SiRA: Sparse Mixture of Low Rank Adaptation

Anonymous ACL submission

Abstract

Parameter Efficient Tuning (PET) techniques such as Low-rank Adaptation (LoRA) are effective methods to adapt Large Language Models to downstream tasks. We propose Sparse mixture of low Rank Adaption (**SiRA**), which uses Sparse Mixture of Experts (SMoE) by enforcing conditional computation with top k LoRA weights. SiRA is optimized through a combination of training techniques, including an auxiliary loss encouraging load balancing, a capacity limit which restricts the maximum number of tokens each expert can process, and novel expert dropout on top of the gating network. Through extensive experiments, we show that SiRA performs better than LoRA and other mixture of expert approaches across different single-task and multiple-task settings. Results show SiRA has more orthogonal low rank spaces and consumes less computing resources compared to other MoE variants.

1 Introduction

Large Language Models (LLMs) have demonstrated impressive capabilities in a wide range of tasks. To adapt these general-purpose models to downstream low resource tasks remains important. To this end, parameter efficient tuning (PET) (Hu et al., 2021; Li and Liang, 2021; Lester et al., 2021; Houlsby et al., 2019; Zhang et al., 2023b; Zaken et al., 2021; Chen et al., 2022), which introduces task specific weights to the frozen foundation model for gradient descent, has been widely adopted with the merit of avoiding the catastrophic forgetting (Luo et al., 2023) of fine-tuning.

However, previous study (Chen et al., 2022) and our findings in Figure 2 show PET is more stable with fewer parameters and more parameters may lead to worse quality. This poses a hidden bottleneck for model quality even when we have enough computation budget. Thereby it remains challenging to introduce capacity under PET in a more efficient way.

We are inspired by recent advancements of the Sparse Mixture of Experts (SMoE) (Bengio et al., 2015; Shazeer et al., 2017; Lepikhin et al., 2020). Such conditional computation efficiently scales model capacity without large increases in training or inference costs. Yet the power of sparse and dynamic computation is less investigated under the PET scenario. In recent years, several recent works have proposed mixture-of-expert models on top of parameter-efficient tuning. Adamix (Wang et al., 2022) uses random gating and does not learn specialized experts. MoLoRA (Zadouri et al., 2023) applies the dense MoE on the top of LoRA, where all experts are averaged using a learned gating. Such dense computation brings inefficiency compared to SMoE which conserves resources and inference computation with the same parameter count by only using a subset of experts. We put a more broad related work discussion in Section 6.5.

To this end, we present **SiRA**, the Sparse Mixture of Low Rank Adaptation. SiRA is building SMoE upon the state of the art PET approach LoRA (Hu et al., 2021). Our research demonstrates that a strategic combination of capacity constraints and expert utilization loss is the key to realizing the potential of sparse LoRA. Additionally, we present a novel dropout mechanism that combats overfitting, proving essential for SiRA's superior performance. This is non trivial since the sheer diversity of routing strategies (Roller et al., 2021; Fedus et al., 2022; Lepikhin et al., 2020; Zhou et al., 2022; Puigcerver et al., 2023) and the lack of clarity on their effectiveness with PET posed a significant challenge, especially with common SMoE limitations like token dropping (Puigcerver et al., 2023) and overfitting (Elbayad et al., 2022). The fact that the MoLoRA paper attempted sparse MOE, but without convincing results, underscores the significance of our findings.

We conducted extensive experiments which verify that the performance of SiRA, is better than LoRA (Hu et al., 2021), its MoE variants Adamix (Wang et al., 2022), MoLoRA (Zadouri et al., 2023) and other PET approaches across a wide range of single task and multitask benchmarks with less TPU hours compared to other MoE variants. Our ablation study further confirmed the effectiveness of the three ingredients as well as the generality of our methods. We also explain the effectiveness of SiRA by empirically showing it facilitates multiple orthogonal low rank spaces to capture diverse knowledge.

