$	$52.5_{0.01}$	$98.6_{0.01}$

Table 10: Full results on commonsense reasoning.

Model	Method	BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Avg.
Mistral	LoRA	76.4	90.7	81.2	91.4	87.0	94.1	83.0	90.4	86.8
	LoTA (ours)	75.3	90.0	83.3	91.4	89.5	95.1	85.0	90.2	87.5

B Related Work

Model Pruning & Quantization. Model pruning and quantization have been receiving increased attention for efficient storage and/or inference of large models. While most of the existing methods prune (Frantar & Alistarh, 2023; Dettmers et al., 2023b; Kim et al., 2023) or quantize (Frantar et al., 2023; Dettmers et al., 2022; Xiao et al., 2023a; Lin et al., 2023) the model weight directly, some focus specifically on compressing the *task vectors* (Isik et al., 2023; Liu et al., 2024a; Yao & Klimovic, 2023) through post-training sparsification or quantization, assuming that the base model is worth the storage cost since it is being used frequently for many tasks. Our proposed PEFT method, LoTA, builds on this observation that the task vectors are highly compressible through sparsification and imposes this sparsity constraint at the beginning of fine-tuning to train *sparse task vectors*.

Lottery Ticket Hypothesis. Motivated by the success of pruning methods at extreme sparsity ratios (Han et al., 2015), Frankle & Carbin (2019) proposed the lottery ticket hypothesis (LTH), claiming the existence of sparse subnetworks (or lottery tickets) that could be trained from scratch to a performance comparably to training the dense model from scratch. While this could potentially provide a way to train models sparsely more efficiently rather than training them densely and pruning them later, finding the lottery tickets, i.e., the sparsity masks, is costly. Initially, Frankle & Carbin (2019) proposed first training the models densely and then extracting the sparsity mask based on the magnitude of the trained dense model's weights. Later, a number of more efficient methods were proposed to find the sparsity masks more efficiently, earlier in the dense training stage (Frankle et al., 2021; Lee et al., 2019; Wang et al., 2020; Tanaka et al., 2020). Our work shows that LTH works successfully for fine-tuning LLMs as well–giving us a sparse adaptation tool, LoTA. We extensively study the tradeoff between the cost of finding the sparsity masks and the performance of the sparsely fine-tuned model. Unlike other studies on LTH for LLM adaptation (Yuan et al., 2024; Xu & Zhang, 2024), our main focus and motivation is to mitigate destructive interference in multi-task adaptation.

Parameter-Efficient Fine-Tuning (PEFT). Many practitioners fine-tune already pre-trained LLMs with less data and compute instead of training them from scratch (Liu et al., 2022a; Wang et al., 2022; Ouyang et al., 2022). While this reduces the cost of LLM training significantly, fine-tuning each and every parameter of these large models for each (or a few) task is still very costly. This has led to a number of parameter-efficient fine-tuning (PEFT) methods reducing the number of trainable parameters during fine-tuning (Zhang et al., 2023c; Li & Liang, 2021; Liu et al., 2022b; Lester et al., 2021; Edalati et al., 2022; Hu et al., 2023; Nikdan et al., 2024; Guo et al., 2020). Among different PEFT methods, low-rank adaptation (LoRA) (Hu et al., 2022) and its variants (Dettmers et al., 2023a; Guo et al., 2024; Li et al., 2024; Kopiczko et al., 2024) have shown similar performance to full fine-tuning in many tasks while reducing the number of trainable parameters through low-rank approximation to model updates during fine-tuning. Our PEFT method, LoTA, while reducing the number of trainable parameters significantly via sparsity, has various other benefits in different applications, such as avoiding catastrophic forgetting (of especially safety alignment), enabling fine-tuning on new tasks more successfully, model merging using sparse task vectors, unlearning, and communication-efficient federated learning (FL). We demonstrate that full fine-tuning and the existing PEFT methods fall short in these applications and significantly underperform LoTA.

Catastrophic Forgetting. When LLMs go through sequential (or continual) multitask learning, i.e., fine-tuned on different tasks sequentially, they often suffer from performance loss on earlier tasks–known as catastrophic forgetting (McCloskey & Cohen, 1989; Dong et al., 2023; Ramasesh et al., 2022; Luo et al., 2023; Wang et al., 2024). To mitigate this, a number of data-centric and architectural solutions have been proposed for language and other domains. Replay-based methods (Rebuffi et al., 2017; Romanov et al., 2018) add a portion of the previously learned data during fine-tuning on a new task, which raises privacy concerns as it requires constant access to previously learned data. Regularization-based approaches (Huang et al., 2021; Aljundi et al., 2018) tend to have poor adaptability to specific tasks. An architecture-based approach, "progressive prompts" Razdaibiedina et al. (2023), sequentially concatenates soft prompts as they are being learned for each task–showing some resistance against forgetting. However, they require access to task identifiers at inference for each task, which is not always feasible. Other architecture-based approaches add additional modules to learn task-specific abilities (Dou et al., 2023; Wu et al., 2024)–requiring customized deployment due to architecture change. Closest to our work, Hui et al. (2024) updates a randomly selected subset of the parameters at each iteration of fine-tuning to preserve the earlier tasks in the not-updated parameters of that iteration. Despite similarities, our work LoTA (1) uses a fixed sparsity mask throughout fine-tuning instead of a new mask at every iteration, which yields sparse task vectors that are useful for other applications such as model merging and communication-efficient FL, and (2) finds data-dependent masks rather than the randomly selected masks in (Hui et al., 2024). Furthermore, unlike (Hui et al., 2024), LoTA not only preserves the earlier tasks on frozen parameters but also constraints the new tasks on a highly sparse subnetwork–providing resistance to catastrophic forgetting even when malicious users attempt to overwrite the earlier tasks via FFT.

Model Merging. Merging multiple task-specific models into a single model with multitask abilities (Wortsman et al., 2022; Jin et al., 2022; Zhang et al., 2023a) has been an appealing alternative to sequential multitask learning, which suffers from catastrophic forgetting and could be inefficient, especially if task vectors are already available. The existing model merging methods include averaging weights of task-specific models (Wortsman et al., 2022), task arithmetic through combining task vectors (Ilharco et al., 2022), weighted aggregation of parameters (Matena & Raffel, 2022; Xiao et al., 2023b), combining task vectors after some post-processing such as trimming low-magnitude deltas (Yadav et al., 2023) or sparsifying the deltas (Yu et al., 2023; Davari & Belilovsky, 2023). Our method, LoTA, directly learns *sparse task vectors*, obviating the need to post-process the task vectors, and outperforms existing model merging methods. Most importantly, LoTA enables merging task vectors trained on heterogeneous datasets, while the other model merging methods are often limited to similar datasets. This advancement is an important step towards scalable FL (Kairouz et al., 2021) with LLMs as it enables merging *sparse task vectors*, which brings communication efficiency, trained over heterogeneous datasets (of each edge device).

We note that we test model merging with LoTA specifically for highly dissimilar datasets to show its compatibility with FL, which considers edge devices with heterogeneous datasets. When used in FL, LoTA can reduce communication and memory costs significantly, which is a main bottleneck when scaling FL to large models. The successful use of LoTA for model merging and arithmetic further shows its promise for unlearning (Ilharco et al., 2022) as well.