Seeded LoRA: Collaborative Fine-Tuning Through Seed Initialization of Adapters

Alejandro R. Salamanca [1] **Ahmet Üstün** [2] **Nicki Skafte Detlefsen** [1] **Tim Dettmers** [3]

Abstract

Parameter-Efficient Fine-Tuning (PEFT) methods enable cost-effective adaptation of pretrained language models to specific tasks and domains. Collaborative Fine-Tuning (CoFT) seeks to merge these specialized models into a single model, often a routed Mixture-of-Expert (MoE) model, to achieve better generalization across domains and tasks. However, current CoFT models require a post-merge fine-tuning stage, making these approaches inaccessible to users lacking fine-tuning expertise. We introduce Seeded LoRA, a novel CoFT approach that does not require post-merge fine-tuning, enabling plug-and-play PEFT adapter merging. Seeded LoRA outperforms LoRA and MoE LoRA (MoLoRA) approaches by an average of 7 percentage points across 16 zero-shot tasks. Seeded LoRA works by initializing a model with a generic seed expert low-rank adapter, ensuring subsequent fine-tuning runs are in the same optimization space, exhibiting linear mode connectivity. This process allows integrating independently fine-tuned models into a single model using a static, untrained soft uniform probability router. We show that this formulation is equivalent to grouped convolution or multi-head processing, explaining its effectiveness. Additionally, we highlight that Seeded LoRA alleviates most routing failures in post-merge fine-tuning, making it a suitable base method for future routed CoFT approaches.

1. Introduction

Fine-tuning pretrained Large Language Models (LLMs) to follow user instructions (Wei et al., 2022) is crucial for developing interactive chatbots. Parameter-Efficient Fine-Tuning (PEFT) (Hu et al., 2021; Liu et al., 2022; Li & Liang, 2021; Zadouri et al., 2023) methods like Low-Rank Adaptation (LoRA) (Hu et al., 2021) enable the creation of numerous domain-specific models (Wolf et al., 2019; Mangrulkar et al., 2022). However, enhancing a model with a new capability, such as code generation, traditionally requires re-training with new data mixes, incurring high computational costs and requiring domain-specific expertise.

Collaborative fine-tuning (CoFT) aims to extend a model's capabilities by merging it with other fine-tuned models, reusing the expertise and computational resources invested in existing models. Current CoFT strategies often necessitate post-merge fine-tuning to enable the successful use of existing PEFT models (Muqeeth et al., 2024; Dou et al., 2024; Zadouri et al., 2023).

In this paper, we introduce **Seeded LoRA**, a CoFT approach that does not require post-merge fine-tuning. Seeded LoRA works by initializing a model with a generic seed expert low-rank adapter before fine-tuning, ensuring subsequent fine-tuning runs are in the same optimization space and exhibit linear mode connectivity (Frankle et al., 2020). This enables merging existing models by averaging (Li et al., 2022; Wortsman et al., 2022; Ilharco et al., 2022). With a common initialization, no post-merge fine-tuning of a router is required; instead, we average the outputs of all PEFT experts, equivalent to a soft mixture of experts with a static router assigning equal probability to each expert. We show that this formulation is equivalent to grouped convolutions (Xie et al., 2017).

Our findings demonstrate that Seeded LoRA provides state-of-the-art CoFT performance, on par with more complex routed models, and establishes its equivalence to grouped convolutions and multi-head processing, explaining the effectiveness of Seeded LoRA and other MoE approaches like Mixtral (Jiang et al., 2024).

We evaluated Seeded LoRA on 16 zero-shot tasks and found that it improves performance significantly, increasing average accuracy from 44.1% (LoRA) and 45.6% (MoLoRA) to 52.5%. Furthermore, our analysis shows that most CoFT routing techniques have subtle failure modes leading to poor

[1]Technical University of Denmark [2]Cohere For AI [3]University of Washington. Correspondence to: Alejandro R. Salamanca <alejandro.rsalamanca@gmail.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

performance. Seeded LoRA initialization can overcome these failures, and uniform routing (averaging) yields performance comparable to routed models. Successful router training is no better than averaging; seeding is more crucial than router training.

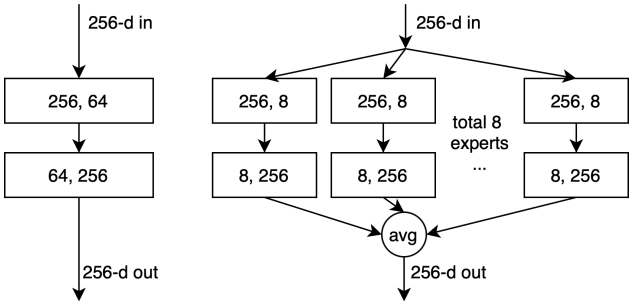

Figure 1. **Left:** LoRA (Hu et al., 2021) adapter with a rank of 64. **Right:** Seeded LoRA with 8 experts, each having a rank of 8. Both adapters have the same parameter count.

2. Related Work

Low-Rank Adaptation (LoRA) freezes the pretrained parameters of a model and adds only a small set of trainable parameters called low-rank adapters (Hu et al., 2021). This decreases the memory required for fine-tuning by a factor of roughly 6x through reduced memory requirements for gradients and optimizers states. Given a pretrained weight matrix $\mathbf{W}_0 \in \mathbb{R}^{h\times o}$ and intermediate token activation $\mathbf{x} \in \mathbb{R}^h$, LoRA adds a low-rank projection to the outputs of the layer as follows:

$$\mathbf{y} = \mathbf{x}\mathbf{W}_0 + \mathbf{x}\mathbf{A}\mathbf{B} \tag{1}$$

where $A \in \mathbb{R}^{h\times r}$ and $B \in \mathbb{R}^{r\times o}$. Only the weights $\mathbf{A}$, and $\mathbf{B}$ are updated during finetuning.

Collaborative Fine-Tuning (CoFT) Traditional fine-tuning of a LLM often results in a model with static capabilities, where introducing new functionalities, such as mathematical problem-solving, might erase previously learned skills due to catastrophic forgetting. Typically, enhancing a model's capabilities involves retraining it from scratch with a comprehensive dataset encompassing both old and new domains, a process that is not only computationally intensive but also demands access to all previous data and domain-specific fine-tuning expertise. CoFT addresses these limitations and extends the capabilities of existing model without necessitating retraining. For instance, incorporating math skills into a model could be achieved by *merging* it with another model specifically fine-tuned for mathematics. Among various integration methods, the most common involves deploying a router to manage the interaction between these specialized models (Muqeeth et al., 2024).