2 Sparse Mixture of Low Rank Adaptation

To increase the capacity of LoRA (Hu et al., 2021) using Mixture of Experts (MoE) without adding too much computational cost, we propose Sparse Mixture of Experts of Low Rank Adaptation (SiRA), which leverages multiple lightweight LoRA adaptors as experts while enforcing sparsity when using the expert modules.

Figure 1 shows an illustration of SiRA. The MoE layer for the adapter consists of E experts, each with their own LoRA weights, $W_1, ..., W_E$. W_k is the product of two low rank matrices $W_k = B_k A_k$. We also assume the base foundation model has W_0 as its frozen weight, which represents either query, key, value, or output projection. We replace the attention projection in each layer of the network with this computation. Our parameter initialization and update method for LoRA. B_k is initialized as Gaussian while A_k is initialized as zeros. We freeze the base model and update the LoRA weight through gradient descent, detailed in Appendix 6.3.

Expert Gating To reduce the computational cost, SiRA only activates a subset of all the expert modules. Formally, during each forward pass, we select K out of E experts using the output scores of a gating network θ_g. The process is mathematically expressed as Equation (1) and (2), where s denotes the token index of the sequence x and $G_{s,e}$ is the gating network output at s-th token e-th experts. The TopK operation renormalizes the gate weights to sum to 1.0.

$$G(x_s) = \text{TopK}(\text{softmax}(\theta_g^T x_s)) \quad (1)$$

$$y_s = \sum_{e=1}^{E} G_{s,e} W_e(x_s) + W_0(x_s) \quad (2)$$

Experts Dropout To avoid the situation that certain experts are over or under-trained, we propose gate dropout. Specifically, we introduce dropout to the gating output G as shown in Equation 3.

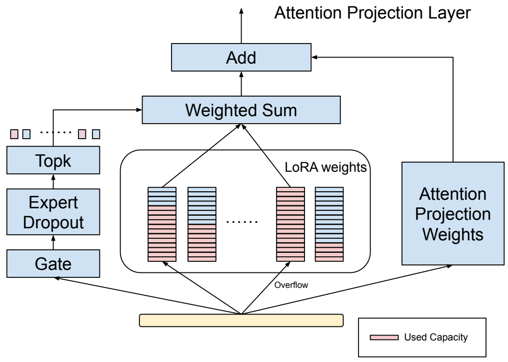

Figure 1: SiRA: Sparse Gated Mixture of LoRA.

$$G(x_s) = \text{TopK}(\text{Dropout}(\text{softmax}(\theta_g^T x_s))) \quad (3)$$

Expert Token Capacity We enforce the capacity constraints for experts following GShard (Lepikhin et al., 2020). Specifically, we restrict that the number of tokens processed by each expert should not exceed a predefined threshold. Once the capacity is reached, the expert simply drops the overflow tokens. If all K experts reach their token capacity before all tokens in a training example are processed, the rest of the tokens will only be encoded using the frozen model parameter W_0.

Auxiliary Loss Beside the normal LM loss, we use the auxiliary loss term to encourage load balancing among different experts following (Shazeer et al., 2017; Lepikhin et al., 2020). We denote the total number of tokens to be S, and there are E experts. We also denote the number of tokens routed to expert e as c_e. By using the mean gates per expert $m_e = Mean_s(\text{Dropout}(\text{softmax}(\theta_g^T x_s)))$ as a differentiable approximation, we express the aux loss in Equation 4.

$$l_{aux} = \frac{1}{E} \sum_{e=1}^{E} \frac{c_e}{S} * m_e \quad (4)$$

3 Experiments

3.1 Evaluation Setup

Baselines and Experiment Configs We specifically compare our model with the Prompt Tuning (Lester et al., 2021), IA3 (Liu et al., 2022), standard LoRA (Hu et al., 2021), Adamix (Wang et al., 2022) and MoLoRA (Zadouri et al., 2023). Note that other adapter approaches are not compared with the SiRA approach is orthogonal and

