A Case Study of Instruction Tuning with Mixture of Parameter-Efficient Experts

Oleksiy Ostapenko[1,2,5] **Lucas Caccia**[1,3,5] **Zhan Su**[4] **Nicolas Le Roux**[1,2,3,5]
Laurent Charlin[1,6,7] **Alessandro Sordoni**[5]
[1]Mila - Quebec AI Institute, [2]Université de Montréal, [3]McGill University,
[4]University of Copenhagen, [5]Microsoft Research, [6]HEC Montréal, [7]Canada CIFAR AI Chair

Abstract

We study the applicability of mixture of parameter-efficient experts (MoPEs) for instruction-tuning large decoder-only language models. Recent literature indicates that MoPEs might enhance performance in specific multi-task instruction-following datasets. In this paper, we extend such previous results and study the applicability of MoPEs in settings previously overlooked: a) with open-domain instruction-following datasets; b) with recent decoder-only models, and c) with downstream out-of-distribution test sets. We build on top of LLaMA1-13B/-7B and LLaMA2-13B. We study different variants of learned routing, namely per-example routing ([PE]), and a more expensive per-token ([PT]) routing. Overall, we are unable to substantiate strong performance gains observed in related studies in our setting. We observe occasional enhancements of LLAMA2 fine-tuned on the Open Platypus dataset in 0-shot SNI evaluation and TruthfulQA evaluation after fine-tuning on a subset of Flan. We also shed some light on the inner workings of MoPEs by comparing different routing strategies. We find that [PE] routing tends to collapse at downstream evaluation time reducing the importance of the router's application. Out code will be publicly released as part of this repository https://github.com/microsoft/mttl.

1 Introduction

While fine-tuning often relies on end-to-end training of a dense base model, several recent works have demonstrated the efficacy of applying a mixture of parameter-efficient experts (MoPEs) techniques for fine-tuning a densely pretrained model on multi-task data [33, 2, 23]. Similar to mixture-of-expert approaches, MoPEs build on the intuition that each example or token should be processed by a subset of specialized experts. MoPEs differ from standard mixture-of-experts approaches in that each expert is a parameter-efficient adapter [10] and the mixture is obtained by combining the parameters of each expert. Concurrent work by Zadouri et al. [33] successfully applies MoPEs to encoder-decoder (T5) models in a multi-task dataset (T0 [21]) demonstrating performance comparable to full fine-tuning at a fraction of its cost.

In this work, we extend previous results and study MoPEs under different experimental conditions, i.e. a) with recent open-domain instruction following datasets such as Platypus [12], Flan-100K and Evol-Instruct [32], b) with large decoder-only models such as LLAMA2-13b, and c) by testing them on out-of-distribution tasks, zero-shot and few-shot. Additionally, we study different design choices for building MoPEs on top of decoder-only transformers: we ablate per-example and per-token routing with both LoRA [11] and IA3 [15] adapters. When using LoRA adapters, we explore a more efficient type of expert consolidation, namely consolidation before the outer-product of the low-rank adapters [2]. Across tasks, our results show that it is difficult to see any improvements with respect to the single-expert baseline, which boils down to standard parameter-efficient fine-tuning.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

We find that the more efficient per-example routing performs similarly to the per-token routing used by [33]. We find that the more efficient way of consolidating experts before the outer-product seems to hurt performance only when applying per-token routing. Taken together, our results indicate that the performance gains reported in the recent work by [33] do not hold in our setting, thus pointing towards doubting the relevance of MoPEs in more general scenarios.

2 Methods

2.1 Parameter-efficient fine-tuning (PEFT)

Parameter-efficient fine-tuning aims to develop methods that enable memory and compute efficient fine-tuning of LLMs. This is usually achieved by infusing a subset of trainable parameters into a large frozen backbone, these infused parameters are also known as adapters [10]. Two prominent examples of PEFT methods are LoRA [11] and IA3 [15].

LoRA infuses a set of learnable low-rank matrices A and B that approximate a weight matrix of the shape identical to the original layer's matrix. The activation of a transformer layer becomes:

$$h = W_0 x + s * AB^T x, \tag{1}$$

where $B \in \mathbb{R}^{d_{in} \times r}$, $A \in \mathbb{R}^{d_o \times r}$, W_0 are the frozen weights of the layer that LoRA is being applied to and $s \geq 1$ is a trainable hyperparameter. IA3 introduces a more efficient way of fine-tuning that consists of scaling the layer's output with a learnable vector l:

$$h = l \odot (W_0 x). \tag{2}$$

Both types of adapters can be applied to any linear layer in the transformer model. While other PEFT methods exists [1, 18], we limit our attention to LoRA and IA3 adapters due to their suitability to MoE-based training as previously demonstrated by [2, 33].

