EWoRA: Expert Weighted Low-Rank Adaptation for Heterogeneous Data

Harsh Kohli *
The Ohio State University
kohli.120@osu.edu

Helian Feng
Amazon
hlfeng@amazon.com

Lenon Minorics
Amazon
minorics@amazon.de

Bhoomit Vasani
Amazon
vbhoomit@amazon.com

Cynthia He
Amazon
xih@amazon.com

Ali Kebarighotbi
Amazon
alikeba@amazon.com

Abstract

Low-Rank Adaptation (LoRA) has emerged as a widely adopted parameter-efficient finetuning (PEFT) approach for language models. By restricting weight updates to a low-rank subspace, LoRA achieves cost-effective finetuning of large, generalist models to more specialized target domains. While LoRA achieves impressive results for a variety of individual downstream tasks, it struggles to capture the diverse expertise needed when presented with a more heterogeneous finetuning corpus. To address this, we propose Expert Weighted Low-Rank Adaptation (EWoRA), a novel LoRA variant that partitions a rank-r adapter into n independent adapters of rank r/n. A lightweight "routing" matrix $\mathbf{W}_r \in \mathbb{R}^{r\times n}$ aggregates the outputs of these adapters by learning specialized weights for each context. Experiments show EWoRA improves performance over LoRA when finetuning on heterogeneous data while generally matching or exceeding LoRA performance on individual finetuning tasks under the same low-rank parameter budget.

1 Introduction

Large Language Models (LLMs) deployed in real-world applications are typically pretrained on broad corpora and then fine-tuned for specialized domains. This process can be prohibitively expensive, especially as model sizes scale to billions of parameters, PEFT methods like LoRA (Hu et al., 2021) have emerged as a more accessible solution by introducing a small number of trainable parameters while keeping the base model frozen. Besides supervised finetuning, LoRA has been studied across various adaptation settings such as continual pretraining (Jiang et al., 2024; Pezeshkpour and Hruschka, 2025; Mao et al., 2024). However, most existing formulations assume homogeneous task data or focus on a single target domain. In contrast, data encountered in practical applications may often exhibit heterogeneity, and require multiple domains of expertise. For instance, a deployed model may be required to tackle tasks involving natural language understanding, mathematical reasoning, code generation, and general knowledge, all in a single workflow. To emulate such scenarios, we evaluate LoRA on a diverse mix of finetuning datasets of varying complexity. Our experiments reveal a substantial performance drop of LoRA when finetuning on the same set of diverse tasks collectively compared to training on each task separately.

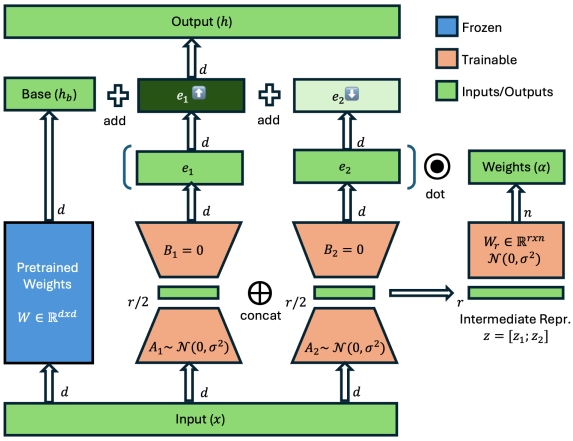

Figure 1: An overview of EWoRA with 2 experts (n=2). The representations from two rank-$\frac{r}{2}$ experts are concatenated and input to a router which assigns scores to each expert. In this illustration, the output of the first expert, E_1 is given a higher weight (darker shade).

To address these challenges, we propose **EWoRA: E**xpert-**W**eighted **Lo**w **R**ank **A**daptation, a novel LoRA variant tailored for heterogeneous data. As illustrated in Figure 1, EWoRA decomposes a rank-r LoRA adapter into n independent rank-r/n experts. A minimal routing matrix $\mathbf{W}_r \in \mathbb{R}^{r\times n}$, processes the concatenated intermediate representations from the lower projection matrix ($\mathbf{A}$), generating a set of weights to dynamically attend to the outputs of the individual experts. Op-

*Work done while at Amazon.

Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 1729–1737

erating within the low-rank subspace, the router incurs negligible overhead in terms of parameters. In EWoRA, individual experts specialize in different tasks or domains, while the router learns to weight the experts appropriately based on context. We also posit that this leads to a more efficient utilization of the parameters of ($\mathbf{A}$) through indirect supervision of intermediate low-rank representations. Thus, our approach may also help alleviate the asymmetry impact observed when finetuning the LoRA projection matrices (Zhu et al.).

