LoRASC: Expressive and Generalizable Low-rank Adaptation for Large Models via Slow Cascaded Learning

Siwei Li[1*], **Yifan Yang**[2*†], **Yifei Shen**[2], **Fangyun Wei**[2],
Zongqing Lu[1], **Lili Qiu**[2], **Yuqing Yang**[2]
[1]Tsinghua University, [2]Microsoft Research Asia
siweili505@outlook.com, yifanyang@microsoft.com

Abstract

Efficient fine-tuning plays a fundamental role in modern large models, with low-rank adaptation emerging as a particularly promising approach. However, the existing variants of LoRA are hampered by limited expressiveness, a tendency to overfit, and sensitivity to hyperparameter settings. This paper presents LoRA Slow Cascade Learning (*LoRASC*), an innovative technique designed to enhance LoRA's expressiveness and generalization capabilities while preserving its training efficiency. Our approach augments expressiveness through a cascaded learning strategy that enables a mixture-of-low-rank adaptation, thereby increasing the model's ability to capture complex patterns. Additionally, we introduce a slow-fast update mechanism and cascading noisy tuning to bolster generalization. The extensive experiments on various language and vision datasets, as well as robustness benchmarks, demonstrate that the proposed method not only significantly outperforms existing baselines, but also mitigates overfitting, enhances model stability, and improves OOD robustness. Code will be release in https://github.com/microsoft/LoRASC very soon.

1 Introduction

Foundation models, which are large-scale models pre-trained on extensive datasets and subsequently adapted for specific downstream tasks, have become integral to contemporary machine learning frameworks. Fine-tuning these models is essential, yet full parameter fine-tuning often encounters significant memory and computational bottlenecks. As a result, Parameter-Efficient Fine-Tuning (PEFT) techniques, which aim to minimize the number of trainable parameters to reduce training costs and improve training stability, have gained increasing prominence. Among these techniques, Low-Rank Adaptation (LoRA) (Hu et al., 2021) stands out due to its efficiency in reducing training costs through low-rank approximation for full-parameter updates. However, despite LoRA's advantages, its limitations in terms of expressiveness and generalization have been noted. Some studies suggest that the inherent low-rankness of LoRA might restrict its expressiveness (Xia et al., 2024; Meng et al., 2024; Lialin et al., 2023; Huang and Wei, 2024), with a preference for overparameterization, while others indicate a tendency for LoRA to overfit or exhibit overconfidence (Lin et al., 2024; Wang et al., 2023).

In this work, we investigate the potential of cascading learning to augment the expressiveness of LoRA. Our approach involves initializing a new LoRA module at the start of each epoch and integrating this module into the backbone network after the epoch concludes. By employing a mixture-of-low-rank adaptation, we effectively increase the rank of the update matrices, while maintaining low training costs, as each cascading step consumes no more parameters and memory than a single LoRA model. Moreover, this method does not add any inference overhead by remerging each LoRA module into the backbone network.

To improve LoRA's generalization capabilities, we draw inspiration from optimization techniques. We repurpose certain strategies from optimizers for LoRA, motivated by the observation that initializing a new LoRA module for each epoch can represent a descent direction for the dataset. In optimization theory, flat minimizers are preferred, as they are associated with better generalization (Hochreiter and Schmidhuber, 1997; Keskar et al., 2016). Inspired by the fact that the moving average mechanism guides models towards flat minimizers (Izmailov et al., 2018), we maintain both fast-updating and its moving average version, the slow-updating

*Equal contribution. This work was done during Siwei Li's internship at Microsoft Research Asia.
†Corresponding author.

LoRA experts. The fast-updating expert is reinitialized regularly to learn from the data over a set number of steps, while the slow-updating expert undergoes updates via a proportional exponential moving average after the fast-updating cycle completes. Additionally, mirroring techniques in deep learning optimizers where noise proportional to the gradient scale is used to find flat minima (Xie et al., 2020), we introduce noise at the beginning of each epoch, with the scale tied to the norm of LoRA's weights.

To verify the effectiveness of the proposed method, we conduct extensive experiments on both language and vision tasks. For language tasks, we utilized the Llama2 model (Touvron et al., 2023) on 12 datasets (e.g., SuperGLUE, SQuAD, DROP, GSM8K, and InstructEval), Alpaca among other instruct following benchmarks to demonstrate the effectiveness of our design. We can directly apply our approach to LoRA, LoRA+ (Hayou et al., 2024), Dora (Liu et al., 2024), and other members of the LoRA family, significantly improving their performance in large model transfer learning. For vision tasks, we also validated our approach on the CLIP pre-trained Vit-bigG model with the ImageNet dataset, showing a significant performance improvement relative to LoRA on domain adaptation datasets such as Image-R and Image-C. The proposed method consistently outperforms the baselines by a large margin.

2 Related Work

2.1 Low-Rank Adaptation Finetuning

LoRA (Hu et al., 2021) has been widely adopted as a parameter-efficient fine-tuning method, demonstrating its effectiveness in various transfer learning scenarios.LoRA+ (Hayou et al., 2024) improves performance and fine-tuning speed by setting different learning rates for the LoRA adapter matrices A and B with a carefully chosen ratio, maintaining the same computational cost as LoRA. Dora (Liu et al., 2024) decomposes the pre-trained weight into two components, magnitude and direction, for fine-tuning, specifically employing LoRA for directional updates to efficiently minimize the number of trainable parameters. Our work introduces a robust cascading learning schedule for various LoRA variants, proving through extensive experiments that it can enhance the training performance of LoRA, LoRA+, and Dora without additional training costs.