CoFT vs Federated Learning In federated learning (Lim et al., 2020), individual models are locally trained and then merged on a central server, preserving user privacy. CoFT, while similar in merging parameters, focuses on final model performance and simplicity, differing in usability and model scale. In CoFT, privacy is not a major consideration while final performance of the merged model is critical; CoFT allows only for a single exchange of model parameters and not successive updates; large models, that cannot be executed on edge devices and introduce new challenges that do not exist at the small scale, are used (Dettmers et al., 2022). Federated learning approaches are usually not suitable CoFT solutions and vice versa due to usability, model scale, single step merging, and importance of final model performance.

Mixture-of-Adapters (MoA) methods such as MoLoRA, SIRA, and LoRAMoE (Zadouri et al., 2023; Zhu et al., 2023; Dou et al., 2024) learn a set of experts $E_1, ..., E_n$ where each expert E_i is a LoRA adapter, combined via a router network R that generates gating scores $s_1, ..., s_n$ for a weighted sum of expert outputs (soft mixture) (Masoudnia & Ebrahimpour, 2014) or the experts with the top-k probability (sparse mixture) (Lepikhin et al., 2020; Shazeer et al., 2017)

$$s_i = R(x)_i = \text{softmax}(W_R^T x) \quad \text{(Router)}$$
$$y = \sum_{i=1}^{n} s_i \cdot E_i(x) \quad \text{(MoA Layer)}$$

Branch-Train-Merge (BTM) Approaches BTM (Li et al., 2022) is a communication-efficient algorithm designed for the parallel training of large language models (LLMs). It facilitates the independent training of model subparts, called Expert Language Models (ELMs), across different data subsets. ELMs form the ELMFOREST and can be dynamically modulated or integrated through ensembling or parameter averaging. Cluster-Branch-Train-Merge (c-BTM) (Gururangan et al., 2023) extends BTM by incorporating unsupervised domain discovery, enabling domain-specific training and forming a sparse ensemble for efficient inference. Branch-Train-MiX (Sukhbaatar et al., 2024) further advances this paradigm by mixing trained domain-specific experts into an MoE model, yielding an efficient LLM with enhanced accuracy-efficiency trade-offs.

Optimization Landscape Initialization via *Seed* Experts. The initialization of fine-tuning from a shared *seed* expert

ensures that the subsequent domain or task specific finetuning occurs in a common optimization space. When neural networks are being trained from a random parameter initialization they quickly settle into a optimization subspace characterized by the principle eigenvalues of the Hessian (Ghorbani et al., 2019; Gur-Ari et al., 2018; Frankle et al., 2019). Once this subspace is entered the principal optimization directions remain fixed for the rest of the training (Frankle et al., 2019; Ghorbani et al., 2019; Gur-Ari et al., 2018) and exhibit linear mode connectivity(Frankle et al., 2020). This means, two neural networks with different random initialization may have different optimization subspaces, but if two neural networks are trained from an initialization that has settled into an optimization subspace the two neural networks will remain in that subspace even if trained on different data (Frankle et al., 2020). If two neural networks exhibit linear mode connectivity, they can be merged by a simple or weighted average (Li et al., 2022; Gururangan et al., 2023; Wortsman et al., 2022; Ilharco et al., 2022). We exploit this property to enable CoFT that does not require any post-merge finetuning.

3. Methodology

This section details the Seeded LoRA approach and its equivalence to grouped convolutions. Additionally, it outlines the experimental setup used to evaluate the proposed method.

3.1. Seeded LoRA

Seeded LoRA builds upon the foundation laid by LoRA and MoLoRA and introduces key improvements over other Mixture-of-Experts (MoE) methods. Unlike the complex dynamic routing mechanisms of these methods, Seeded LoRA uses a static uniform router that does not require post-merge finetuning. The router assigns equal weight to each expert's contribution to the final output. This simplicity avoids the need for additional training and is formalized in the following equation:

$$\mathbf{y} = \mathbf{x}\mathbf{W}_0 + \underbrace{\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}\mathbf{A}_i\mathbf{B}_i}_{\text{Seeded LoRA update}} \tag{2}$$

Here, $\mathbf{W}_0$ represents the base model parameters, $\mathbf{x}$ is the input, and $\mathbf{B}_i$ and $\mathbf{A}_i$ are the i_{th} LoRA adapter.

During inference, Seeded LoRA can merge the adapters into the weights, reducing inference overhead significantly:

$$y = \mathbf{x}\left(\mathbf{W}_0 + \frac{1}{N}\sum_{i=1}^{N}\mathbf{A}_i\mathbf{B}_i\right) \tag{3}$$

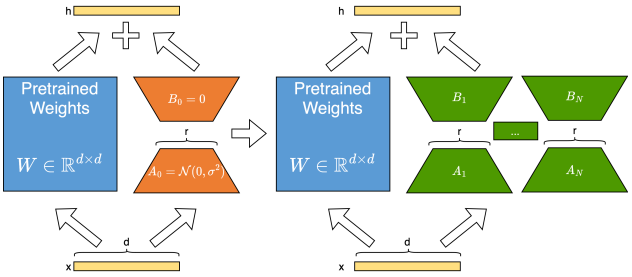

Figure 2. **Left:** *Seed Expert* training. This expert is a regular LoRA adapter. **Right:** Seeded LoRA is a combination of LoRA adapters. Each adapter, trained using the *Seed Adapter* as starting point, acts as an Expert in this MoA model. Inputs are sent to every expert, and the outputs are averaged and added to the pretrained model output.

3.2. Equivalence to Grouped Convolutions

This section establishes the direct equivalence between Seeded LoRA's approach and grouped convolutions (Xie et al., 2017) (refer to Appendix B), under specific initialization strategies. This equivalence helps to explain the effectiveness of Seeded LoRA, as grouped convolutions often yield improvements over regular convolutions (Xie et al., 2017; Zhang et al., 2018) and exhibit distinct features across groups (Krizhevsky et al., 2012b).

LoRA as 1x1 Convolution At its core, a LoRA layer functions as a sequence of 1x1 convolutions that first minimize the feature dimension before restoring it:

$$\text{LoRA A Layer:} \quad h \xrightarrow{1\times1,r} r \tag{4}$$

$$\text{LoRA B Layer:} \quad r \xrightarrow{1\times1,h} h \tag{5}$$

where r is the reduced dimensionality for the adaptation, and h is the input and output dimension. In Seeded LoRA, numerous LoRA adapters are merged through Uniform Routing.