Empirical results show that EWoRA matches standard LoRA on single-task benchmarks and significantly outperforms it when jointly finetuning on multiple tasks. We further compare against AdaLoRA (Zhang et al., 2023b), an adaptive-rank extension of LoRA, observing consistent gains in both single-task and mixed-task settings. Although EWoRA differs in both motivation and structure from model merging approaches, we also include model merging baselines in our work. Finally, we analyze the router's behavior under mixed-task finetuning, demonstrating how expert weighting evolves across layers and varies by target module.

2 Related Work

Low-rank Adaptation Methods. While LoRA has become the de facto standard for efficient finetuning of LLMs, numerous variants have emerged that improve LoRA performance in certain settings. DoRA (yang Liu et al., 2024), decomposes the pretrained weight into magnitude and direction components, utilizing LoRA specifically for directional updates to enhance learning capacity and training stability. VeRA (Kopiczko et al., 2024) improves efficiency further by sharing low-rank projection matrices across layers and only learning two additional scaling vectors per layer. Delta-LoRA (Zi et al., 2023) proposes a method to update the pretrained weight matrix (W) without explicitly computing its gradients. Several approaches have proposed efficiency or performance improvements by leveraging the asymmetry in the low-rank matrices of the adapters through freezing of the down-matrix ($\mathbf{A}$) or separate learning rates for the two projection matrices (Zhu et al.; Zhang et al., 2023a; Hayou et al., 2024). Other methods have sought to improve performance through adaptive rank utilization across layers (Zhang et al., 2023b), dynamic rank selection at inference (Valipour et al., 2023) or high-rank adaptation through a single, square trainable matrix (Jiang et al., 2024).

Model-merging Methods. TIES (Yadav et al., 2023) and DARE (Yu et al.) are popular approaches to combine abilities from multiple homologous models while reducing interference and eliminating redundant parameters. Such methods are static in nature - using a linear or SVD combination of task vectors. More recently, MoErging methods (Yadav et al., 2024) have been proposed for dynamic routing among specialized models or adaptive combination on a per-task or per-query basis. Among these, MoLE (Wu et al., 2024) and X-LoRA (Buehler and Buehler, 2024) are most reminiscent of EWoRA and learn a dense or sparse routing among experts. However, this approach typically requires *a priori* knowledge of the different domains in the finetuning corpus, as well as careful segregation of the data. EWoRA, on the other hand, does not require explicit domain partitioning or multiple separate finetuning runs. Also, as noted previously, our router takes as input the low-rank intermediate representation and utilizes only a fraction of the parameters of other routing methods that typically act on full-rank input or output representations. We further elaborate on differences between EWoRA and related MoErging methods in Appendix A.

3 Method

EWoRA splits the standard down ($\mathbf{A}$) and up ($\mathbf{B}$) projection matrices of LoRA into multiple lower-ranked "experts", and introduces a trainable router to weight each expert. We detail our method below:

Base Module. Let $\mathbf{x} \in \mathbb{R}^d$ be the input to a transformer module with a frozen pretrained weight matrix $\mathbf{W} \in \mathbb{R}^{d \times d}$, producing the base representation:

$$\mathbf{h_b} = \mathbf{W}\,\mathbf{x} \quad \in \mathbb{R}^d.$$

Expert Decomposition. While LoRA employs rank-r matrices $\mathbf{A}$ (down-projection) and $\mathbf{B}$ (up-projection), EWoRA partitions them into n experts, each of rank r/n. For expert $i \in \{1, \ldots, n\}$ - $\mathbf{A}_i \in \mathbb{R}^{d \times (\frac{r}{n})}$, $\mathbf{B}_i \in \mathbb{R}^{(\frac{r}{n}) \times d}$. We initialize $\mathbf{A}_i$ with the `kaiming_uniform` distribution (He et al., 2015) while $\mathbf{B}_i$ is zero-initialized. Each expert produces intermediate representations:

$$\mathbf{z}_i = \mathbf{A}_i\mathbf{x} \quad \in \mathbb{R}^{r/n}.$$

We then concatenate all $\mathbf{z_i}$ to form:

$$\mathbf{z} = [\mathbf{z}_1; \ldots; \mathbf{z}_n] \quad \in \mathbb{R}^r.$$

	Code	Math	General		Mixed			
	HumanEval	GSM8K	HellaSwag	Winogrande	HumanEval	GSM8K	HellaSwag	Winogrande
Base	28.74/41.66	39.35/38.97	61.25/81.09	74.27	28.74/41.66	39.35/38.97	61.25/81.09	74.27
LoRA	31.35/46.22	**73.84/73.09**	**66.58/84.23**	71.82	21.09/35.02	37.83/22.44	62.69/82.31	74.35
LoRA+TIES	-	-	-	-	28.59/42.22	37.76/37.60	61.26/81.14	73.88
X-LoRA	-	-	-	-	21.38/26.71	47.31/47.28	62.26/81.86	67.32
AdaLoRA	29.07/43.04	71.04/70.20	62.63/82.34	76.24	28.96/44.60	48.37/47.84	61.09/80.74	75.22
EWoRA	**35.41/49.36**	69.90/69.52	64.64/83.94	**79.24**	**34.61/49.45**	**55.42/55.27**	**62.92/82.60**	**77.58**

Table 1: Accuracy of base (Mistral 7B) and finetuned models using different low-rank adaptation methods.

Routing. A trainable router $\mathbf{W}_r \in \mathbb{R}^{n \times r}$, initialized from a small uniform distribution in the interval $[-10^{-2}, 10^{-2}]$, attends to each expert:

$$\boldsymbol{\alpha} = \mathbf{W}_r \mathbf{z} \quad \in \mathbb{R}^n.$$

Expert Aggregation. $\mathbf{B}_i$ from each expert projects $\mathbf{z_i}$ back to d-dimensions. These representations are combined through a weighted sum:

$$\mathbf{e}_i = \mathbf{B}_i \mathbf{z}_i \quad \in \mathbb{R}^d.$$

$$\mathbf{e} = \sum_{i=1}^{n} \alpha_i \mathbf{e}_i \quad \in \mathbb{R}^d.$$

Final Output. We sum the base and experts outputs to get the final representation ($\mathbf{h}$):

$$\mathbf{h} = \mathbf{h_b} + \mathbf{e} = \mathbf{W}\mathbf{x} + \sum_{i=1}^{n} \alpha_i \mathbf{B}_i(\mathbf{A}_i \mathbf{x}).$$

Parameter Efficiency. EWoRA has a trainable parameter count comparable to LoRA ($2dr$) with only a small additional overhead from the router (nr) which is negligible for large d. We illustrate, let us consider a single linear layer with $d = 4096$, LoRA rank $r = 32$, and number of experts $n = 4$.

1. **Standard LoRA (rank $r = 32$)** adds

$$2 \times d \times r \;=\; 2 \times 4096 \times 32 \;=\; 262{,}144$$

2. **EWoRA (rank $r = 32$, with $n = 4$)** adds the same 262,144 plus a router overhead of

$$n \times r \;=\; 4 \times 32 \;=\; 128$$

Thus, EWoRA adds $262{,}144 + 128 = 262{,}272$ parameters in total for this layer.

The difference of 128 parameters is negligible relative to $d \times d = 4096^2 = 16{,}777{,}216$ parameters in the frozen weight $\mathbf{W}$ or 262,144 parameters in the LoRA projection matrices.

Concretely, when finetuning on the Mistral 7B base model with adapters on all target modules [Q_PROJ, K_PROJ, V_PROJ, O_PROJ, GATE_PROJ, UP_PROJ, DOWN_PROJ], EWoRA has a total of 83915648 trainable parameters compared to LoRA 83886080 for $n = 4$ and $r = 32$. This is a difference of 29568 additional parameters representing an increase of $\approx 0.03\%$ trainable parameters and $\approx 0.0004\%$ overall parameters due to the EWoRA routing matrices.

4 Experiments & Results

Setup. Following recent studies (Biderman et al., 2024; Zhang et al., 2024; Gu et al., 2025), we evaluate EWoRA on three representative tasks: code generation, mathematical reasoning, and general reasoning. For code generation, we train on Magicoder-Evol-Instruct-110K (Wei et al., 2023) and evaluate on HumanEval (Chen et al., 2021). For math, we train on MetaMathQA (Yu et al., 2024) and evaluate on GSM8K (Cobbe et al., 2021). For general reasoning, we use HellaSwag (Zellers et al., 2019) and Winogrande (Sakaguchi et al., 2020), training and evaluating on their combined trains and test splits respectively. To assess multi-domain adaptation, we create a **mixed-task** setup by training on all three datasets together and evaluating across their respective test sets. Mixing is performed using simple concatenation (with shuffling at train time). While we acknowledge that careful data curation methods such as weighted sampling might improve performance further, we hope to replicate data in real-world applications which can be noisy and require multiple, diverse skillsets (their distribution or even presence may not be known apriori). Mistral 7B (Jiang et al., 2023) is used as the base model in our experiments, with rank $r = 32$ for all methods and $n = 4$ experts in EWoRA. For the model merging and MoErging baselines, we use TIES and X-LoRA respectively to combine outputs from single-task LoRA adapters. We report multiple metrics: Pass@1/Pass@10 for code, strict/flexible match for math, and standard/normalized accuracy for Hel-