Approach	δ Params	FinQA (EN)		ForumSum (EN)			SP (SW)	QA-in (SW)	NER (SW)	SP (BN)	QA-in (BN)	QA-cross (BN)
		em	f1	bleurt	rougeL	f1	accuracy	f1	span-f1	accuracy	f1	f1
PromptTuning	0.0024%	4.0	4.0	95.80	28.94	18.90	0.22	63.93	45.01	0.76	62.83	55.07
IA3	0.0140%	1.8	2.1	96.98	32.81	23.06	21.65	72.04	86.78	22.87	69.06	64.55
LoRA	0.0419%	5.0	5.6	96.70	33.97	23.54	27.63	82.08	88.95	33.52	80.34	76.81
LoRA(R=8)	0.0838%	3.0	3.2	96.53	34.67	23.98	31.27	81.99	89.41	35.84	74.96	77.32
LoRA(R=16)	0.1676%	2.4	2.4	96.46	34.43	23.12	31.57	81.47	89.14	36.06	72.69	**78.94**
LoRA(R=32)	0.3353%	2.2	2.2	96.32	34.11	22.64	29.84	78.55	88.58	33.27	70.01	77.07
LoRA(R=64)	0.6706%	1.2	1.2	96.21	33.48	23.37	24.28	79.37	87.81	28.54	69.06	69.37
Adamix	0.6706%	5.6	6.0	95.95	35.10	23.88	**33.22**	81.24	89.00	**39.03**	81.70	76.07
MoLoRA	0.7264%	5.6	6.4	97.05	34.37	24.79	32.50	82.33	89.33	36.28	79.06	76.75
SiRA	0.7264%	**5.8**	**6.6**	**97.14**	**35.67**	**25.83**	32.52	**83.00**	**89.95**	38.61	**82.10**	76.93

Table 1: Performance Comparison For Single Tasks

could be applied on top of them as well. We choose PALM2-FLAN (Passos et al., 2023) as the foundation model [1]. We follow the default configurations in (Hu et al., 2021) to inject LoRA weights into the attention projections and set the intrinsic rank as 4. Larger intrinsic ranks are also applied to LoRA for fair comparisons. We use 16 experts by default across all MoE based approaches. We set prompt length as 25 for prompt tuning following (Lester et al., 2021). We use the XXS model size unless otherwise specified. We tuned the hyperparameters to find the optimal point that best suits the baselines. To tune SiRA, we freeze those hyperparameters and only tune extra hyperparameters which exist only in SIRA. See Appendix 6.3 for more hyperparameters.

Datasets and Metrics We evaluate on the datasets which the model wasn't pre-trained on:[2]

XTREME-UP (Ruder et al., 2023) is a multilingual multitask dataset. We choose two of the underrepresented languages—Swahili (SW) and Bengali (BN)—and evaluate on several NLP tasks. We follow Ruder et al. (2023) for each task's splits and evaluation metrics.

FinQA (Chen et al., 2021) is a QA dataset in the financial domain which requires complex reasoning. The answers in the dataset are DSL programs. We only evaluate metrics based on surface form matching, *i.e.*, exact match and F1 scores.

ForumSum (Khalman et al., 2021) is a diverse conversation summarization dataset with human written summaries. We report BLEURT (Sellam et al., 2020), ROUGEL, and F1 scores.

[1]We choose the instruction tuned model instead of the pretrain model as base model which is more practical when applying LoRA. Thus pretrained models like LLAMA or pretrained GPT is not considered here.

[2]Since our base model (Chung et al., 2022) had been exposed to many public datasets during training, we choose datasets that are not consumed yet.

Models	bleurt	rougeL	f1
Prompt Tuning	96.34	30.36	20.54
IA3	96.94	33.93	24.27
LoRA	96.91	36.59	26.60
LoRA(r=8)	97.01	36.78	26.89
LoRA(r=16)	96.80	36.63	25.86
LoRA(r=32)	96.50	36.50	25.78
LoRA(r=64)	96.13	35.96	25.02
Adamix	96.52	36.79	25.80
MoLoRA	97.02	36.77	26.96
SiRA	**97.35**	**37.08**	**27.56**

Table 2: Results with PALM2-XS on ForumSum.