Grouped Convolutions Grouped Convolutions process input channels in separate groups, each undergoing independent convolutions. This concept mirrors Seeded LoRA, employing sequences of 1x1 convolutions for channel reduction and subsequent projection back to the original size. The outputs are then summed across groups.

Initialization for Equivalence The main difference between grouped convolutions and Seeded LoRA is that grouped convolutions aggregate through a sum while Seeded LoRA uses the average. For equivalence to grouped convolutions, we can initialize Seeded LoRA accordingly. Given that LoRA's output variance is linked to the input channel count (h), initializing LoRA adapters with a variance

of $\frac{1}{h}$ leads to standard normal $N(0, 1)$ outputs (Glorot & Bengio, 2010). Correspondingly, grouped convolutions, which distribute variance $N(0, h)$ per group, sum this across all groups. Aligning Seeded LoRA's variance necessitates equating the number of LoRA adapters (e) with grouped convolution's groups (k), achieved by initializing Seeded LoRA's adapters with $N(0, \sqrt{1/(h \times e)})$. This setup ensures the output variances from Seeded LoRA and grouped convolutions are identical, demonstrating that both formulations are the same.

Additionally, multi-headed methods, akin to grouped convolutions, partition the input into multiple *heads*, processing each with independent weight matrices. Seeded LoRA's approach, where each LoRA adapter acts as a *head*, closely aligns with this concept, enabling parallel processing of inputs with specialized adapters.

3.3. Experimental Setup

We evaluated Seeded LoRA using the Llama 2 7B model (Touvron et al., 2023), comparing it to LoRA and MoLoRA with equivalent parameter budgets on various zero-shot tasks. The experiments leverage a range of datasets for instruction fine-tuning (Appendix C) containing a mix of general knowledge, code, and mathematics, totaling 282,360 data points[1]. A single *seed expert* model is trained on a random 10% subset of the fine-tuning dataset, ensuring all subsequent experts share a common optimization space. The exact data is not important. Li et al. trained a seed model on Javascript, and experimented local mode connectivity when training across code-unrelated domains such as medical, law, Arxiv, or C4.

We assessed performance on a range of tasks using the *lm-evaluation-harness* (Gao et al., 2023) covering diverse areas like natural language inference, arithmetic reasoning, commonsense reasoning, and question answering.

We use a multi-stage fine-tuning process to simulate the existence of independent open-source LoRA models, starting with a pretrained LLM $\mathcal{M}$, trained on a variety of topics. We aim to improve $\mathcal{M}$'s performance in N specific areas of expertise. To achieve this, we fine-tune $\mathcal{M}$ with N corresponding datasets, $\mathcal{D} := \{D_1, \ldots, D_N\}$, where each dataset is related to a specific domain. For each fine-tuning run we follow the following steps:

1. *Seed Expert* Training A single *seed expert* model is trained on a random subset (e.g., 10%) of the finetuning dataset $\mathcal{D}$. This ensures all subsequent experts share a common optimization space that exhibits linear mode connectivity.

[1]https://huggingface.co/datasets/alexrs/seededlora-data

2. *Dataset-Specific Expert* Training Each expert is then fine-tuned independently on its data. This simulates independent training of open-source models if initialized with a seed expert.

3. Combining Experts Finally, Seeded LoRA incorporates all fine-tuned adapters using soft uniform routing.

3.4. Baseline Methods

The baseline methods used for comparison include LoRA and MoLoRA. LoRA freezes the pretrained parameters of a model and adds only a small set of trainable parameters called low-rank adapters. MoLoRA extends this approach by combining multiple LoRA adapters using a router network that generates gating scores for a weighted sum of expert outputs. For MoLoRA, we use the same number of experts as Seeded LoRA, following the method described in Zadouri et al.. All methods present an equivalent parameter budget.

4. Results

Table 1 summarizes the zero-shot accuracy achieved by each model. Seeded LoRA outperforms LoRA and MoLoRA on average. This suggests Seeded LoRA's ability to effectively leverage expert knowledge for broader task applicability. Notably, Seeded LoRA exhibits superior performance in tasks like arithmetic reasoning (2D and 4D), where LoRA and MoLoRA struggle. Compared to LoRA and MoLoRA, Seeded LoRA consistently achieves better results across various tasks. Despite some tasks where other models scored marginally higher, Seeded LoRA's overall performance shows effectiveness as a fine-tuning approach, especially in a multi-task setting. This highlights Seeded LoRA's ability to maintain or improve benchmark performance while introducing chatbot abilities, a common challenge when fine-tuning LLMs (Luo et al., 2023; Dou et al., 2024).

Appendix F contains results for each individual expert in Seeded LoRA.

5. Conclusions

In this paper, we introduced Seeded LoRA, a novel PEFT method that leverages LoRA adapters to enable collaborative fine-tuning. Seeded LoRA distinguishes itself by enabling the integration of expert knowledge from independently trained adapters, without requiring additional computational resources for post-merge fine-tuning by leveraging a seed adapter as the starting point for fine-tuning.

Our analysis shows that Seeded LoRA outperforms existing PEFT methods in a set of 16 zero-shot tasks. We also show

TASK	LORA	MOLORA	SEEDED LORA
ANLI R1	**36.30**	34.80	35.20
ANLI R2	**37.50**	34.80	32.50
ANLI R3	**38.75**	32.50	34.42
ARC CHALLENGE	37.20	39.59	**44.45**
ARITH. 2DS	00.00	11.65	**83.70**
ARITH. 4DS	00.00	14.05	**52.15**
BB CAUSAL JUDG. MC	52.11	52.11	**53.16**
BLIMP CAUSATIVE	74.40	75.80	**75.90**
CB	44.64	39.29	30.36
COPA	86.00	87.00	**88.00**
HELLASWAG	**57.87**	57.71	57.46
RTE	53.79	54.87	63.18
TRUTHFULQA MC1	27.29	28.64	**30.35**
WIC	**51.25**	50.94	50.00
WINOGRANDE	70.32	**71.27**	70.32
WSC	37.50	**45.19**	38.46
AVERAGE	44.05	45.63	**52.47**

Table 1. Zero-shot accuracy of LoRA, MoLoRA, and Seeded LoRA (ours) on multiple evaluation tasks. All models were fine-tuned using instruction-tuning with Llama 2 as the base model.

that Seeded LoRA can add the ability to generate appropriate responses for most prompts to a pretrained model (Ouyang et al., 2022; Longpre et al., 2023). Additionally, we study different routing methods commonly used in MoA models, and highlighting subtle failures that negatively affect the performance of the final model (refer to Appendix A).