2.2 Mixture of Experts (MoE)

In the context of transformer models [26], mixture-or-experts methods usually come in two flavours: ones that use external routing information like cluster assignments [13, 8, 20]; and ones that learn the routing end-to-end. In the latter category, which is particularly relevant in the context of this work, some of the feed-forward layers in the transformer are replaced with a set of K experts $\{E(\cdot;\theta_0)...E(\cdot;\theta_K)\}$ and a router R. The router R is usually another feed-forward network that produces a k-dimensional vector indicating the routing probabilities for each expert. Routing is usually realized through weighted averaging of the outputs of the experts, i.e. the output of an MoE layer is defined as $h = \sum_{k=1}^{K} R(x)_k E(x;\theta_k)$, where we denote parameters of expert k as θ_k. Such routing can be prohibitively expensive (at least in a vanilla implementation) as each example has to be passed through each of the activated experts. To tackle this limitation several works proposed to use sparse routing, where only top k experts are used [7].

The majority of existing MoE methods are designed for the pre-training phase. More recently, several studies have begun to explore the application of MoEs in the fine-tuning phase, either by continuously fine-tuning pre-trained MoEs [23] or by applying MoE techniques atop dense pre-trained models [2, 33]. In this paper, we study the latter approaches, which we present in the next section.

2.3 Mixture of Parameter-Efficient Experts (MoPEs)

Recent work studied fine-tuning of LLMs using a mixture of parameter-efficient experts (MoPEs), which are MoEs that use PEFT adapters such as LoRA as experts [2, 33, 17]. MoPEs can be considered as a way of increasing capacity during fine-tuning of a dense pre-trained model and are expected to inherit some desirable properties of the MoEs, such as modularity and specialization.

Merging The output of a MoPE layer is computed by applying the average expert, whose parameters are weighted combination of all experts' parameters at the MoPE layer with weights coming from the router: this ensures scalability to a large number of experts, given that there is no need to compute the output of each expert independently, and differentiability of the expert routing operation. Given

the representations at a layer in a LLM, the output of MoPEs is computed as:

$$h = E(x; \overline{\theta}),\ \overline{\theta} = \sum_{k=1}^{K} R(x)_k \theta_k. \tag{3}$$

For example, in the case of LoRA experts, MoPEs compute:

$$\overline{AB} = \sum_{k=1}^{K} R(x)_k A_k B_k^\top,\ \ h = W_0 x + s * \overline{AB} x, \tag{4}$$

which might be expensive given that, in the naive formulation, this requires explicitly building the $d_{in} \times d_{out}$ matrix $\overline{AB}$. We call this option MA (merging after the outer product). We can optimize this computation by first computing $B_k^\top x$, for each k, then multiplying it with A_k, and then averaging the outputs. A cheaper alternative is to skip the averaging of the full LoRA matrices and average both A and B before taking the outer product:

$$\bar{A} = \sum_{k=1}^{K} R(x)_k A_k,\ \ \bar{B} = \sum_{k=1}^{K} R(x)_k B_k,\ \ h = W_0 x + s * \bar{A}\bar{B}^\top x, \tag{5}$$

which leads to a less expressive combination given that the rank of the outer product cannot be greater than the rank of the LoRA matrices. We ablate these two variants in our experiments.

Routing LLMs compute a hidden representation for each token in the input sequence, i.e. $x \in \mathbb{R}^{s \times d}$, where s is the sequence length. Therefore, routing can be done both per-example (PE) [17] or per-token (PT) [33]. Per-token routing computes different routing probabilities for each position s_i. Per-example routing conditions the router by taking the average of the input across the sequence dimension and applying the resulting routing to all tokens in x.

3 Experimental results

Methods We test the following methods. Single expert methods which are equivalent to applying a single low-rank adapter with rank 16 are denoted as LoRA$_{(R16)}$. MoPEs variants are denoted with R, standing for *router*. Hence, R[PE, MA]$_{(E8,\ R4)}$ denotes a MoPE with 8 experts, using per-example routing, each is a LoRA adapter with rank 4, and experts are averaged after the outer product application. Other routing variants are denoted according to this nomenclature.

Fine-tuning Our main experiments use LLaMA2-13B as a base model and fine-tune it on the Platypus instruction following dataset [12]. This dataset consists of 25K input-output pairs gathered from different open datasets and curated specifically to enable fast and efficient fine-tuning of LLaMA2 models with PEFT adapters. We also experiment with other instruction datasets such as FLAN-100K, a 100K example subset of the flan dataset[16].