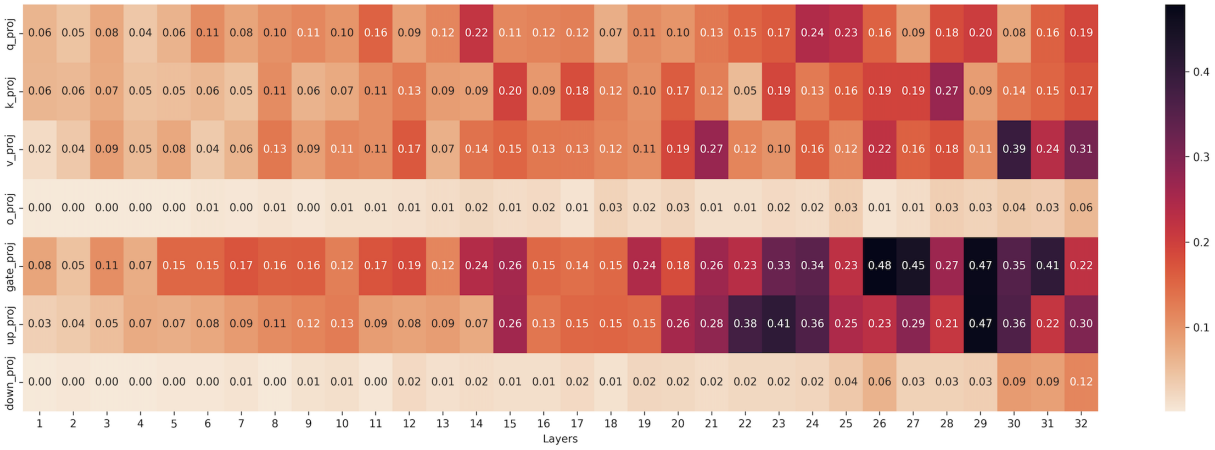

Figure 2: Average Pairwise L1 Distance between mean expert weight vector for Code, Math, and General. We demonstrate the pairwise L1 distance $\text{Dist}^{(\ell,m)}$ for averaged expert weight $\boldsymbol{\alpha}^{(\ell,m)}_{(\text{task\_type})}$ per layer and per LLM modules in the heatmap. The darker the color the more distinct the expert routing is for different tasks.

laSwag. Further details are provided in (§B).

Results. Table 1 summarizes our findings. In **single-task** settings, EWoRA is the best performing variant on code and Winogrande, whereas LoRA does better on math and HellaSwag. EWoRA also surpasses AdaLoRA on all tasks except math. In the **mixed-task** setup, EWoRA consistently beats all baselines across all test splits, demonstrating its effectiveness in heterogeneous finetuning.

5 Analysis

To understand how EWoRA routes between experts for different tasks, we analyze the weights assigned to each expert by the routers (the $\boldsymbol{\alpha}$ vectors in Section 3). In our setup, there are $n = 4$ experts, so the router produces a 4-dimensional vector $\boldsymbol{\alpha}^{(\ell,m)}_{(\text{task\_type})}$ at layer ℓ and module m, for a given input. Here, m spans all the target modules of Mistral 7B. We sample over inputs of each task type and compute the expert weight average per task type:

$$\overline{\boldsymbol{\alpha}}^{(\ell,m)}_{\text{task\_type}} = \frac{1}{N_{\text{task\_type}}} \sum_{i=1}^{N_{\text{task\_type}}} \boldsymbol{\alpha}^{(\ell,m)}_{(\text{task\_type}),i}$$

We then measure how distinctly each adapter routes between task types. We compute the average pairwise distance between the three averaged expert weight vectors $\overline{\boldsymbol{\alpha}}^{(\ell,m)}_{\text{code}}$, $\overline{\boldsymbol{\alpha}}^{(\ell,m)}_{\text{math}}$, $\overline{\boldsymbol{\alpha}}^{(\ell,m)}_{\text{general}}$. Given the L1 distance ($d(\cdot,\cdot)$), we define:

$$\text{Dist}^{(\ell,m)} = \frac{d(\overline{\boldsymbol{\alpha}}_{\text{code}}, \overline{\boldsymbol{\alpha}}_{\text{math}}) + d(\overline{\boldsymbol{\alpha}}_{\text{math}}, \overline{\boldsymbol{\alpha}}_{\text{gen}}) + d(\overline{\boldsymbol{\alpha}}_{\text{gen}}, \overline{\boldsymbol{\alpha}}_{\text{code}})}{3}$$

Higher $\text{Dist}^{(\ell,m)}$ indicates greater divergence of expert weights across domains. Figure 2 visualizes these distances across layers (columns 1–32) and target modules (rows), revealing two key trends:

1. **Layer-wise Specialization:** Early layers exhibit lower L1 distances, suggesting that domain specialization occurs primarily in later layers. This aligns with prior observations that lower layers capture general representations, while deeper layers encode task-specific information (Zhang et al., 2023b).