We also established a theoretical foundation linking Seeded LoRA's operational principles with Grouped Convolutions, showing that under specific initializations, Seeded LoRA operates similarly to grouped convolutions. This equivalence sheds light on the method's underlying efficiency and offers a bridge between PEFT methods and established convolutional techniques.

6. Limitations & Future Work

Despite Seeded LoRA's demonstrated efficacy in enhancing the zero-shot performance of LLMs across a variety of tasks while adding chatbot capabilities to pretrained models, there are inherent limitations that require further exploration:

Past models are unseeded While future models can be initialized via Seeded LoRA, currently available models are not seeded and as such not initialized in the same optimization subspace.

Inherent limits of averaging While a simple average of expert outputs works in Seeded LoRA, this has inherent limits as ineffective experts add more and more noise decreasing the signal to noise ratio. As such, when too many experts are merged more advanced weighted averaging techniques will become necessary.

Scalable Expert Management To address scalability, further research should explore how to develop efficient algorithms for expert selection and routing that minimize computational overhead. Techniques such as sparse expert selection, where only a subset of the most relevant experts are activated for a given input, could improve Seeded LoRA's performance.

Number of Experts Determining an optimal number of experts for a given task or dataset remains an open question. Techniques for dynamically adjusting the number of experts based on task complexity or data characteristics could be beneficial.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Reproducibility Statement

The code to fine-tune using LoRA, MoLoRA, and Seeded LoRA, as well as the evaluation code, can be found in Github[2].

References

Chaudhary, S. Code alpaca: An instruction-following llama model for code generation. https://github.com/sahil280114/codealpaca, 2023.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Conover, M., Hayes, M., Mathur, A., Xie, J., Wan, J., Shah, S., Ghodsi, A., Wendell, P., Zaharia, M., and Xin, R. Free dolly: Introducing the world's first truly open instruction-tuned llm, 2023.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332, 2022.

Dou, S., Zhou, E., Liu, Y., Gao, S., Zhao, J., Shen, W., Zhou, Y., Xi, Z., Wang, X., Fan, X., Pu, S., Zhu, J., Zheng, R.,

[2]https://github.com/alexrs/SeededLoRA

Gui, T., Zhang, Q., and Huang, X. Loramoe: Alleviate world knowledge forgetting in large language models via moe-style plugin, 2024.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. Stabilizing the lottery ticket hypothesis. *arXiv preprint arXiv:1903.01611*, 2019.

Frankle, J., Dziugaite, G. K., Roy, D., and Carbin, M. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pp. 3259–3269. PMLR, 2020.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

Ghorbani, B., Krishnan, S., and Xiao, Y. An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*, pp. 2232–2241. PMLR, 2019.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Gur-Ari, G., Roberts, D. A., and Dyer, E. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*, 2018.

Gururangan, S., Li, M., Lewis, M., Shi, W., Althoff, T., Smith, N. A., and Zettlemoyer, L. Scaling expert language models with unsupervised domain discovery, 2023.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition, 2015.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models, 2021.

Ilharco, G., Ribeiro, M. T., Wortsman, M., Gururangan, S., Schmidt, L., Hajishirzi, H., and Farhadi, A. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., Casas, D. d. l., Hanna, E. B., Bressand, F., et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K. (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012a. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012b.

Lee, A. N., Hunter, C. J., and Ruiz, N. Platypus: Quick, cheap, and powerful refinement of llms, 2023.

Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., and Chen, Z. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

Li, G., Hammoud, H. A. A. K., Itani, H., Khizbullin, D., and Ghanem, B. Camel: Communicative agents for "mind" exploration of large scale language model society, 2023.

Li, M., Gururangan, S., Dettmers, T., Lewis, M., Althoff, T., Smith, N. A., and Zettlemoyer, L. Branch-train-merge: Embarrassingly parallel training of expert language models, 2022.

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation, 2021.

Lian, W., Goodson, B., Pentland, E., Cook, A., Vong, C., and "Teknium". Openorca: An open dataset of gpt augmented flan reasoning traces. https://https://huggingface.co/Open-Orca/OpenOrca, 2023.

Lim, W. Y. B., Luong, N. C., Hoang, D. T., Jiao, Y., Liang, Y.-C., Yang, Q., Niyato, D., and Miao, C. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(3): 2031–2063, 2020.

Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., and Raffel, C. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning, 2022.

Longpre, S., Hou, L., Vu, T., Webson, A., Chung, H. W., Tay, Y., Zhou, D., Le, Q. V., Zoph, B., Wei, J., and Roberts, A. The flan collection: Designing data and methods for effective instruction tuning, 2023.

Luo, Y., Yang, Z., Meng, F., Li, Y., Zhou, J., and Zhang, Y. An empirical study of catastrophic forgetting in large language models during continual fine-tuning, 2023.

Mangrulkar, S., Gugger, S., Debut, L., Belkada, Y., Paul, S., and Bossan, B. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft, 2022.

Masoudnia, S. and Ebrahimpour, R. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42: 275–293, 2014.

Mukherjee, S., Mitra, A., Jawahar, G., Agarwal, S., Palangi, H., and Awadallah, A. Orca: Progressive learning from complex explanation traces of gpt-4, 2023.

Muqeeth, M., Liu, H., Liu, Y., and Raffel, C. Learning to route among specialized experts for zero-shot generalization, 2024.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback, 2022.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Sukhbaatar, S., Golovneva, O., Sharma, V., Xu, H., Lin, X. V., Rozière, B., Kahn, J., Li, D., tau Yih, W., Weston, J., and Li, X. Branch-train-mix: Mixing expert llms into a mixture-of-experts llm, 2024.

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models, 2023.

Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. PMLR, 2022.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. Aggregated residual transformations for deep neural networks, 2017.

Zadouri, T., Üstün, A., Ahmadian, A., Ermiş, B., Locatelli, A., and Hooker, S. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning, 2023.

Zhang, X., Zhou, X., Lin, M., and Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6848–6856, 2018.

Zhu, Y., Wichers, N., Lin, C.-C., Wang, X., Chen, T., Shu, L., Lu, H., Liu, C., Luo, L., Chen, J., and Meng, L. Sira: Sparse mixture of low rank adaptation, 2023.