2.2 Combination of LoRA

LoRAhub (Huang et al., 2023) presents a simple framework designed for the purposeful assembly of LoRA modules trained on diverse tasks, aiming to achieve adaptable performance on unseen tasks. MOLE (Huang and Wei, 2024) treats each layer of trained LoRAs as a distinct expert and implements hierarchical weight control by integrating a learnable gating function within each layer. LoRAFlow (Wang et al., 2024) utilizes dynamic weights to adjust the impact of different LoRAs. These methods are not in conflict with *LoRASC*, as they focus on learning the combination of LoRA experts across different domains, while our method aims to learn more generalizable experts within a single domain using slow cascade learning.

ReLoRA (Lialin et al., 2023) enhances LoRA's fitting ability by incrementally merging learned LoRA parameters into the main network and restarting optimizer parameters during training. It also proposes a jagged cosine scheduler to implement a learning rate resume strategy at each step. COLA (Xia et al., 2024) explores a similar approach but in a simpler manner, merely restarting optimizer parameters when initializing new LoRAs without adjusting the learning rate schedule. Our work employs a simpler cascading learning strategy where each expert learns independently for each epoch, without additional design for learning schedules or optimizer parameters. Additionally, we incorporate noise tuning and slow-fast update strategy, ensuring robustness in each expert merged into the pre-trained model. Our method can be applied to various LoRA variants, demonstrating effectiveness across multiple tasks in both language and image domains.

3 Methods

3.1 LoRA

For parameter-efficient fine-tuning, we focus on the LoRA technique, which has been widely used due to its lower training cost. Instead of updating all parameters of the model, LoRA inserts low-rank matrices into each layer of the pre-trained model, which are then fine-tuned. This reduces the computational burden and the risk of overfitting.

Given a pre-trained weight matrix $W_0 \in \mathbb{R}^{d \times k}$ in a neural network, LoRA approximates the update ΔW using two low-rank matrices $A \in \mathbb{R}^{d \times r}$ and $B \in \mathbb{R}^{r \times k}$, where $r \ll \min(d, k)$. The update is defined as:

$$\Delta W = BA \tag{1}$$

During fine-tuning, instead of updating W, we update A and B, which results in:

$$W = W_0 + \Delta W = W_0 + BA \tag{2}$$

This low-rank adaptation significantly reduces the number of trainable parameters from $d \times k$ to $r \times (d + k)$.

3.2 *LoRASC*

3.2.1 Cascading LoRA Learning

Due to the reparameterization nature of low-rank adaptation (LoRA) fine-tuning, employing multiple LoRA experts incurs the same inference cost as using a single LoRA expert. This characteristic makes LoRA particularly suitable for integration with cascading learning to enhance performance in transfer learning tasks. As analyzed in ReLoRA (Lialin et al., 2023), reinitializing new LoRA modules during the learning schedule can progressively increase the model's rank, thereby improving its fitting ability.

In *LoRASC*, we default to learning one LoRA expert per epoch. After training one LoRA expert, it is merged into the main network, and the next expert learns based on the optimized residuals. The optimization schedule for each single LoRA expert is a compressed version of the original full-training schedule: for instance, if a model was originally trained for N epochs, each expert in *LoRASC* completes training in 1 epoch with fixed starting and ending learning rates with the same but compressed scheduler. This makes *LoRASC* easy to apply to any large model transfer learning scenario using LoRA, without requiring changes to hyperparameters. The only necessary adjustment is an increase in the learning rate. Since the number of training steps is smaller comparing to full training, each step must be larger to cover the same distance. Additionally, Li et al. (Li et al., 2019) found that higher learning rates can lead to stronger generalization ability, which might also explain the improved out-of-domain performance of our method.

Mathematically, the cascading LoRA learning can be described as follows:

1. For each epoch t, train a new LoRA expert (A_t, B_t) to minimize the residual error, where $\mathcal{L}$ is the fine-tuning loss function:

$$(A_t, B_t) = \arg\min_{A_t, B_t} \mathcal{L}\left(W_{t-1} + B_t A_t\right), \tag{3}$$

2. Merge the trained LoRA expert into the main network:

$$W_t = W_{t-1} + B_t A_t \tag{4}$$

By iteratively merging each new LoRA expert into the main network, LoRA cascading progressively enhances the model's capacity to fit the data without increasing the inference cost.

3.2.2 LoRA Slow-Fast Update

To enhance the generalization of large model transfer learning, we aim to avoid local optima at each step of cascading. Even with low-rank adaptation, this issue persists due to the imbalance between model parameters and training data. Inspired by SWA (Izmailov et al., 2018), which averages model parameters over several epochs to find a more generalized solution, we employ a sliding average method to ensure the stability and robustness of each LoRA merged into the main network.

Specifically, during training, we maintain two LoRA experts at each cascading step t as shown in Fig. 1: a slow-updating LoRA $(A_t^{\text{slow}}, B_t^{\text{slow}})$ and a fast-updating LoRA $(A_t^{\text{fast}}, B_t^{\text{fast}})$. At step 0, both slow and fast LoRA share the same initialization. During each cascading iteration, fast LoRA undergoes fine-tuning, and after completion, it is averaged with slow LoRA. The slow LoRA is then merged into the pre-trained model, while the fast-updating LoRA is reinitialized for the next iteration. We control the retention proportion of the slow expert with a hyperparameter α.