2. **Module-Specific Specialization:** Feed-forward projection modules, particularly GATE_PROJ and UP_PROJ, show the highest L1 distances. This suggests that domain-specific knowledge is concentrated in these components, whereas attention projections (Q_PROJ, K_PROJ, V_PROJ, O_PROJ) exhibit more uniform expert weightings across tasks.

Both these findings align with prior work on adaptive rank allocation (Zhang et al., 2023b). We extend this analysis into tasks within general reasoning, comparing routing between HellaSwag and Winogrande. LoRA fine-tuning degrades performance on Winogrande relative to the base model, while EWoRA improves it. We hypothesis the degradation from LoRA is likely due to data imbalance. In significant imbalanced mixed training setups, large datasets (e.g., HellaSwag) can dominate smaller ones (e.g., Winogrande), leading to LoRA overfitting to the majority domain. This results in stronger performance on the larger dataset but weaker generalization to the minority domain. In contrast, EWoRA can dynamically allocate its low-rank adapters to underrepresented tasks. Appendix

C further analyzes the pairwise routing distances between HellaSwag and Winogrande, showing distinct expert weighting for different input types.

Recent studies suggest LoRA-finetuned models may outperform full fine-tuning on source domains not seen during finetuning (Biderman et al., 2024). In (§D), we carry out similar experiments to assess "forgetting" when finetuning with EWoRA relative to other baselines. EWoRA demonstrates a slight advantage due to improved retention of unseen tasks, leading to a better learning-forgetting tradeoff even in the single-task setting.

6 Conclusion & Future Work

We introduce EWoRA, an extension of LoRA for fine-tuning LLMs on heterogeneous data. By partitioning low-rank adapters into experts and dynamically weighting them via a lightweight router, EWoRA enables more effective adaptation to diverse target domains while maintaining parameter efficiency. Our experiments show that EWoRA outperforms baselines in mixed-task setups, while remaining competitive in single-task settings.

Analysis of expert routing patterns reveals that domain specialization primarily occurs in later layers and feed-forward projection modules, suggesting opportunities for further optimization. Future work could explore integrating adaptive rank allocation or expert pruning to further improve performance. Adaptive allocation could be extended to individual experts, potentially assigning asymmetric ranks to each expert within a target module. This strategy might benefit cases where one target domain is significantly more complex than another in a mixed setup (e.g. code and general reasoning).

Limitations

EWoRA enables improved finetuning of LLMs on heterogeneous data and is evaluated across three domains (general reasoning, code generation, and arithmetic reasoning) using a modern LLM. This mix helps us assess our methods across a spectrum of complexity levels - tasks that are difficult for the base model and remain difficult for the fine-tuned model (coding), tasks that are difficult for the base model but are relatively easily learnt during fine-tuning (math), and tasks that the base model is already competent in (commonsense reasoning). However, due to computational constraints, we do not explore additional model families (e.g., LLaMA (Dubey et al., 2024)), larger models (13B, 30B, 70B), or alternative training paradigms (e.g., continual pretraining). Our current setup is in line with several recent studies that have demonstrated results on the same mix of tasks and datasets (Biderman et al., 2024; Zhang et al., 2024; Gu et al., 2025) using a single, modern LLM. We acknowledge that exploring models of varying sizes, as well as other training setups such as continual pre-training would offer interesting insights. We leave this to future work and also invite the broader community to investigate this further.

Unlike standard LoRA, which allows adapters to be merged back into the base model post-training — eliminating inference overhead — EWoRA's expert-weighting is input-dependent and must run at inference time. This prevents direct merging and introduces a small increase in inference cost. While running adapters in parallel is advantageous for multi-adapter deployments (avoiding multiple large-base instances), merging remains more efficient when only a single adapter is needed.

Finally, LoRA is quite sensitive to hyperparameters such as learning rate (Biderman et al., 2024). Our experiments follow a fixed hyperparameter selection process: we conduct pilot runs on code and math datasets using LoRA to tune learning rate, rank, and epochs, and then apply the best settings across AdaLoRA and EWoRA. Consequently, we do not investigate EWoRA's sensitivity to hyperparameters in this work.