A. The Pitfalls of Routing: Analysis of Subtle Routing Failures

While the main contribution of this paper is a simple approach that allows for collaborative finetuning without any routing, we did extensive experiments of routing approaches. In this section, we highlight subtle failures and show how to debug routing approaches to be able to develop routing methods that might outperform Seeded LoRA.

Experimental Setup To investigate the impact of the routing mechanism, we employed Unsupervised Domain Discovery (Gururangan et al., 2023) to cluster a selected dataset into multiple smaller datasets via k-means. Subsequently, a *seed* expert, several domain-specific experts, and a routing layer designed for expert selection through soft merging were developed and trained. Comparative analyses were conducted between Seeded LoRA, LoRA, and MoLoRA, examining configurations with 5, 10, 15, and 30 experts, while ensuring parameter and computational resources remained consistent across these variations.

We then analyzed MoLoRA's routing layer's capability to accurately assign tokens to appropriate experts. To do this we inspect the router probabilities while evaluating with the EleutherAI Eval Harness(Gao et al., 2023). We mainly track two quantities: (1) aggregate probability mass over all tokens, (2) normalized proportion of top-k experts activities for all tokens. Normalized proportions mean that we keep track of all top-k expert counts and then divide by the total number of activated experts.

The evaluation tasks included HellaSwag, which tests common-sense reasoning, and TruthfulQA, which aims at addressing failures in truthfulness.

The next paragraphs will discuss subtle routing failures that we observed in these experiments.

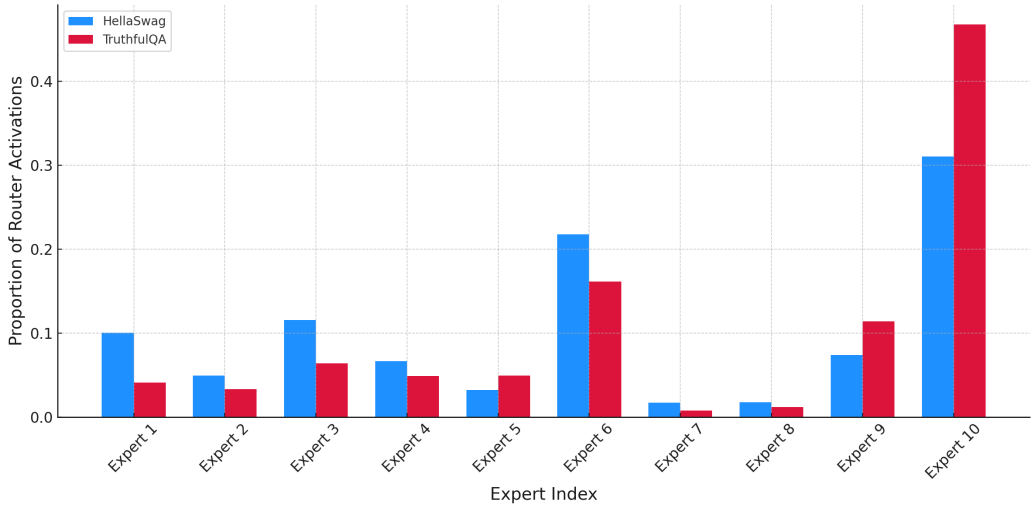

Figure 3. Activation patterns of the MoLoRA routing layer for two distinct tasks, HellaSwag and TruthfulQA, across a set of ten experts. The uniformity in activation distribution suggests similar utilization of experts for both tasks meaning that experts did not specialize in any of the trained 10 domains.

Failure I: Uniform specialization A subtle pattern of expert specialization failure occurs if for a particular evaluation task only a few experts are activated, but that the distribution of experts remains fixed for other tasks. This indicates that all tasks are learned across all experts with one particular weighted average. This pattern is depicted in Figure 3.

Failure II: Linear increase in performance with linear increased k in top-k routing. If a router and experts are trained successfully, then adding the top-k experts in order of their routing probability should increase the performance in a non-linear manner. A non-linear increase indicates, that routing probability p is proportional to the expert specialization. A linear increases indicate, while the router assigns a higher p to some experts, all experts are interchangeable and provide similar performance despite different routing probabilities. This is essentially, non-specialization combined with a uncalibrated router. See Figure 4 for this failure and successful specialization.

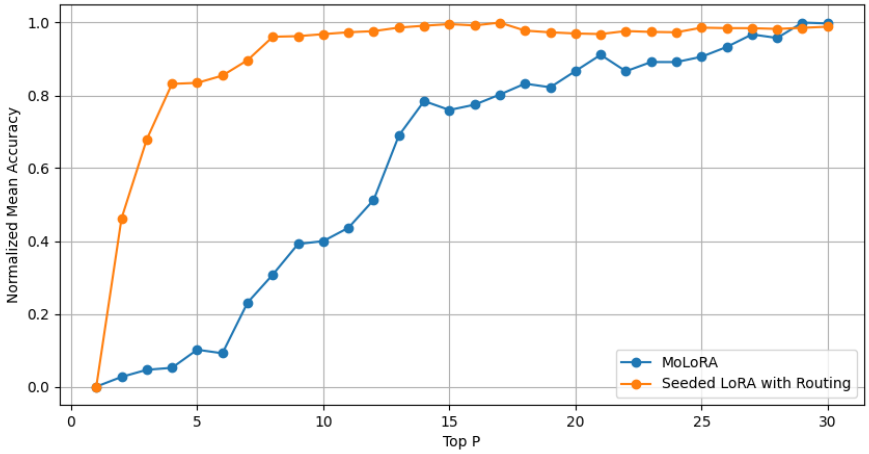

Figure 4. Normalized mean accuracy of Seeded LoRA with Routing, and MoLoRA as a function of the number of top K experts considered for 30 experts. Seeded LoRA, with its independently trained specialized experts, displays a steep increase in performance with a smaller number of top experts, highlighting the benefits of expert specialization. MoLoRA, trained end-to-end, shows a more gradual improvement, reflecting different dynamics in expert utilization.

Failure III: Uniform routing. While uniform routing where each expert has the same probability p often yields better performance with more experts (Jiang et al., 2024) , we show in Section 3.2 that this routing pattern is equivalent to grouped convolution and multi-head processing. As such, despite its improved performance, uniform routing represents a routing failure since performance of the model is the same with and without routing. We show this experimentally for Seeded LoRA with and without routing.

Discussion. Here we depicted common routing failures. Seeded LoRA shows that through the adoption of a *seed* Expert and the application of uniform routing, it is possible to avoid these challenges while simplifying the architecture. We believe that routers can be trained to improve the performance of LoRA MoE approaches, but Seeded LoRA is a strong baseline that we are unable to beat with any current routing approaches. The failure cases in this section can be used to develop routing mechanisms that improve over Seeded LoRA.