In all experiments, the base model is quantized in 8-bit format [6] and we train PEFT adapters with float-32 precision. At generation time, we load the model in float-16 precision for inference. We use the hyperparameters introduced by [12] for fine-tuning, namely, we train for one epoch with a maximum input length of 4096, cosine learning rate annealing, batch size of 16 and micro-batch size of 1 and the learning rate of 3e-4. We use LoRA ranks of 16 and 32 for LoRA baselines and rank 4 for the MoPEs in order to keep the number of trainable parameters comparable. We fix the LoRA alpha to 16. In preliminary experiments, we noted that further increasing the LoRA rank results in decreased downstream performance.

Metrics To reliably evaluate the properties of our MoPEs, we analyze the following factors: downstream performance, routing diversity and routing entropy. The latter two factors, when considered together, provide valuable insights into the robustness of the routing mechanism, helping us detect potential routing collapses.

To evaluate the ability of the model to generalize to new tasks we use zero- and two-shot evaluation on Super-Natural Instructions dataset [29], which consists of 120 open-ended generation tasks; the MMLU [9] benchmark – a collection of 57 multiple-choice classification tasks formulated in natural language, ARC [5], 0-shot TruthfulQA (TQA) [14] and 10-shot HellaSwag [34].

We measure specialization through the lens of the average (normalized) entropy of the routing distribution:

$$\mathrm{H}(r^{(l)}) = \frac{-\frac{1}{b}\sum_{i=1}^{b}\mathbb{E}[\log(r_i^{(l)})]}{\log(k)}, \tag{6}$$

where b is the batch size, k is the number of experts and r if the output of the router. We can average this quantity over the L layers of the base model to gain a more aggregated view. We measure the diversity of the routing through (normalized) negative mutual information (MI) between examples and routings:

$$-\mathrm{MI}^{(l)} = \frac{-\mathbb{E}[\log(\overline{r}^{(l)})]}{\log(k)} - \mathrm{H}(r^{(l)}), \tag{7}$$

where $\overline{r}^{(l)}$ is the batch-averaged routing for layer l.

3.1 Results

	0-SNI	2-SNI	MMLU	0-ARC	25-ARC	TQA	10-HS	AVG.
LLAMA2 13B	7.13	27.6	55.5	49.2	59.6	36.9	82.2	45.4
LORA (R16)	33.4 ±3.3	54.5 ±0.5	**58.3 ±0.4**	52.9 ±0.1	59.9 ±0.4	**44.1 ±0.3**	82.3 ±0.1	55.1
R[PE] (E8,R4)	37.4 ±4.3	**54.9 ±1.2**	58.23 ±1.1	**54.6 ±0.3**	59.5 ±1.2	41.9 ±0.8	82.3 ±0.2	55.5
R[PE, MA] (E8,R4)	**38.5 ±2.2**	54.8 ±1.2	57.7 ±0.3	54.7±0.8	**60.6 ±0.4**	42.5 ±0.5	82.2 ±0.2	**55.8**
R[PT] (E8,R4)	19.3	54.5	58.6	54.7	59.6	43.4	82.4	53.2
R[PT, MA] (E8,R4)	37.4 ±1.6	54.1 ±0.2	57.01 ±1.0	52.7 ±1/1	59.8 ±0.4	43.1 ±0.3	82.4 ±0.0	55.2

Table 1: LLaMA2 13B fine-tuned with different versions of MoPEs and evaluated on 0-shot SNI, 2-shot SNI (Rouge-L, ↑) [28], 5-shot MMLU [9] (Accuracy, ↑), 0-shot and 25-shot ARC [5], 0-shot TruthfulQA (TQA) [14] and 10-shot HellaSwag [34] downstream tasks. We do not seed R[PT] variant due to its poor performance on 0-shot SNI.

	0-SNI	MMLU	0-ARC	25-ARC	TQA	10-HS
LORA(R16)	49.9	38.2	45.4	49.1	33.7	77.3
R[PE](E4)	48.7	36.8	46.0	48.5	36.0	77.4
R[PE](E8)	**50.5**	**38.3**	45.1	47.8	33.5	77.3
R[PE](E12)	49.1	37.3	47.4	49.7	36.2	77.4
R[PT](E4)	48.8	37.5	45.4	49.7	34.3	77.6
R[PT](E8)	49.9	36.8	47.0	49.1	33.0	77.6
R[PT](E12)	49.2	37.5	**47.4**	**49.7**	**36.2**	**77.8**

Table 2: Results on Flan-100K fine-tuned using LLAMA1 7B as a base model – this is a subset of 100K examples from Flan [16] dataset used in [30]. Here we only tested [PE] and [PT] versions without MA.