References

Loubna Ben Allal, Niklas Muennighoff, Logesh Kumar Umapathi, Ben Lipkin, and Leandro von Werra. 2022. A framework for the evaluation of code generation models. https://github.com/bigcode-project/bigcode-evaluation-harness.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John Patrick Cunningham. 2024. LoRA learns less and forgets less. *Transactions on Machine Learning Research*. Featured Certification.

Eric L Buehler and Markus J Buehler. 2024. X-lora: Mixture of low-rank adapter experts, a flexible framework for large language models with applications in protein mechanics and molecular design. *APL Machine Learning*, 2(2).

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph,

Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code.

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

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*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. A framework for few-shot language model evaluation.

Naibin Gu, Zhenyu Zhang, Xiyu Liu, Peng Fu, Zheng Lin, Shuohuan Wang, Yu Sun, Hua Wu, Weiping Wang, and Haifeng Wang. 2025. Beamlora: Beam-constraint low-rank adaptation. *arXiv preprint arXiv:2502.13604*.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. LoRA+: Efficient low rank adaptation of large models. In *Forty-first International Conference on Machine Learning*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034.

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.

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. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Ting Jiang, Shaohan Huang, Shengyue Luo, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, Qi Zhang, Deqing Wang, and Fuzhen Zhuang. 2024. Mora: High-rank updating for parameter-efficient fine-tuning. *Preprint*, arXiv:2405.12130.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M Asano. 2024. VeRA: Vector-based random matrix adaptation. In *The Twelfth International Conference on Learning Representations*.

Chu-Cheng Lin, Xinyi Wang, Jonathan H Clark, Han Lu, Yun Zhu, Chenxi Whitehouse, and Hongkun Yu. 2024. Inducing generalization across languages and tasks using featurized low-rank mixtures. *arXiv preprint arXiv:2402.17934*.

Tongxu Luo, Jiahe Lei, Fangyu Lei, Weihao Liu, Shizhu He, Jun Zhao, and Kang Liu. 2024. Moelora: Contrastive learning guided mixture of experts on parameter-efficient fine-tuning for large language models. *arXiv preprint arXiv:2402.12851*.

Yuren Mao, Yuhang Ge, Yijiang Fan, Wenyi Xu, Yu Mi, Zhonghao Hu, and Yunjun Gao. 2024. A survey on lora of large language models. *Frontiers of Computer Science*, 19(7).

Pouya Pezeshkpour and Estevam Hruschka. 2025. Learning beyond the surface: How far can continual pre-training with lora enhance llms' domain-specific insight learning? *Preprint*, arXiv:2501.17840.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2023. DyLoRA: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3274–3287, Dubrovnik, Croatia. Association for Computational Linguistics.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. 2023. Magicoder: Source code is all you need. *arXiv preprint arXiv:2312.02120*.

Xun Wu, Shaohan Huang, and Furu Wei. 2024. Mixture of loRA experts. In *The Twelfth International Conference on Learning Representations*.

Prateek Yadav, Colin Raffel, Mohammed Muqeeth, Lucas Caccia, Haokun Liu, Tianlong Chen, Mohit Bansal, Leshem Choshen, and Alessandro Sordoni. 2024. A survey on model moerging: Recycling and routing among specialized experts for collaborative learning. *CoRR*, abs/2408.07057.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. Ties-merging: Resolving interference when merging models.

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. In *Forty-first International Conference on Machine Learning*.

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. In *Forty-first International Conference on Machine Learning*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng YU, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations*.

Ted Zadouri, Ahmet Üstün, Arash Ahmadian, Beyza Ermis, Acyr Locatelli, and Sara Hooker. 2024. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning. In *The Twelfth International Conference on Learning Representations*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.

Jingfan Zhang, Yi Zhao, Dan Chen, Xing Tian, Huanran Zheng, and Wei Zhu. 2024. Milora: Efficient mixture of low-rank adaptation for large language models fine-tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 17071–17084.

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

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

Jiacheng Zhu, Kristjan Greenewald, Kimia Nadjahi, Haitz Sáez de Ocáriz Borde, Rickard Brüel Gabrielsson, Leshem Choshen, Marzyeh Ghassemi, Mikhail Yurochkin, and Justin Solomon. Asymmetry in low-rank adapters of foundation models. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*.

Bojia Zi, Xianbiao Qi, Lingzhi Wang, Jianan Wang, Kam-Fai Wong, and Lei Zhang. 2023. Delta-lora: Fine-tuning high-rank parameters with the delta of low-rank matrices. *arXiv preprint arXiv:2309.02411*.