B. Convolutions

Convolutional Neural Networks (CNNs) CNNs automate feature extraction from images using layers of convolutional kernels. These kernels, through the convolution operation, identify patterns and features within the input data, making them essential for tasks such as image and video recognition, image classification, and medical image analysis. The convolution operation is mathematically represented as:

$$F(i,j) = (K * X)(i,j) = \sum_m \sum_n K(m,n)X(i-m, j-n) \tag{6}$$

where F is the feature map resulting from applying the kernel K to the input image X at coordinates (i, j).

Convolutional Kernels Convolutional kernels are the core components of CNNs, allowing the network to capture spatial hierarchies of features. Early layers might capture basic patterns such as edges and textures, while deeper layers combine these features to detect more complex patterns. The design of CNN architectures such as ResNet (He et al., 2015) demonstrates how deep networks can effectively learn a wide variety of features by applying convolutional kernels across multiple layers.

Grouped Convolutions Grouped convolutions, introduced in (Krizhevsky et al., 2012a), extend the convolutional operation by dividing the input and kernels into groups, allowing each group to perform convolutions independently. This method reduces computational requirements and parameters while maintaining the network's effectiveness. ResNeXt (Xie et al., 2017) leverages grouped convolutions, introducing the concept of cardinality to efficiently scale the model's capacity. This approach demonstrates the significant advantages of grouped convolutions in deep learning architectures:

$$F_g = K_g * X_g \tag{7}$$

where F_g represents the feature map produced by the g^{th} group's convolution of kernel K_g with input X_g.

C. Data

The experiments for Seeded LoRA leverage a composite dataset[3] for instruction fine-tuning containing a mix of general knowledge, code, and mathematics, totaling 282,360 data points. Instruction fine-tuning contrasts with traditional supervised fine-tuning, which primarily aims to correlate input data with corresponding outputs. The data originates from various sources:

- `Open-Orca/OpenOrca` (Lian et al., 2023): collection of augmented FLAN (Wei et al., 2022) data that aligns with the distributions outlined in the Orca paper (Mukherjee et al., 2023).
- `TokenBender/code_instructions_122k_alpaca_style`[4]: coding questions following the Alpaca template.
- `camel-ai/math` (Li et al., 2023): composed of 50K problem-solution pairs obtained using GPT-4.
- `yahma/alpaca-cleaned` (Taori et al., 2023): contains a cleaned version of the Alpaca dataset.
- `garage-bAInd/Open-Platypus` (Lee et al., 2023): dataset focused on improving LLM logical reasoning skills and was used to train the Platypus2 models.
- `sahil2801/CodeAlpaca-20k` (Chaudhary, 2023): contains 20K code problems in the Alpaca format.
- `c-s-ale/dolly-15k-instruction-alpaca-format`: cleaned and alpaca formatted version of Dolly (Conover et al., 2023), a corpus of more than 15,000 records generated by thousands of Databricks employees.
- `hendrycks/competition_math` (Hendrycks et al., 2021): consists of problems from mathematics competitions, including the AMC 10, AMC 12, AIME, and more. Each problem has a full step-by-step solution, which can be used to teach models to generate answer derivations and explanations.
- `gsm8k` (Cobbe et al., 2021): dataset of 8.5K high quality linguistically diverse grade school math word problems. The dataset was created to support the task of question answering on basic mathematical problems that require multi-step reasoning.

DATASET	COUNT	PERCENTAGE (%)
OPEN-ORCA/OPENORCA	70565	24.99
TOKENBENDER/CODE_INSTRUCTIONS_122K_ALPACA_STYLE	60979	21.60
CAMEL-AI/MATH	50000	17.71
YAHMA/ALPACA-CLEANED	25880	9.17
GARAGE-BAIND/OPEN-PLATYPUS	24926	8.83
SAHIL2801/CODEALPACA-20K	20022	7.09
C-S-ALE/DOLLY-15K-INSTRUCTION-ALPACA-FORMAT	15015	5.32
HENDRYCKS/COMPETITION_MATH	7500	2.66
GSM8K	7473	2.65

Table 2. Distribution of elements and their respective percentages across various datasets.

[3]https://huggingface.co/datasets/alexrs/seededlora-data

[4]https://huggingface.co/datasets/TokenBender/code_instructions_122k_alpaca_style

D. Baselines Training Details

For training Seeded LoRA and MoLoRA experts, the following hyperparameters per expert, for a total of 9, were used:

- Rank: 16 (dimensionality of the low-rank adaptation space)
- LoRA Alpha: 8
- LoRA Dropout: 0.05
- Epochs: 2

The baseline LoRA adapter was trained using:

- Rank: 128 (dimensionality of the low-rank adaptation space)
- LoRA Alpha: 64
- LoRA Dropout: 0.05
- Epochs: 2

E. Seeded LoRA Fine-Tuning using Unsupervised Domain Discovery

Building upon the foundation laid by c-BTM (Gururangan et al., 2023), we experimented with Unsupervised Domain Discovery to create clusters to train experts.

The process begins by segmenting the data using k-means clustering. This divides the data (denoted by X with N samples) into K distinct clusters (C). Each cluster is characterized by a centroid (μ_j), representing the average feature vector of its members. The k-means algorithm aims to minimize the inertia, ensuring data points within each cluster are similar.

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (\|x_i - \mu_j\|^2)$$

This method is flexible in its data representation. You can use any encoding method that captures the dataset's information suitable for unsupervised domain discovery. In this case, we create embeddings of our data. Afterwards, experts are trained as shown in Section 3.3.

Results for this setting can be seen in Appendices I and J, and an overview of this training method can be seein in Figure 5.