Limited gains with MoPEs We show our main result for Open Platypus dataset in Table 1. We stick 8 experts as this performed best in the preliminary experiments. Overall, across all downstream evaluation tasks, MoPEs perform very similarly to the baseline LoRA, with a slight improvement on 0-shot ARC and a slight decrease in performance on TruthfulQA as compared to a single expert baseline LoRA. We also note that the MMLU result we obtained for LoRA is significantly higher than reported in the original Platypus paper[12] (they reported 56.7%). Interestingly, one result stands out – we observe a consistent performance improvement for MoPEs for 0-shot SNI.

	0-SNI
LORA (R16)	**46.79**
R[PE] (E8)	42.08
R[PE, MA] (E8)	42.82
R[PT] (E8)	24.47
R[PT,MA] (E8)	41.02

Table 3: Improved prompt for 0-shot SNI with LLaMA2 13B.

In order to ensure the robustness of these results in the context of 0-shot SNI, we conducted additional evaluations using a slightly alternated prompt [1], that resulted in further performance gains for all models, with simple LoRA now outperforming best MoPE also on 0-shot SNI as shown in Table 3.

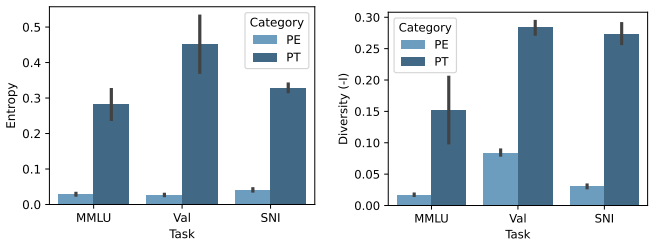
[1]We remove "Now complete the following example" from the prompt before giving the actual test example.

	0-SH. SNI			MMLU (ACC)		
	R	UNIF.$-\mu$.	DST.$-\mu$	R	UNIF.$-\mu$.	DST.$-\mu$
R[PE] (E8)	39.03	19.31	36.32	59.08	55.91	58.57
R[PE, MA] (E8)	40.19	19.41	38.86	57.66	55.91	57.83
R[PT, MA] (E8)	38.34	13.51	13.38	55.80	56.77	57.29

Table 4: Ablation. The effect of using uniform (UNIF-μ) and training dataset average (DST-μ) routing at downstream evaluation time.

Absence of substantial gains for MoPEs method transfer to LLama1 7b (c.f. Table 5). In the following, we use our standard prompt for 0-shot SNI evaluations to ensure consistency. Additionally to Open Platypus, we also experiment with Evol-Instruct (c.f. Table 7) and a 4x larger dataset, Flan-100K. The results for Flan-100K are presented in Table 2. Here we observe slight improvements on 0-shot ARC and a 2.5 percent-points improvement on TruthfullQA. This is a promising result showcasing that further investigation of MoPEs on larger datasets might lead to larger performance gains.

No significant improvement with per-token routing While per-token routing requires more computation as every token is processed by the router, per-example routing only processes a single averaged token for each example. In their recent work [33] demonstrated that per-token routing has an advantage in the context of encoder-decoder architecture as compared to per-example routing[2]. In contrast to [33], we do not observe any significant gains when applying per-token routing. On the contrary, per-token routing appears to be more brittle, as it only works in combination with merging after the outer product (MA) and is more sensitive to wrong routing at test time as we discuss next.



(a) Average routing entropy (b) Average routing diversity

Figure 1: Later-averaged entropy and diversity of routing for PT and PE models with 8 (R[PE](E8), R[PE, MA](E8), R[PT, MA](E8)) calculated on validation examples (valid) and downstream evaluation time (MMLU and 0-shot SNI).

Importance of training and downstream routing In this section, we examine the characteristics and importance of the learned router during the downstream evaluation. To this end, we substitute the learned router at downstream evaluation time with a uniform routing — *UNIF.-*μ, or the average routing distribution, denoted as *DST.-*μ and derived from taking the average routing distribution at each layer on 20% of the training set. In Table 4 we observe that while uniform routing significantly decreases the downstream performance of [PE] routed models at MMLU and 0-shot SNI, we observe a smaller drop is present in the *DST.-*μ case.