A Comparison with MoErging Approaches

MoLE/X-LoRA: Both of these methods take pre-trained adapters on distinct tasks and then finetune a mechanism to route between them. However, this method assumes segregated data for the distinct tasks on which the adapters were initially trained. Our method is more applicable to real-world data which may be noisy and require multiple, diverse skillsets (their distribution or even presence may not be known *a priori*). We utilize the intermediate representation as input to the router adding minimal overhead in terms of the number of parameters (described in (§3)). This is not true for MoLE/X-LORA as they take full-rank representations as input, resulting in a much larger overhead. Moreover, MoLE/X-LoRA require a separate finetuning set to train the routing mechanism which, in our case, will be a mix of the same individual training sets. EWoRA is able to implicitly learn from a mix of datasets and we demonstrate our method on a simple concatenation of tasks of varying difficulty as well as dataset size. EWoRA shows better generalization to tasks underrepresented in the final mix of our training data (described in Section 5). Our method has potential for cross-task learning as all the matrices or experts are exposed to the entire mix of training data as opposed to methods like MoLE/X-LoRA where each expert is trained on one task split in isolation.

FLix: In the featurized low-rank mixtures approach (Lin et al., 2024), separate adapters are learnt for each language-task pair in a mixture. While the adapters are learnt simultaneously, this method also relies on segregation of the train data as inputs are routed based on $f_i(x) = 1$ where f_i is a sparse vector and $f_i(x) = 1$ indicates that input x has feature i. In this setting, for an input to a language-task pair, the language as well as task is known *a priori* and there is some learning across tasks in the same language as well as across languages involving the same tasks. However, EWoRA implicitly learns this routing behavior without distinguishing between inputs. FLix

assumes knowledge of language and task for a new test input whereas EWoRA does not and can route amongst experts dynamically based on a new input. Moreover, the soft weighting mechanism potentially allows for better cross-task learning such as for code and math where skills might be complementary than a hard routing similar to FLix.

MoLORA: This method (Zadouri et al., 2024) is perhaps the most comparable to EWoRA in terms of architectural similarity as well as training methodology - simultaneous training of the router and adapter matrices, soft routing, and no label supervision or segregation of data at train or inference time. However, routing occurs with full-rank inputs which makes each individual routing operation considerably more expensive whereas EWoRA uses intermediate low-rank representations for the routing operation. This lightweight operation is what allows EWoRA to route amongst experts at an individual matrix or adapter level as opposed to MoLoRA which routes at a token level (though ablations at a sentence-level have also been carried out). This allows more flexible routing and the later layers are able to learn task-level features with greater distinction and more aggressive routing between experts. This is highlighted in the analysis in Sections 5 and (§C), and the related heatmaps (Figures 2 & 3).

MoELoRA: While MoELoRA (Luo et al., 2024) performs a similar routing between experts, the routing decision is deferred to the intermediate representations in EWoRA leading to significantly lower parameter utilization relative to MoELoRA for the same rank R. We employ a soft-routing mechanism in EWoRA vs hard-routing in MoELoRA. Instead of a sparse-gating mechanism in MoELoRA, we use router weights to attend to each expert allowing a flexible composition of any number of experts depending on the task. This allows for better cross-task generalization where multiple experts can be combined for contexts where multiple skills might be required simultaneously. The soft-weighting mechanism also helps overcome the load-imbalance problem inherent to MoE models that is also mentioned in MoELoRA. Unlike MoELoRA, we do not introduce auxiliary losses in order to introduce load-balancing or explicitly enforce expert specialization. However, as we see in our analysis in Section 5 - our method implicitly achieves expert specialization and there is more distinct routing in later layers and in the projection matrices. Finally, we also feel that the motivation for sparse Mixture of Experts models, which is increasing model capacity without increasing the number of active parameters at inference time, does not translate well in this case as the number of active parameters is dominated by the base model. On the other hand, our approach is able to flexibly utilize all parameters through the soft-weighting mechanism.