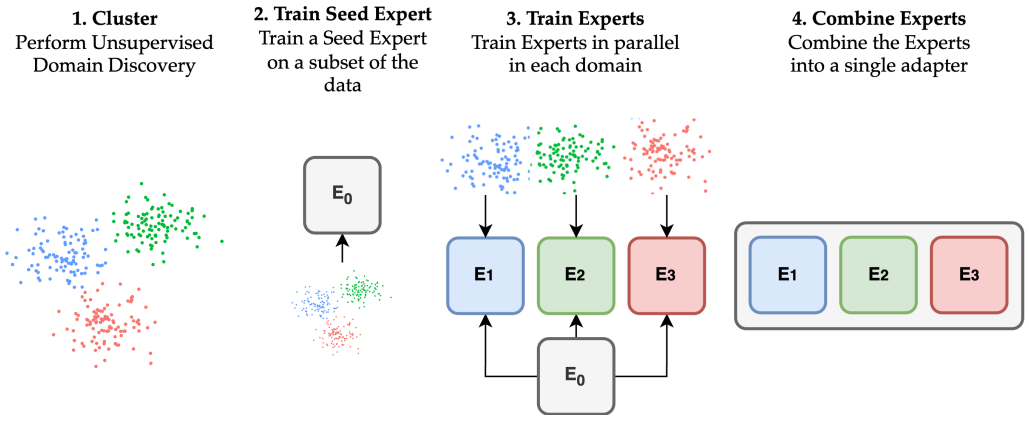

Figure 5. Seeded LoRA fine-funing using Unsupervised Domain Discovery. This method contains four stages: 1) **Unsupervised Domain Discovery** on an unlabelled dataset to create domain-specific datasets. 2) ***Seed Expert* Training**. 3) ***Cluster-Specific Expert* Training**. 4) **Expert Combination**.

F. Evaluation results for Seeded LoRA Experts trained on individual datasets

Table 3 contains the evaluation results for Seeded LoRA with experts trained on the individual datasets.

TASK	SEED EXPERT	EXP. 0	EXP. 1	EXP. 2	EXP. 3	EXP. 4	EXP. 5	EXP. 6	EXP. 7	EXP. 8
ANLI R1	36.10	34.70	35.30	33.90	34.00	31.90	35.20	37.00	34.70	38.00
ANLI R2	35.50	34.20	34.30	33.40	33.40	31.20	34.30	37.50	32.30	33.90
ANLI R3	34.67	35.92	34.33	32.58	33.42	32.67	33.50	35.83	33.25	35.33
ARC CHALLENGE	43.77	34.73	46.16	40.96	45.56	42.32	45.22	44.54	42.75	43.34
ARITHMETIC 2DS	54.00	00.00	94.50	46.35	72.65	58.45	91.10	96.25	53.30	43.65
ARITHMETIC 4DS	37.10	00.00	52.80	39.45	44.70	39.90	50.35	58.35	37.90	37.00
BB C.J. MC	50.53	52.11	47.89	49.47	47.89	54.21	50.53	52.63	52.11	51.58
BLIMP CAUSATIVE	76.50	68.10	75.00	75.30	74.10	73.70	76.40	75.70	77.80	77.30
CB	26.79	26.79	39.29	21.43	30.36	21.43	28.57	32.14	28.57	26.79
COPA	88.00	88.00	86.00	85.00	88.00	86.00	87.00	88.00	88.00	88.00
HELLASWAG	57.12	57.97	57.04	56.84	57.77	57.25	57.12	57.42	57.38	57.22
RTE	60.65	52.71	63.18	51.62	64.98	58.84	61.01	59.21	59.93	58.84
TRUTHFULQA MC1	30.35	31.95	28.52	33.90	34.39	31.09	27.91	28.76	28.89	28.03
WIC	50.16	50.00	49.69	50.00	49.84	50.00	50.00	50.00	50.00	50.00
WINOGRANDE	69.61	70.96	71.35	68.51	70.72	70.24	70.40	69.77	69.69	69.77
WSC	39.42	36.54	40.38	36.54	41.35	36.54	50.96	37.50	41.35	38.46
AVERAGE	50.26	42.16	53.48	47.20	51.44	48.48	53.09	53.78	49.24	48.57

Table 3. Zero-shot accuracy of Seeded LoRA experts on multiple evaluation tasks for experts trained on individual datasets.

G. Evaluation results for Seeded LoRA trained on individual datasets with no *Seed Expert*

Table 4 contains the evaluation results for Seeded LoRA trained on individual datasets with no *Seed Expert*.

TASK	**SEEDED LORA (NO SEED)**	EXP. 0	EXP. 1	EXP. 2	EXP. 3	EXP. 4	EXP. 5	EXP. 6	EXP. 7	EXP. 8
ANLI R1	37.10	36.40	37.00	33.20	38.20	32.20	35.60	35.70	36.30	35.90
ANLI R2	38.60	34.80	38.30	31.50	37.80	31.60	37.10	38.60	37.60	36.80
ANLI R3	37.83	35.25	35.42	33.25	36.25	32.00	37.67	36.08	37.17	37.83
ARC CHALLENGE	43.34	34.22	44.80	40.36	44.54	42.24	43.43	44.28	42.58	42.66
ARITHMETIC 2DS	49.80	00.00	93.90	46.00	44.55	44.45	51.90	52.85	50.10	48.90
ARITHMETIC 4DS	37.85	00.00	45.15	40.20	41.90	38.40	36.25	35.80	37.05	36.70
BB C.J. MC	52.63	51.05	47.37	48.42	48.95	51.05	55.79	53.16	51.58	52.63
BLIMP CAUSATIVE	74.00	63.70	74.10	75.30	73.70	75.00	75.30	73.20	74.00	73.90
CB	39.29	26.79	37.50	30.36	28.57	17.86	48.21	44.64	48.21	53.57
COPA	87.00	90.00	87.00	84.00	88.00	87.00	87.00	87.00	88.00	87.00
HELLASWAG	57.38	57.81	57.14	56.82	57.35	57.28	57.14	57.41	57.26	57.07
RTE	62.82	52.71	59.21	54.51	63.18	59.57	61.37	57.04	62.45	62.45
TRUTHFULQA MC1	28.64	31.95	27.29	33.05	33.05	30.35	25.21	27.42	25.46	24.36
WIC	50.00	50.00	50.47	50.00	50.00	50.00	50.00	50.00	49.53	49.84
WINOGRANDE	70.32	71.59	70.09	68.90	70.01	70.56	69.69	69.30	69.06	69.14
WSC	38.46	36.54	44.23	36.54	36.54	36.54	38.46	37.50	38.46	38.46
AVERAGE	50.31	42.04	53.05	47.65	49.53	47.25	50.63	49.99	50.30	50.45

Table 4. Zero-shot accuracy of Seeded, trained on individual datasets without Seed Expert, on multiple evaluation tasks.

H. Seeded LoRA with Seed Expert in the final model

We also experimented with including the *Seed Expert* in the final model. This resulted in an average score of 52.29, showing that the *Seed Expert* does not contribute to the accuracy of the model and only serves to achieve a common initialization subspace.