3.2 Performance of SiRA

We evaluate the single tasks performance in Table 1. We also conducted experiments on two multitask settings on language swahili (SW) and bengali(BN), and two multilingual settings for QA in languages task (QA-in) and QA across languages task(QA-cross). We report numbers in Table 7 and Table 8. Results are averaged from 3 experiments.

Prompt tuning and IA3 generally perform worse than LoRA based approaches with fewer parameters. For LoRA, $R = 4$ achieves better performance for the multitask and multilingual settings. Although for some single tasks, $R = 8$ or $R = 16$ achieves better results. But further increasing R will decrease performance in all cases. This suggests that more parameters does not necessarily mean quality gains.

In general, the MoE based approaches can achieve better performance than LoRA. Notably when compared to MoLoRA, SiRA achieves constantly better performance among all the tasks, which demonstrates that "sparse" MoE is better than "full". Adamix shows some small advantage on the Semantic Parsing task, but overall loses to SiRA across all other tasks. SiRA outperforms all other baselines in most single and multi tasks settings. Note that SiRA uses less than 1% extra parameters compared to the foundation model, causing limited memory and computation overhead.

PALM2-XS Backbone Model We also change our base model from Flan-PALM2-XXS to Flan-

Configs	bleurt	rougeL	f1
R=4, K=2, C=2, E=16	96.87	34.51	24.73
R=4, K=4, C=4, E=16	96.60	34.66	25.34
R=4, K=6, C=6, E=16	96.75	34.73	24.55
R=4, K=8, C=8, E=16	96.76	**35.31**	**25.64**
R=4, K=10, C=10, E=16	**97.51**	35.10	25.19
R=4, K=12, C=12, E=16	96.96	34.49	24.24
R=4, K=4, C=2, E=16	96.33	34.15	24.13
R=4, K=4, C=4, E=16	96.60	34.66	25.34
R=4, K=4, C=6, E=16	97.14	**35.67**	**25.83**
R=4, K=4, C=8, E=16	**97.31**	34.97	25.24
R=4, K=4, C=10, E=16	97.25	34.75	25.57
R=4, K=4, C=12, E=16	96.50	34.44	23.94
R=2, K=4,C=4,E=16	96.20	34.70	24.76
R=4, K=4,C=4,E=16	96.60	34.66	**25.34**
R=6, K=4,C=4,E=16	**97.05**	**34.75**	25.02
R=8, K=4,C=4,E=16	96.90	34.68	24.19
R=4, K=4,C=4,E=8	96.58	34.61	24.51
R=4, K=4,C=4, E=16	96.60	34.66	**25.34**
R=4, K=4,C=4, E=24	96.68	**34.85**	24.77
R=4, K=4,C=4, E=32	**96.69**	34.78	24.05

Table 3: Self ablations on the hyper-parameter Rank(R), topK(K), expert capacity(C), and Expert(E) on ForumSum.

Approach	bleurt	rougeL	f1
SiRA	**97.14**	**35.67**	**25.83**
- aux loss	96.37	35.09	25.11
- Expert Dropout	97.09	34.73	24.55
+ SMoE-Dropout	96.30	34.24	24.32

Table 4: Gating ablations on ForumSum.

PALM2-XS, which has a much larger size. We report the ForumSum results in Table 2. When switching to a larger LLM, the overall performance is better since the base is stronger. However, the overall trend and conclusion has not changed: SiRA still outperforms all other baselines. This shows SiRA can generalize to larger backbones.

3.3 Ablation Study

Hyper-parameter Ablations We choose a simple config (R=4, K=4, C=4, E=16) and then change each of them while keeping the rest. We share the ablations on ForumSum in Table 3. An interesting finding is that increasing the number of experts or the capacity per expert will not always increase the scores, which justifies why the full MoE based approach is not as good as SiRA. The overall performance is slightly better with larger R when R < 8. Besides, we found that performance improves when we change E=8 to E=16, but further increasing E does not help. These findings suggest that a proper value of these hyper-parameters need to be found within a reasonable range but they are not that sensitive.