The update rules are given by:

$$A_{t+1}^{\text{slow}} = \alpha A_t^{\text{slow}} + (1 - \alpha) A_t^{\text{fast}} \tag{5}$$

$$B_{t+1}^{\text{slow}} = \alpha B_t^{\text{slow}} + (1 - \alpha) B_t^{\text{fast}} \tag{6}$$

By employing this slow-fast update strategy, *LoRASC* ensures that each merged LoRA expert contributes to a more generalized solution, enhancing the overall stability and performance of the model in transfer learning scenarios.

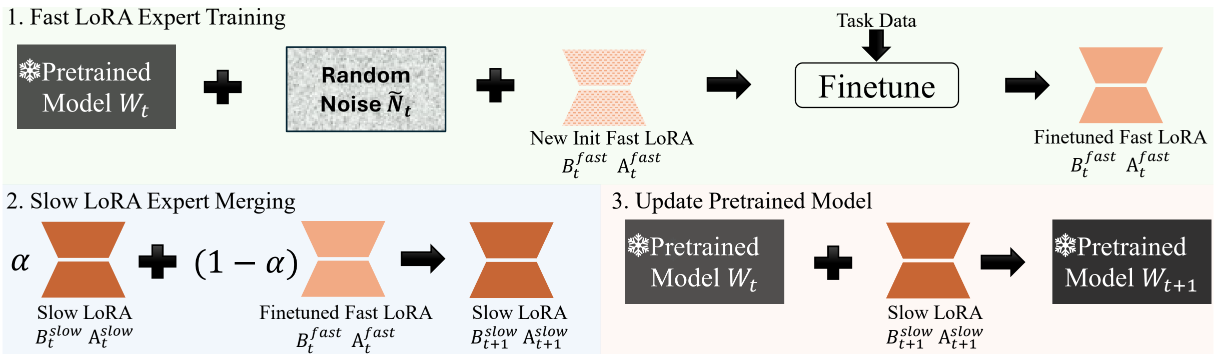
1. Fast LoRA Expert Training
Pretrained Model W_t
Random Noise $\widetilde{N}_t$
New Init Fast LoRA B_t^{fast} A_t^{fast}
Task Data
Finetune
Finetuned Fast LoRA B_t^{fast} A_t^{fast}
2. Slow LoRA Expert Merging
α
Slow LoRA B_t^{slow} A_t^{slow}
$(1-\alpha)$
Finetuned Fast LoRA B_t^{fast} A_t^{fast}
Slow LoRA B_{t+1}^{slow} A_{t+1}^{slow}
3. Update Pretrained Model
Pretrained Model W_t
Slow LoRA B_{t+1}^{slow} A_{t+1}^{slow}
Pretrained Model W_{t+1}

Figure 1: Iterative pipeline of *LoRASC*. Here, t represents the iteration step, and BA denotes the low-rank learnable vectors in LoRA. The backbone network W always has its gradients turned off, and α is the hyperparameter controlling the pace of the slow-fast update. Our method follows three stages: 1. Fast LoRA expert training, where noise is added to the backbone network, followed by training the fast LoRA on the task data. 2. Slow LoRA expert merging, where a portion of the learned fast LoRA is weighted and merged into the slow LoRA. 3. Update the pretrained model, merging the updated slow LoRA into the backbone network, and prepare for the next iteration.

3.2.3 Cascading Noisy Tuning

To further enhance generalization, we introduce random noise to the pre-trained model before each new LoRA fine-tuning step. Unlike NoisyTune (Wu et al., 2022), which adds uniform noise to different parameter matrices according to their standard deviations only once at the beginning of fine-tuning, we apply noise before training each new expert. This approach helps the model escape local optima at every slow LoRA step, thereby reducing the risk of overfitting.

Additionally, the presence of the slow-updating LoRA module indicates the direction of parameter changes under the new task. Therefore, we use the standard deviation of the slow LoRA weights to determine the noise scale rather than the pre-trained model's weights. Incorporating this noise before every expert ensures that the model continuously explores robust and flatten parameter spaces, thus improving generalization and reducing the tendency to overfit.

The perturbation is defined as:

$$\widetilde{N}_t = U\left(-\frac{\lambda}{2}, \frac{\lambda}{2}\right) \cdot \mathrm{std}(B_t^{\mathrm{slow}} A_t^{\mathrm{slow}}) \tag{7}$$

where std stands for standard deviation. The function $U(a, b)$ represents uniform distribution noise ranged from a to b, and λ is a hyperparameter that controls the relative noise intensity.