TASK	SEEDED LORA WITH *seed* EXPERT
ANLI R1	35.70
ANLI R2	32.90
ANLI R3	34.75
ARC CHALLENGE	44.71
ARITHMETIC 2DS	81.85
ARITHMETIC 4DS	49.15
BB CAUSAL JUDGEMENT MC	48.95
BLIMP CAUSATIVE	77.20
CB	30.36
COPA	88.00
HELLASWAG	57.39
RTE	64.98
TRUTHFULQA MC1	31.82
WIC	50.00
WINOGRANDE	70.48
WSC	38.46
AVERAGE	52.29

Table 5. Zero-shot accuracy of Seeded LoRA with *seed* expert in the final model trained on clusters on multiple evaluation tasks.

I. Evaluation results for Seeded LoRA Experts trained on Clusters

Table 6 contains the results for Seeded LoRA trained on clusters using Unsupervised Domain Discovery. Table 7 contains the results for each expert in this model.

TASK	SEEDED LORA
ANLI R1	34.90
ANLI R2	32.10
ANLI R3	35.08
ARC CHALLENGE	44.62
ARITHMETIC 2DS	85.20
ARITHMETIC 4DS	50.25
BB CAUSAL JUDGEMENT MC	50.00
BLIMP CAUSATIVE	77.30
CB	26.79
COPA	88.00
HELLASWAG	57.48
RTE	64.98
TRUTHFULQA MC1	31.46
WIC	50.00
WINOGRANDE	70.48
WSC	37.50
AVERAGE	52.25

Table 6. Zero-shot accuracy of Seeded LoRA trained on clusters on multiple evaluation tasks.

TASK	SEED EXPERT	EXP. 0	EXP. 1	EXP. 2	EXP. 3	EXP. 4	EXP. 5	EXP. 6	EXP. 7	EXP. 8
ANLI R1	36.10	33.20	36.30	37.10	34.00	34.10	33.50	34.20	36.50	36.40
ANLI R2	35.50	32.70	33.80	34.10	32.80	33.40	31.30	31.90	32.70	32.50
ANLI R3	34.67	33.25	35.92	33.58	32.67	32.92	33.08	33.33	33.33	32.50
ARC CHALLENGE	43.77	41.89	35.92	46.08	43.94	42.83	41.81	43.69	45.05	45.22
ARITHMETIC 2DS	54.00	83.70	00.00	92.30	45.95	93.25	32.85	76.35	67.90	79.20
ARITHMETIC 4DS	37.10	49.35	00.00	45.45	38.50	55.35	33.60	43.10	41.70	44.35
BB C.J. MC	50.53	52.11	52.11	47.89	47.37	51.58	48.95	50.53	47.89	47.37
BLIMP CAUSATIVE	76.50	73.00	65.60	75.80	77.60	76.90	78.00	77.50	76.00	75.40
CB	26.79	44.64	35.71	19.64	16.07	26.79	23.21	35.71	26.79	26.79
COPA	88.00	87.00	90.00	88.00	87.00	88.00	86.00	88.00	87.00	88.00
HELLASWAG	57.12	57.47	57.93	57.48	57.24	57.31	56.94	57.03	57.17	57.33
RTE	60.65	51.62	53.07	62.82	65.70	63.18	64.26	59.57	61.73	63.18
TRUTHFULQA MC1	30.35	30.60	32.19	30.48	31.33	32.19	33.05	31.21	30.72	30.11
WIC	50.16	50.00	50.00	50.00	49.53	50.00	50.00	50.00	50.31	50.16
WINOGRANDE	69.61	70.88	70.64	71.03	70.48	71.51	70.48	71.82	70.48	70.48
WSC	39.42	36.54	36.54	41.35	52.88	36.54	36.54	36.54	47.12	41.35
AVERAGE	49.39	51.74	42.85	52.06	48.94	52.86	47.09	51.28	50.77	51.27

Table 7. Zero-shot accuracy of Seeded LoRA experts, each one trained on a different cluster, on multiple evaluation tasks.

J. Evaluation results for SharedLoRa trained on Clusters with no *Seed Expert*

Table 8 contains the evaluation results for Seeded LoRA trained on clusters with no *Seed Expert*.

TASK	**SEEDED LORA (NO SEED)**	EXP. 0	EXP. 1	EXP. 2	EXP. 3	EXP. 4	EXP. 5	EXP. 6	EXP. 7	EXP. 8
ANLI R1	38.60	33.80	34.90	37.30	37.10	36.40	37.50	37.50	36.10	36.40
ANLI R2	37.30	32.20	34.20	38.90	38.30	34.10	36.90	37.10	37.90	36.50
ANLI R3	37.00	33.75	32.50	38.25	36.92	35.42	37.83	37.75	38.17	37.83
ARC CHALLENGE	43.52	43.77	42.75	43.94	43.94	42.92	43.26	43.09	42.83	43.69
ARITHMETIC 2DS	51.75	75.60	89.70	65.75	49.80	47.50	49.95	49.40	50.40	50.30
ARITHMETIC 4DS	37.45	51.85	54.45	36.75	36.75	38.00	36.60	36.85	36.35	36.25
BB C.J. MC	53.68	52.11	52.63	50.00	48.95	48.42	48.42	46.32	51.05	52.11
BLIMP CAUSATIVE	74.70	75.70	71.00	75.10	73.80	77.30	74.30	77.20	73.60	74.10
CB	39.29	33.93	08.93	32.14	44.64	28.57	44.64	39.29	42.86	42.86
COPA	88.00	86.00	88.00	88.00	87.00	87.00	87.00	86.00	87.00	88.00
HELLASWAG	57.20	57.32	57.60	57.22	57.09	57.06	56.87	57.08	57.28	57.14
RTE	63.54	66.06	62.82	60.29	63.54	62.82	62.45	63.18	63.54	62.82
TRUTHFULQA MC1	28.52	29.38	34.39	27.91	28.27	30.35	28.27	29.74	24.97	25.58
WIC	50.00	50.31	50.00	50.00	49.53	50.16	49.69	49.69	49.84	49.84
WINOGRANDE	69.38	70.32	70.88	69.77	69.53	70.01	69.22	69.61	68.90	69.61
WSC	38.46	36.54	36.54	38.46	39.42	45.19	38.46	40.38	37.50	38.46
AVERAGE	50.52	51.79	51.33	50.61	50.28	49.45	50.08	50.01	49.89	50.09

Table 8. Zero-shot accuracy of Seeded LoRA, trained on Clusters without Seed Expert, on multiple evaluation tasks.