Gating ablations We compare SiRA with 3 more cases: 1) removing the aux loss, 2) removing the gate dropout, and 3) using a static routing based dropout SMoE-Dropout (Chen et al., 2023a) instead. Results in Table 4 suggested that the learned gating is better than static, and both the gate dropout and aux loss help the performance.

Approach	Steps/Sec	Converge steps(k)	TPU time(h)
Lora	1.14	2	0.487
Adamix	1.02	40	10.89
MoLoRA	0.09	1.6	4.94
SiRA	0.30	2	1.85

Table 5: Resource consumption (Training) comparison

Approach	Cosine Similarity
Adamix	0.23500
MoLoRA	0.00700
SiRA	**0.00028**

Table 6: Diversity of Low Rank Spaces

3.4 Comparisons of Resource Consumption

We share resource consumption stats in Table 5. SiRA achieves higher steps per second than MoLoRA because SiRA only uses K experts each layer instead of all. Adamix needs many more steps to converge than other methods possibly because of random token distribution. Overall SiRA consumes less TPU time than Adamix and MoLoRA.

3.5 Analysis of Expert Weights

We analyze the orthogonality of expert weights following recent works (Wang et al., 2023). In each layer we measure the absolute value of average cosine similarity between each pair of expert weights. We compute each expert weight by multiplying the low rank matrices to produce $W_e = A_e * B_e$. We share the cosine similarity in Table 6. The cosine similarity averaged over the layers for SiRA is significantly lower than other MoE based approaches and pretty close to 0. This indicates that the experts in our method learn more diverse concepts than other MoE based approaches. This is beneficial as Liu et al. (2023a) show. Interestingly, we also found the SiRA does not learn to route different tasks to different experts. We provide further analysis in Appendix 6.4.

4 Conclusion

This paper introduced SiRA, a Sparse Mixture of Expert variant of LoRA. By leveraging sparse and dynamic computation with a few training optimizations, SiRA achieved better performance than LoRA and other baselines across different tasks while consuming less resources. Our analysis suggested that SiRA provides more orthogonal low rank sub-spaces than others.

5 Limitation

We did not report results on open sourced models such as LLaMA, GPT, or Roberta. As we choose an instruction tuned model as the base model which is more practical for adding LoRA. Thus pretrained LLaMA or GPT is out of scope. And the affiliation of the authors of the paper is not permitted to run LLaMA2 models due to Meta's license, even for benchmarking in a research paper. The main focus of our experiments is on generative tasks on top of LLMs which is more trendy in recent years, so we did not choose smaller bert based models like Roberta.

References

Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. 2015. Conditional computation in neural networks for faster models. *arXiv preprint arXiv:1511.06297*.

Guanzheng Chen, Fangyu Liu, Zaiqiao Meng, and Shangsong Liang. 2022. Revisiting parameter-efficient tuning: Are we really there yet? *arXiv preprint arXiv:2202.07962*.

Tianlong Chen, Zhenyu Zhang, Ajay Jaiswal, Shiwei Liu, and Zhangyang Wang. 2023a. Sparse moe as the new dropout: Scaling dense and self-slimmable transformers. *arXiv preprint arXiv:2303.01610*.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zitian Chen, Yikang Shen, Mingyu Ding, Zhenfang Chen, Hengshuang Zhao, Erik G. Learned-Miller, and Chuang Gan. 2023b. Mod-squad: Designing mixtures of experts as modular multi-task learners. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 11828–11837. IEEE.

Joon-Young Choi, Junho Kim, Jun-Hyung Park, Wing-Lam Mok, and SangKeun Lee. 2023. Smop: Towards efficient and effective prompt tuning with sparse mixture-of-prompts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14306–14316.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Maha Elbayad, Anna Sun, and Shruti Bhosale. 2022. Fixing moe over-fitting on low-resource languages in multilingual machine translation. *arXiv preprint arXiv:2212.07571*.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1):5232–5270.