3.3 Overview

With LoRA cascading learning, slow-fast updates and noisy tuning, the pipeline of our *LoRASC* is as follows:

$$\widetilde{W}_{t-1} = W_{t-1} + \widetilde{N}_t \tag{8}$$

$$(A_t^{\mathrm{fast}}, B_t^{\mathrm{fast}}) = \arg\min_{A_t^{\mathrm{fast}}, B_t^{\mathrm{fast}}} \mathcal{L}\left(\widetilde{W}_{t-1} + B_t^{\mathrm{fast}} A_t^{\mathrm{fast}}\right) \tag{9}$$

$$A_t^{\mathrm{slow}} = \alpha A_{t-1}^{\mathrm{slow}} + (1-\alpha) A_t^{\mathrm{fast}} \tag{10}$$

$$B_t^{\mathrm{slow}} = \alpha B_{t-1}^{\mathrm{slow}} + (1-\alpha) B_t^{\mathrm{fast}} \tag{11}$$

$$W_t = \widetilde{W}_{t-1} + B_t^{\mathrm{slow}} A_t^{\mathrm{slow}} \tag{12}$$

LoRASC pipeline can be seen in Fig. 1. Although we use vanilla LoRA to show slow casdade learning, *LoRASC* should be able to boost the performance of any LoRA variants, such as DoRA (Liu et al., 2024), LoRA+ (Hayou et al., 2024), LoRA-FA (Zhang et al., 2023), etc. Moreover, *LoRASC* is easy to implement, and we provide pseudocode with more detailed explanations in Algorithm 1.

4 Experiments

We conducted extensive experiments to demonstrate the effectiveness and robustness of *LoRASC* across both NLP and CV domains.

For language tasks, we conducted our language experiments using the popular open-source large language model, Llama2[1]. We evaluated our approach on several NLU and GLU tasks, selecting

[1] `https://huggingface.co/meta-llama/Llama-2-7b-hf`

Algorithm 1 Pseudo Code for *LoRASC*

```
Require: Pre-trained model weights $W_0$, number of epochs $T$, loss function $\mathcal{L}$, slow update parameter $\alpha$, noise parameter $\lambda$
 1: Initialize $W \leftarrow W_0$
 2: Initialize $A^{\text{slow}}$, $B^{\text{slow}}$   ▷ Initialize slow LoRA matrices
 3: Initialize $A^{\text{fast}} \leftarrow A^{\text{slow}}$, $B^{\text{fast}} \leftarrow B^{\text{slow}}$   ▷ Fast LoRA matrices initialized from slow ones
 4: for epoch $t = 1$ to $T$ do
 5:     if $t > 1$ then
 6:         Reinitialize $A^{\text{fast}}$, $B^{\text{fast}}$   ▷ Reinitialize fast LoRA matrices for subsequent epochs
 7:     end if
 8:     $\widetilde{W} \leftarrow W + U\left(-\frac{\lambda}{2}, \frac{\lambda}{2}\right) \cdot \text{std}(B^{\text{slow}} A^{\text{slow}})$
 9:     optimizer $\leftarrow$ InitializeOptimizer($A^{\text{fast}}$, $B^{\text{fast}}$)
10:     lr_scheduler $\leftarrow$ InitializeLRScheduler(optimizer)
11:     for batch in training data do
12:         Forward pass: $L \leftarrow \mathcal{L}(\widetilde{W} + B^{\text{fast}} A^{\text{fast}})$
13:         Backward pass: Compute gradients
14:         optimizer.step()
15:         lr_scheduler.step()
16:     end for
17:     Update slow LoRA:
18:     $A^{\text{slow}} \leftarrow \alpha A^{\text{slow}} + (1 - \alpha) A^{\text{fast}}$
19:     $B^{\text{slow}} \leftarrow \alpha B^{\text{slow}} + (1 - \alpha) B^{\text{fast}}$
20:     Merge slow LoRA into main network: $W \leftarrow \widetilde{W} + B^{\text{slow}} A^{\text{slow}}$
21: end for
22: return $W$
```

both SuperGLUE (Wang et al., 2019a) tasks (including classification and multiple-choice) and generation tasks. We also tested the model's performance in mathematical reasoning using the GSM8K dataset (Cobbe et al., 2021). Additionally, we performed instruction tuning experiments to verify the transfer learning capability of our method, achieving significant improvements on key metrics such as MMLU (Hendrycks et al., 2020), DROP (Dua et al., 2019), BBH (Srivastava et al., 2022) and HumanEval (Chen et al., 2021).

For visual tasks, we chose the CLIP ViT-bigG/14 (Cherti et al., 2022)[2] as our pretrained model, fine-tuning it on the ImageNet-1K (Deng et al., 2009) training set and testing it on the validation set. Subsequently, we evaluated the trained model on perturbed datasets such as ImageNet-A (Hendrycks et al., 2021b), ImageNet-C (Hendrycks and Dietterich, 2019), ImageNet-R (Hendrycks et al., 2021a), ImageNet-V2 (Recht et al., 2019), ImageNet-Sketch (Wang et al., 2019b) and Stylized-ImageNet (Geirhos et al., 2018) demonstrating our method's robustness and generalization capabilities.

[2]https://huggingface.co/laion/CLIP-ViT-bigG-14-laion2B-39B-b160k

4.1 Implementation Details

For all experiments, we exclusively fine-tuned the query and value projection matrices in the attention layers as delineated by Malladi et al. (2023) and Ren et al. (2024). The fine-tuning process utilized single NVIDIA H100 GPU. For all tasks, we explored several learning rates and reported the optimal performance. For the hyper-parameters of *LoRASC*, we explored the factor α of Slow-Fast Update in $\{0.5, 0.6, 0.8\}$ to control the updating ratio. Additionally, we selected the noise intensity from $\{0.1, 1, 10\}$, which is a significantly smaller set compared to the default 7 in NoistTune (Wu et al., 2022). All the reported results were averaged across 3 distinct random seeds.