B Additional Experimental Details

For our evaluation, we use the `lm-evaluation-harness` (Gao et al., 2024) for math and general tasks, and `bigcode-evaluation-harness` (Ben Allal et al., 2022) for code. For the X-LoRA baseline, we choose dense-routing with a single 512-dimensional hidden layer. The following hyperparameters are used in all our training experiments:

- **Sequence length**: 1024
- **Rank** (r): 32
- **LoRA Scaling Parameter** (α): 32
- **Num Training Epochs**): 4
- **Num EWoRA Experts** (n): 4
- **Optimizer**: AdamW
- **Learning rate**: 5e-5 for Code, 1e-4 for Math/General/Mixed
- **LR Scheduler**: Cosine Annealing
- **Min_lr rate**: 0.1
- **Warmup ratio**: 0.1
- **Precision**: bf16
- **Warmup ratio**: 0.1
- **Weight Decay**: 0

For all our evaluations using the lm-evaluation harness (gsm8k, hellaswag, winogrande) we use the default parameters:

- **Temperature**: 0
- **Metric**: pass@1
- **Few-shot**: 5

- **Precision**: bf16

For code evaluation on the HumanEval benchmark using the bigcode-evaluation-harness, we use the following parameters:

- **Temperature**: 0.2
- **Metric**: pass@1/pass@10
- **Few-shot**: 0
- **Top_p**: 0.95
- **N_samples**: 50
- **Precision**: bf16

In all our results, we report both the flexible and strict-match scores as reported by the lm-evaluation-harness for math (gsm8k), and standard and normalized accuracy for both hellaswag and ARC. Similarly, for code (HumanEval), we report the pass@1 and pass@10 results as reported by the bigcode-evaluation-harness.

C Routing in the General Model

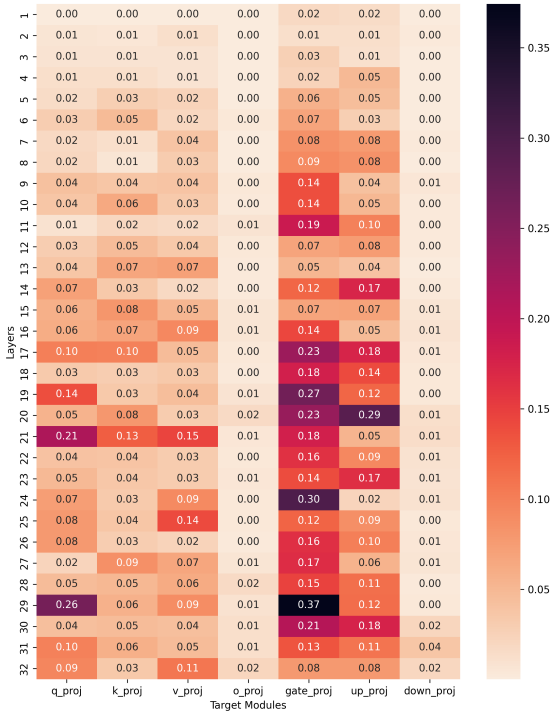

Figure 3: Routing in the general reasoning model.

Similar to the mixed model analysis in Section 5, we analyze the routing behavior of EWoRA for different task types (Hellaswag, Winogrande) in the general reasoning model. Since we have two different task types, we take the L1 difference $d(\overline{\boldsymbol{\alpha}}^{(\ell,m)}_{\text{hellaswag}} - \overline{\boldsymbol{\alpha}}^{(\ell,m)}_{\text{winogrande}})$. As seen in Figure 3, the general model shows a similar routing behavior to the mixed model which helps EWoRA better adapt to the under-represented set (Winogrande).

D Evaluating on the Source Domain

Similar to Biderman et al. (2024), we use the models finetuned on the code and math target domains and evaluate on HellaSwag, Winogrande, and the ARC-Challenge (Clark et al., 2018) benchmarks. The evaluation tasks are termed "source-domain" tasks as the base pretrained model already exhibits good competency on them as evidenced by its performance metrics. Thus, they provide a good benchmark to evaluate potential degradation in model performance when finetuned to a specialized target domain (such as code or math).

Results, as shown in Table 2, indicate that EWoRA is competitive with baselines - LoRA and AdaLoRA - when evaluated on the forgetting tasks (performing better than both in 4 out of 6 cases). This suggests that EWoRA retains at least as much source-domain knowledge as LoRA, leading to significantly less forgetting than full-finetuning as shown by Biderman et al. (2024). Given the generally-better performance on the target-domain in our experiments, we can say that EWoRA also leads to a better learning-forgetting tradeoff than conventional LoRA.

	HellaSwag	Winogrande	ARC
Mistral 7B (Base)	61.25/81.09	74.27	50.0/54.10
Code-Finetuned			
LoRA	61.22/80.90	**75.06**	**50.26/54.01**
AdaLoRA	61.21/80.96	74.03	50.0/53.75
EWoRA	**61.84/81.12**	73.09	48.04/52.05
Math-Finetuned			
LoRA	58.37/74.06	60.69	41.38/44.62
AdaLoRA	61.19/78.14	68.43	41.64/45.82
EWoRA	**61.64/79.72**	**72.22**	**45.31/49.83**

Table 2: Accuracy on source-domain tasks.