Yingbo Gao, Christian Herold, Zijian Yang, and Hermann Ney. 2022. Revisiting checkpoint averaging for neural machine translation. *arXiv preprint arXiv:2210.11803*.

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

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*.

Misha Khalman, Yao Zhao, and Mohammad Saleh. 2021. ForumSum: A multi-speaker conversation summarization dataset. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4592–4599, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Sneha Kudugunta, Yanping Huang, Ankur Bapna, Maxim Krikun, Dmitry Lepikhin, Minh-Thang Luong, and Orhan Firat. 2021. Beyond distillation: Task-level mixture-of-experts for efficient inference. *arXiv preprint arXiv:2110.03742*.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. 2021. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*, pages 6265–6274. PMLR.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Boan Liu, Liang Ding, Li Shen, Keqin Peng, Yu Cao, Dazhao Cheng, and Dacheng Tao. 2023a. Diversifying the mixture-of-experts representation for language models with orthogonal optimizer.

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

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2023b. Gpt understands, too. *AI Open*.

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

Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Wen-tau Yih, and Madian Khabsa. 2021. Unipelt: A unified framework for parameter-efficient language model tuning. *arXiv preprint arXiv:2110.07577*.

Alex Passos, Andrew Dai, Bryan Richter, Christopher Choquette, Daniel Sohn, David So, Dmitry (Dima) Lepikhin, Emanuel Taropa, Eric Ni, Erica Moreira, Gaurav Mishra, Jiahui Yu, Jon Clark, Kathy Meier-Hellstern, Kevin Robinson, Kiran Vodrahalli, Mark Omernick, Maxim Krikun, Maysam Moussalem, Melvin Johnson, Nan Du, Orhan Firat, Paige Bailey, Rohan Anil, Sebastian Ruder, Siamak Shakeri, Siyuan Qiao, Slav Petrov, Xavier Garcia, Yanping Huang, Yi Tay, Yong Cheng, Yonghui Wu, Yuanzhong Xu, Yujing Zhang, and Zack Nado. 2023. Palm 2 technical report. Technical report, Google Research.

Edoardo Maria Ponti, Alessandro Sordoni, Yoshua Bengio, and Siva Reddy. 2023. Combining parameter-efficient modules for task-level generalisation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 687–702.

Joan Puigcerver, Carlos Riquelme, Basil Mustafa, and Neil Houlsby. 2023. From sparse to soft mixtures of experts. *arXiv preprint arXiv:2308.00951*.

Stephen Roller, Sainbayar Sukhbaatar, Jason Weston, et al. 2021. Hash layers for large sparse models. *Advances in Neural Information Processing Systems*, 34:17555–17566.

Sebastian Ruder, Jonathan H. Clark, Alexander Gutkin, Mihir Kale, Min Ma, Massimo Nicosia, Shruti Rijhwani, Parker Riley, Jean-Michel A. Sarr, Xinyi Wang, John Wieting, Nitish Gupta, Anna Katanova, Christo Kirov, Dana L. Dickinson, Brian Roark, Bidisha Samanta, Connie Tao, David I. Adelani, Vera Axelrod, Isaac Caswell, Colin Cherry, Dan Garrette, Reeve Ingle, Melvin Johnson, Dmitry Panteleev, and Partha Talukdar. 2023. Xtreme-up: A user-centric scarce-data benchmark for under-represented languages.

Thibault Sellam, Dipanjan Das, and Ankur P Parikh. 2020. Bleurt: Learning robust metrics for text generation. In *Proceedings of ACL*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. 2023. Orthogonal subspace learning for language model continual learning. *arXiv preprint arXiv:2310.14152*.

Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022. Adamix: Mixture-of-adaptations for parameter-efficient model tuning. *arXiv preprint arXiv:2210.17451*.

Ted Zadouri, Ahmet Üstün, Arash Ahmadian, Beyza Ermiş, Acyr Locatelli, and Sara Hooker. 2023. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning. *arXiv preprint arXiv:2309.05444*.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*.