4.2 Main Results

4.2.1 *LoRASC* for Large Language Model

Experiment setting. For in-domain language transfer learning, we consider the SuperGLUE dataset collection (Wang et al., 2019a), including: BoolQ (Clark et al., 2019), CB (De Marneffe et al., 2019), COPA (Roemmele et al., 2011), MultiRC (Khashabi et al., 2018), ReCoRD (Zhang et al., 2018), RTE (Socher et al., 2013), WiC (Pilehvar and Camacho-Collados, 2019), and WSC (Levesque et al., 2012). We also include SST-2 (Dagan et al., 2005) , GSM8K (Cobbe et al., 2021) and two question answering(QA) datasets, SQuAD (Rajpurkar et al., 2016) and DROP (Dua et al., 2019). And we directly used 8-shot direct prompting or GSM8K evaluation[3]. We adhered to the experimental configuration described by Malladi et al. (2023), randomly selecting 1000 examples for training, 500 for validation, and 1000 for testing across each dataset. The AdamW optimizer was employed, with training spanning 5 epochs, consistent with the baseline settings. A linear learning rate schedule was implemented, with the initial learning rate selected from $\{1\times10^{-5}, 5\times10^{-5},$

[3]https://github.com/allenai/open-instruct

Task	SST-2	RTE	CB	BoolQ	WSC	WIC	MultiRC	COPA	ReCoRD	SQuAD	DROP	GSM8K
Task type	classification							multiple choice		generation		math
LoRA	95.5	87.4	91.1	85.7	70.2	72.4	85.3	85.0	81.2	90.4	51.6	19.5
w/ **COLA**	95.9	87.7	91.1	85.7	66.4	72.6	85.3	82.0	81.4	90.6	51.6	21.0
w/ LoRASC												
+ Cascade	95.8	87.7	92.9	86.1	71.1	72.3	86.3	**88.0**	81.6	91.8	52.5	21.5
++ Slow LoRA	96.0	88.0	96.4	86.8	74.0	72.1	86.3	**88.0**	82.1	92.7	55.3	**27.5**
+++ Noise Tuning	**96.1**	**88.1**	**96.5**	**87.4**	**75.0**	**72.7**	**86.6**	**88.0**	**82.2**	**92.9**	**56.7**	**27.5**
LoRA+	95.7	87.0	91.4	85.9	69.2	72.1	85.7	87.0	81.3	90.5	55.8	22.0
w/ LoRASC												
+ Cascade	95.7	87.0	**92.9**	86.2	71.2	72.8	85.3	**88.0**	**81.9**	91.2	55.8	19.5
++ Slow LoRA	95.7	**88.1**	**92.9**	85.9	67.3	73.5	85.7	**88.0**	**81.9**	**92.0**	56.3	23.0
+++ Noise Tuning	**95.8**	**88.1**	**92.9**	**86.3**	**71.4**	**74.1**	**86.1**	**88.0**	**81.9**	**92.0**	**56.4**	**24.0**
DoRA	95.4	87.4	96.4	85.7	72.1	71.5	84.7	88.0	81.1	91.1	54.8	21.0
w/ LoRASC												
+ Cascade	95.8	87.4	96.4	85.8	65.4	**72.8**	84.1	88.0	81.6	91.7	52.6	22.5
++ Slow LoRA	95.8	88.1	96.4	85.8	65.4	**72.8**	86.1	88.0	81.9	92.8	54.8	25.0
+++ Noise Tuning	**96.0**	**88.5**	**96.5**	**87.6**	**75.6**	**72.8**	**86.8**	**89.0**	**82.2**	**93.3**	**56.5**	**25.5**

Table 1: Comparative Performance of LoRA, LoRA+, and DoRA enhanced with *LoRASC* across multiple in-domain fine-tuning datasets.

Method	MMLU	DROP	HEval	BBH	GSM8K
LoRA	45.83	32.76	31.26	13.41	11.5
w/ LoRASC					
+ Cascade	45.53	32.71	**31.61**	14.02	11.5
++ Slow LoRA	45.68	**33.74**	31.38	**17.07**	12.5
+++ Noise	**45.98**	33.02	**31.61**	15.24	**16.5**

Table 2: Results on instruction-following tasks. The model was trained on Alpaca and evaluated on InstructEval metrics and GSM8K. *LoRASC* consistently achieves the best performance compare to vanilla LoRA.

$1{\times}10^{-4}$, $5{\times}10^{-4}$, $1{\times}10^{-3}$}. By default the batch size was set to 4 and the LoRA rank was set to 8. For LoRA+, we adhered to its setup by fixing the learning rate of B matrices to be 16 times that of A matrices. DoRA decomposes the pre-trained weight into magnitude and direction components, with LoRA efficiently updating the direction component. This means that each LoRA expert represents DoRA's direction component. When applying *LoRASC* to DoRA, we maintain continuous training of the magnitude while applying our technique to the direction component. We follow the standard procedure of merging and reinitializing LoRA and align it with the slow-fast update and noisy tuning.

For instruction tuning, we use the Alpaca[4] (Taori et al., 2023) dataset for training. The batch size was set to 128. We follow the training scripts of Ren et al. (2024) in our experiment. We finetune our model for 3 epochs. A linear learning rate schedule was applied, with the initial learning rate selected from {$1{\times}10^{-4}$, $3{\times}10^{-4}$, $5{\times}10^{-4}$, $1{\times}10^{-3}$}. For evaluation we use InstructEval[5] (Chia et al., 2023), 5-shot direct prompting for MMLU , 3-shot direct prompting for BBH and DROP, 0-shot direct prompting for HEval.