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

Renrui Zhang, Jiaming Han, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, Peng Gao, and Yu Qiao. 2023b. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. 2022. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114.

6 Appendix

6.1 Effect of LoRA rank

We investigate the effect of LoRA rank in Figure 2.

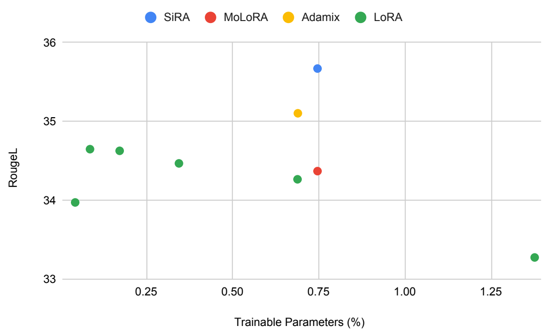

Figure 2: SiRA vs LoRA on ForumSum Task. We increase the rank of LoRA (rank=4, 8, 16, 32, 64, 128) and report the RougeL as a metrics. Notably increasing the rank does does not help the performance. SiRA (rank=4) can achieve higher quality by leveraging the sparse mixture of experts.

6.2 Multitask Results

We put the multitasking and multilugual results in Table 7 and Table 8.

6.3 Training and Model selection

During supervised finetuning, SFT, we use 8 Tensor Processing Units (TPU) V3 chips for fine-tuning. The batch size is 64, and the maximum training step is 30000. We use the Adafactor optimizer (Shazeer and Stern, 2018) with a learning rate of 0.0005. Both the input and output sequence lengths are set to match the dataset requirements. The training dropout rate is 0.05. The expert dropout rate is set to 0.5. In our experiments, Adamix, MoLoRA and SiRA are based on the same hyper-parameter setups to be fair. For our method, we only tune the hyperparameters which are specific to our method, for example the gate dropout rate. We decode on the validation sets of each task every 100 steps. And we report test results from the best checkpoints according to the validation scores. For multitask results, the checkpoint is picked by the average of each tasks metrics. For the reported numbers in section 3.2, we use topk $K = 4$ as default. Yet we found $K = 8$ is better for BN multitask and QA (in-lang) multilingual setting, and $K = 12$ better for QA (cross-lang) experiments. Capacity wise, $C = K$ yields constant good results across experiments, yet $C = K + 2$ achieves better results for ForumSum.

Table 7: Performance Comparison For Multi Tasks

Approach	δ params	SW Multitask				BN Multitask			
		SP(accuracy)	QA-in(f1)	NER(span-f1)	Average	SP(accuracy)	QA-in(f1)	QA-cross(f1)	Average
PromptTuning	0.0024%	0.59	65.34	0.21	29.21	1.05	61.04	68.75	43.62
IA3	0.0140%	18.98	64.58	83.86	55.81	20.87	61.63	68.44	50.31
LoRA	0.0419%	28.06	77.71	88.28	64.69	32.06	79.27	75.03	62.12
LoRA(R=8)	0.0838%	29.71	74.13	88.69	64.17	35.65	76.17	72.17	61.33
LoRA(R=16)	0.1676%	32.52	71.55	88.92	64.33	34.41	72.69	71.70	59.60
LoRA(R=32)	0.3353%	29.08	66.48	88.39	61.32	33.87	67.16	70.49	57.17
LoRA(R=64)	0.6706%	27.11	67.29	85.09	59.83	30.28	68.37	71.39	56.68
Adamix	0.6706%	**35.14**	76.99	89.01	67.10	**38.41**	79.49	75.09	64.33
MoLoRA	0.7264%	33.44	79.91	88.92	65.66	35.98	78.14	**76.37**	63.49
SiRA	0.7264%	33.98	**81.26**	**89.04**	**68.10**	37.71	**82.17**	75.50	**65.13**

Table 8: Performance Comparison for Multilingual Tasks with diverse LoRA variants.

Approach	δ params	QA-in (9)	QA-cross (25)
PromptTuning	0.0024%	74.55	62.05
IA3	0.0140%	80.68	61.70
LoRA	0.0419%	85.09	69.41
LoRA(R=8)	0.0838%	85.12	69.94
LoRA(R=16)	0.1676%	84.68	69.50
LoRA(R=32)	0.3353%	82.43	66.38
LoRA(R=64)	0.6706%	80.26	64.10
Adamix	0.6706%	84.75	70.42
MoLoRA	0.7264%	85.14	70.70
SiRA	0.7264%	**86.38**	**70.86**

6.4 Does the gate learn task specifies

We use the Swahili multitask experiment to study what the gate is learning. We measure the average entropy of each gate weight distribution before TopK is applied. The average entropy for the QA (in language) task decreases from 1.6 to 1.13 nats during training. This indicates that the model learns to give certain gates more weight as it trains.

We also measure the average correlation coefficients between each task index and each gate index similar to (Chen et al., 2023b). We convert the task index to a one hot encoding for this. At the end of training, the average correlation was about .025, which is not significant. The correlation between gates and languages in the multilingual experiment is not significant either. This suggests that our gating mechanism does not learn to route different tasks to different gates.

6.5 Related Works

Parameter Efficient Tuning (PET) Parameter Efficient Tuning has a variety of flavors such as Adapters (Houlsby et al., 2019), Prefix Tuning (Li and Liang, 2021; Liu et al., 2021), Prompt Tuning (Lester et al., 2021), P-tuning (Liu et al., 2023b), attention-injection (Zhang et al., 2023b), LoRA (Hu et al., 2021; Dettmers et al., 2023), and combinations of PET methods (Mao et al., 2021). In this paper, our focus is on LoRA as it has been found to achieve better results, although the methods could be applied to other flavors as well. Some previous works such as AdaLoRA (Zhang et al., 2023a) tried to solve the problem of allocating the parameter budget in the low budget settings. Our method, on the other hand, is targeting the problem of scaling up the parameter through dynamic computing.

Mixture of Experts (MoE) Leveraging Mixture of Experts in neural networks has been extensively studied, with different approaches to find the optimal assignment between expert and tokens, including reinforcement learning (Bengio et al., 2015), linear programs (Lewis et al., 2021), fixed rules (Roller et al., 2021), top-1 gating (Fedus et al., 2022), top-2 gating (Lepikhin et al., 2020), top-k gating (Shazeer et al., 2017), reverse expert choosing (Zhou et al., 2022), and soft assignment (Puigcerver et al., 2023). However, most of the previous works focus on foundation model architectures, where MoE is applied on the Feed-Forward parts of the transformer layer.

Several recent works have proposed mixture-of-expert models on top of parameter-efficient tuning(Choi et al., 2023; Wang et al., 2022; Zadouri et al., 2023). SMoP (Choi et al., 2023) focuses on prompt tuning and uses per sequence routing, while our work is using per token routing based on LoRA. Adamix (Wang et al., 2022) randomly chooses an expert in training and averages all the experts during inference. This method is similar to checkpoint averaging (Gao et al., 2022) as the experts are randomly chosen and don't learn to specialize. It also empirically has significant longer training time caused by uniform token distributing. MoLoRA (Zadouri et al., 2023) applies the dense MoE on the top of LoRA, where all experts are averaged using a learned gating. Compared to this work, our method can achieve better efficiency since we only use a subset of experts which conserves training resources and inference computation with the same parameter count. The MoLoRA paper attempted sparse MOE, but without convincing results, presumably since they did not use the dropout and capacity constraints we describe in our work.

Multitask Parameter Efficient Tuning Another track of the MoE work is for multitasking, such as Task-MoE (Kudugunta et al., 2021) and Skill Selection (Ponti et al., 2023). These approaches assume the external task-id as an extra input for training and inference. Although we experiment with MoE in multitask settings, it does not require the task-id of inputs. Interestingly, our experiments suggest that the gating does not learn anything regarding the task specific or language specific information to distribute the token, demonstrating the fundamental difference from the above approaches.