***LoRASC* exhibits excellent adaptability to LoRA variants.** In the experiments shown in Table 1, *LoRASC* outperforms the COLA across various tasks, demonstrating the effectiveness of our LoRA cascading technique. Moreover, *LoRASC* effectively boosted the performance of LoRA, LoRA+, and DoRA across 12 in-domain training datasets encompassing four major tasks: classification, multiple choice, generation, and mathematics. *LoRASC* achieved significant improvements across all these tasks, demonstrating its ability to enhance the learning capabilities and in-domain generalization of the LoRA family of models. Moreover, the progressive addition of cascading learning, slow-fast updates,

[4]https://github.com/tatsu-lab/stanford_alpaca/

[5]https://github.com/declare-lab/instruct-eval

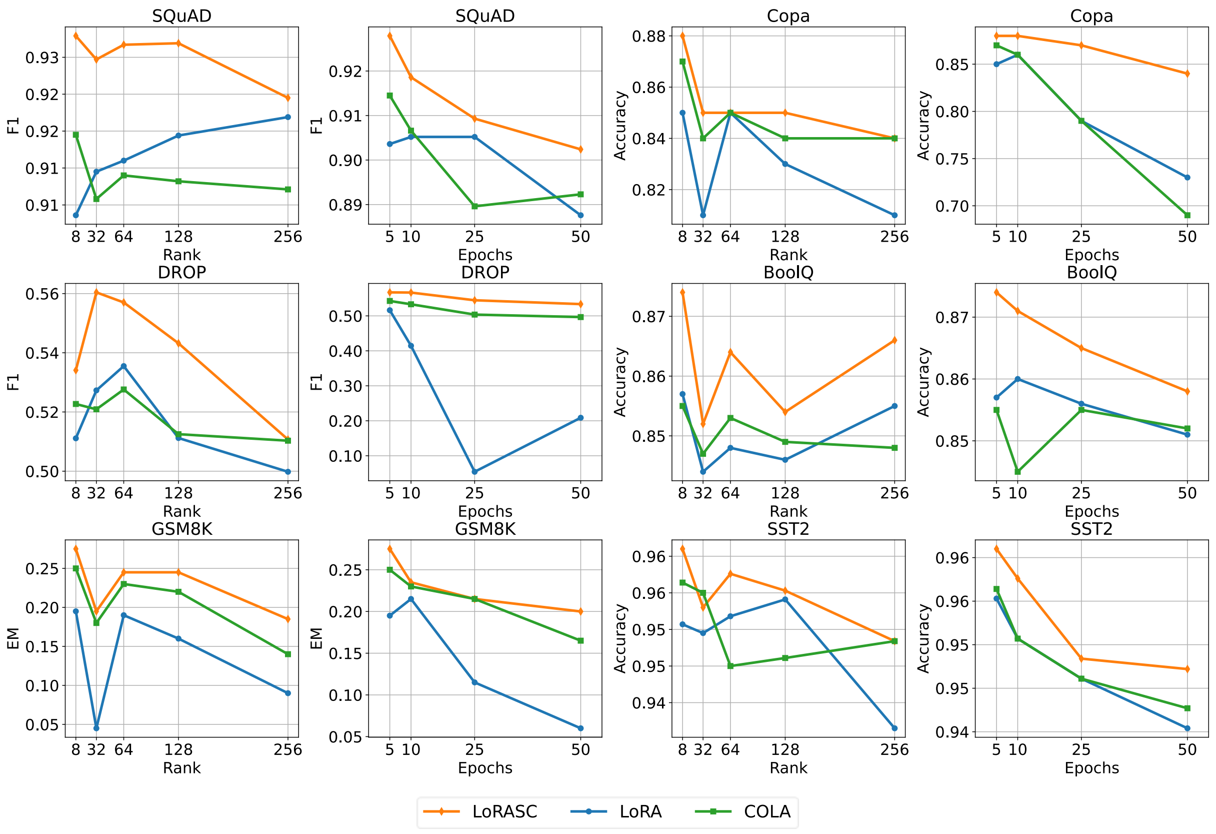

Figure 2: Performance of *LoRASC* compared to LoRA and COLA across various ranks and learning schedules in a subset of text transfer learning tasks. It can be observed that *LoRASC* consistently achieves stable performance improvements across all ranks and learning schedules, particularly at higher ranks and longer epochs, where *LoRASC* can mitigate performance degradation caused by overfitting.

and noisy tuning further improved performance, validating the design of our approach. The robust slow cascading strategy not only enhanced overall performance but also provided strong generalization capabilities.

***LoRASC* on Instruction-Following tasks.** Table 2 presents the performance of our proposed method, *LoRASC*, applied to LoRA across several instruction-following tasks. These instruction-following tasks are particularly challenging due to the weak correlation between the training data and the benchmarks, making them entirely out-of-domain tests. Despite this difficulty, our method achieved notable improvements across various evaluation metrics used in InstructEval and GSM8K. Furthermore, the design of slow-fast updates and noisy tuning still steadily enhanced the performance of cascading learning, further validating the effectiveness of our approach and motivation.

4.2.2 *LoRASC* for CLIP ViT-bigG

Experiment setting. For the ImageNet-1K visual classification task, to validate the transfer performance of our method on larger vision models, we selected CLIP ViT-bigG/14 as our pre-training backbone.We utilized the AdamW optimizer and a cosine scheduler, training for a total of 10 epochs on the ImageNet-1K training set. The batch size was fixed at 64, and the learning rate was chosen from $\{1\times10^{-4}, 5\times10^{-4}, 1\times10^{-3}\}$. For evaluation, we first test our model on the ImageNet-1K validation set using top-1 accuracy. To demonstrate the improvement in our method's transferability and robustness, we conducted further tests on robustness benchmarks from Mao et al. (2022) for transfer learning tasks.

Evaluation of *LoRASC* on ImageNet and Robustness Benchmarks. Table 3 showcases the performance of our proposed method, *LoRASC*, applied to LoRA on ImageNet-1K and several robustness benchmarks, including IN-V2, IN-C, IN-R, IN-A, IN-SK, and IN-ST. These benchmarks test the model's robustness and generalization ability beyond the standard ImageNet dataset. Our method demonstrates consistent improvements in top-1 ac-

Method	**ImageNet**	**IN-V2**	**IN-C**	**IN-R**	**IN-A**	**IN-SK**	**IN-ST**
LoRA	87.1	77.7	66.2	87.1	72.6	64.9	24.1
w/ LoRASC							
+ Cascade	87.1	77.5	66.7	88.5	**73.6**	65.4	24.3
++ Slow LoRA	87.7	78.3	**66.8**	88.1	73.4	65.2	24.1
+++ Noise Tuning	**87.8**	**78.4**	**66.8**	**88.7**	73.4	**65.5**	**24.4**

Table 3: Top-1 accuracy of various methods on ImageNet-1K and 6 robustness benchmarks. The table compares the baseline LoRA with our three proposed techniques. Our approach demonstrates improved robustness on the ViT-bigG model across all the evaluated benchmarks.

Experts	**RTE**	**DROP**	**WIC**	**BoolQ**	**ReCoRD**	**SST-2**	**SQuAD**
2	87.0	53.8	72.4	85.3	81.3	95.5	92.0
5	**88.1**	**56.7**	**72.6**	**87.4**	**82.2**	**96.1**	**92.9**
25	86.7	51.2	70.5	83.5	81.4	95.5	92.2
125	83.8	50.2	70.5	84.5	81.2	95.1	90.7
1250	83.8	49.4	69.4	85.3	81.1	92.9	88.1

Table 4: Evaluation with varing expert number of *LoRASC*. The highest average performance for each task is highlighted in bold.

curacy across all evaluated benchmarks. *LoRASC* consistently enhances the robustness and generalization of the ViT-bigG model across these challenging benchmarks, validating the effectiveness of cascading learning, slow-fast updates, and noisy tuning in improving model performance in diverse and robust scenarios.

4.3 Ablation Study and Analysis

Larger Ranks and Longer Epochs. As shown in Fig. 2, *LoRASC* consistently achieves more stable performance on datasets such as SQuAD, DROP, and GSM8K compared to both LoRA and COLA, which also employs a cascading strategy. This validates our motivation: *LoRASC* is a training strategy that retains LoRA's beneficial properties while seamlessly enhancing its fitting ability and robust generalization.

Ablation for *LoRASC* Expert Cascade Frequency. *LoRASC* defaults to updating once per epoch, as each expert completes training on the entire dataset within one epoch. In Table 4, we experimented with different update frequencies. In this setting, we trained for a total of 5 epochs, with each epoch consisting of 250 iterations, resulting in a total training period of 1250 iterations. The table shows that having 5 experts, corresponding to one new expert per epoch, yields the optimal performance. Interestingly, we observe that even with 1250 experts, where a new expert is initialized every iteration, the model still achieves highly competitive performance. In this extreme case, following Algorithm 1, the model cannot iterate the learning rate as each backpropagation step is immediately followed by the initialization of a new expert. We speculate that the strong generalization capability of slow cascading compensates for the weak fitting ability in this scenario. With 2 experts(one expert every 2.5 epochs), which aligns with COLA's default setting for this scenario, the performance is lower than *LoRASC*'s default of one expert per epoch. This may be due to the model being more prone to local optima after 2.5 epochs, which negatively impacts the effectiveness of slow cascading.

5 Conclusion

In this paper, we address the limitations of fine-tuning large pre-trained models, particularly the issue of overfitting and the high computational costs associated with transferring these models to niche tasks. We introduce *LoRASC*, an extension to the LoRA technique, which incorporates cascading learning, slow-fast updates, and noisy tuning to improve model expressiveness and generalization.

We provide a detailed analysis of *LoRASC* and demonstrate its effectiveness through extensive experiments in both the natural language processing (NLP) and computer vision (CV) domains. Our method consistently outperforms baseline LoRA models and their variants (LoRA+, Dora) across multiple datasets and tasks, including SuperGLUE, SQuAD, DROP, GSM8K, and various instruction-following benchmarks. Additionally, our method enhances the robustness and transferability of vision models on ImageNet and several robustness benchmarks.

Limitations

While *LoRASC* attempts to find a better balance between model convergence and generalization, it does not fundamentally resolve the issue. Our proposed mechanisms of slow-fast updating and noisy tuning can enhance model generalization and prevent overfitting; however, if the magnitude of these adjustments is too large, it may still lead to difficulties in model convergence. Therefore, it is necessary to adjust the α parameter in the slow-fast merging process and λ in the intensity of noise added to each expert according to the specific task. In our experiments, only a few candidate adjustments were needed to significantly outperform vanilla LoRA, yet this still incurs additional costs. Adaptive adjustment of these parameters according to the task is a direction for future work that we intend to explore.

Additionally, while this study only explores LoRA cascading learning for single training tasks and finds it to effectively enhance model performance, in practice, we could combine LoRA experts from multiple domains, similar to the MoLE (Huang and Wei, 2024) approach, to further improve model capabilities. In such cases, how to better perform slow cascading would be an interesting issue to address.

References

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

Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. 2022. Reproducible scaling laws for contrastive language-image learning. *arXiv preprint arXiv:2212.07143*.

Yew Ken Chia, Pengfei Hong, Lidong Bing, and Soujanya Poria. 2023. Instructeval: Towards holistic evaluation of instruction-tuned large language models. *arXiv preprint arXiv:2306.04757*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936.

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

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, pages 177–190. Springer.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, volume 23, pages 107–124.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. 2018. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. Lora+: Efficient low rank adaptation of large models. *arXiv preprint arXiv:2402.12354*.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. 2021a. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8340–8349.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Dan Hendrycks and Thomas Dietterich. 2019. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. 2021b. Natural adversarial examples. In *Proceedings of the IEEE/CVF conference on* *computer vision and pattern recognition*, pages 15262–15271.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Flat minima. *Neural computation*, 9(1):1–42.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2023. Lorahub: Efficient cross-task generalization via dynamic lora composition. *Preprint*, arXiv:2307.13269.

Shaohan Huang and Furu Wei. 2024. Mixture of lora experts. In *ICLR 2024*.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, 34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018, pages 876–885. Association For Uncertainty in Artificial Intelligence (AUAI). Publisher Copyright: © 34th Conference on Uncertainty in Artificial Intelligence 2018. All rights reserved.; 34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018 ; Conference date: 06-08-2018 Through 10-08-2018.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*.

Yuanzhi Li, Colin Wei, and Tengyu Ma. 2019. Towards explaining the regularization effect of initial large learning rate in training neural networks. *Advances in neural information processing systems*, 32.

Vladislav Lialin, Namrata Shivagunde, Sherin Muckatira, and Anna Rumshisky. 2023. Stack more layers differently: High-rank training through low-rank updates. *arXiv preprint arXiv:2307.05695*.

Yang Lin, Xinyu Ma, Xu Chu, Yujie Jin, Zhibang Yang, Yasha Wang, and Hong Mei. 2024. Lora dropout as a sparsity regularizer for overfitting control. *arXiv preprint arXiv:2404.09610*.

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

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. 2023. Fine-tuning language models with just forward passes. *Advances in Neural Information Processing Systems*, 36:53038–53075.

Xiaofeng Mao, Yuefeng Chen, Xiaodan Li, Gege Qi, Ranjie Duan, Rong Zhang, and Hui Xue. 2022. Easyrobust: A comprehensive and easy-to-use toolkit for robust computer vision. `https://github.com/alibaba/easyrobust`.

Xiangdi Meng, Damai Dai, Weiyao Luo, Zhe Yang, Shaoxiang Wu, Xiaochen Wang, Peiyi Wang, Qingxiu Dong, Liang Chen, and Zhifang Sui. 2024. Periodiclora: Breaking the low-rank bottleneck in lora optimization. *arXiv preprint arXiv:2402.16141*.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. 2019. Do imagenet classifiers generalize to imagenet? In *International conference on machine learning*, pages 5389–5400. PMLR.

Pengjie Ren, Chengshun Shi, Shiguang Wu, Mengqi Zhang, Zhaochun Ren, Maarten de Rijke, Zhumin Chen, and Jiahuan Pei. 2024. Mini-ensemble low-rank adapters for parameter-efficient fine-tuning. *arXiv preprint arXiv:2402.17263*.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in neural information processing systems*, volume 32.

Hanqing Wang, Bowen Ping, Shuo Wang, Xu Han, Yun Chen, Zhiyuan Liu, and Maosong Sun. 2024. Lora-flow: Dynamic lora fusion for large language models in generative tasks. *arXiv preprint arXiv:2402.11455*.

Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. 2019b. Learning robust global representations by penalizing local predictive power. *Advances in Neural Information Processing Systems*, 32.

Xi Wang, Laurence Aitchison, and Maja Rudolph. 2023. Lora ensembles for large language model fine-tuning. *arXiv preprint arXiv:2310.00035*.

Chuhan Wu, Fangzhao Wu, Tao Qi, Yongfeng Huang, and Xing Xie. 2022. Noisytune: A little noise can help you finetune pretrained language models better. *arXiv preprint arXiv:2202.12024*.

Wenhan Xia, Chengwei Qin, and Elad Hazan. 2024. Chain of lora: Efficient fine-tuning of language models via residual learning. *arXiv preprint arXiv:2401.04151*.

Zeke Xie, Issei Sato, and Masashi Sugiyama. 2020. A diffusion theory for deep learning dynamics: Stochastic gradient descent exponentially favors flat minima. *arXiv preprint arXiv:2002.03495*.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. 2023. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning. *arXiv preprint arXiv:2308.03303*.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint arXiv:1810.12885*.