To understand why *DST.-*μ still performs well, we examine the average entropy and diversity of routing distribution averaged over layers for the validation set, and downstream datasets, which we plot in Figure 1. As evidenced from Figure 1b, *there is a large drop in routing diversity* between validation (in-distribution) and downstream evaluation on both MMLU and SNI (out-of-distribution). This, coupled with low entropy routing as plotted in Figure 1a, suggests a routing collapse during downstream evaluation, which explains a large performance drop when using uniform routing and an insignificant drop when using dataset average routing for the [PE] variant. For a full picture, we include a per-layer diversity and entropy in Figures 3 and 4.

[2]Importantly, in their version of per-example routing, a separate model was applied to produce example embeddings, while here, similarly to [17], we compute average token.

Unlike the per-example ([PE]) routing, per-token ([PT]) routing exhibits distinct characteristics. First, it has a much larger entropy both in and out-of-distribution. Second, it exhibits a larger diversity as we show in Figure 1b. Both these observations point to the absence of collapse in PT routing which also explains a large drop in performance when disabling routing at evaluation time (reliably seen on SNI in Table 4[3]).

IA3 underpeforms LoRA To evaluate the impact of different adapter architectures, we employ IA3. The results are summarized in Table 6. The validation loss curves are plotted in Figure 2. IA3 significantly under-performs LoRA in our experiments introducing no improvement over the base model. We find that adding more experts does not result in performance gains contrary to the observation of [33]. This can be attributed to the extreme parameter efficiency of IA3, which comes at the cost of lower versatility.

4 Related Work

Instruction tuning Instruction tuning is a technique of fine-tuning LLMs on input-response pairs formulated in natural language. Early progress in this field was mainly achieved through transforming the classical NLP tasks such as summarization, sentiment classification etc., into a unified text-to-text format [19]. This also involved scaling the number of tasks into hundreds [21] or even thousands [4], and expanding the repertoire of instruction templates into hundreds[31]. These works have shown that scaling the data led to increasing improvement of performance across unseen tasks. Recently, a new thread of research has emerged, an open domain instruction fine-tuning, which seeks to fine-tune LLMs on more general purpose instruction datasets [29, 32, 27, 12, 24, 3]. Many of these works rely on synthetic datasets generated by superior LLMs [32, 27, 12, 24] and tune smaller decoder-only models such as LLaMA1 [25] or LLaMA2 [25]. In this work, we follow this recent trend and rely on LLaMA2 as our foundation model. We fine-tune on the recently proposed Open Platypus dataset [12] – a dataset containing 25K input-output samples curated specifically for effective and efficient PEFT-based fine-tuning of the instruction following models.

Mixture of experts Mixture-of-experts in NLP have been developed originally for the pre-training phase with the motivation of scaling the parameter count of large models while keeping the computational cost similar to the dense model by only sparsely activating a subset of experts [22, 7, 36, 35]. Several recent works focused on using MoEs in the context of fine-tuning pre-trained language models. Notably [23] demonstrated that MoE fine-tuned on large corpus of instruction following tasks extracted from classical NLP tasks outperform dense fine-tuning in terms of both 0-shot and few-shot fine-tuning performance on unseen tasks. Similarly to our work, [33] used a parameter-efficient MoEs with experts represented with PEFT modules like low-rank adapters [11]. They tune an encoder-decoder based T5 [19] model on a dataset of 62 tasks obtained through converting classical NLP tasks into instruction following format using large set of prompt templates [21]. Here we follow a recent trend of fine-tuning decoder-only model on a general but relatively small dataset of for open-domain instruction following. The choice of a smaller dataset is justified also by the fact that we start from a larger but also much better base model LLaMA2.

Parameter efficient fine-tuning is a strategy aimed at minimizing the memory and time requirements during the fine-tuning process of Language Models (LMs). It achieves this by training a limited set of additional parameters within selected layers while keeping the backbone of the model frozen [10, 1]. Several prominent PEFT techniques have emerged in recent times. For instance, LoRA [11] introduces low-rank weights, while IA3 [15] focuses on scaling the activations. These approaches are geared towards streamlining and enhancing the efficiency of the fine-tuning process.

5 Conclusion

Our investigation raises doubts about the effectiveness of mixture of parameter-efficient experts (MoPEs) for open-domain instruction fine-tuning, particularly within the constraints of the relatively small fine-tuning datasets employed in this domain. To overcome these limitations, future research

[3]We do not observe a drop on MMLU in [PT] case, arguably because for this particular run routed model attained relatively small MMLU acc. not significantly better than base.

could explore the application of MoPEs in conjunction with external routing priors, such as those derived from data clustering, which may enhance routing quality.

References

[1] Alan Ansell, Edoardo Maria Ponti, Anna Korhonen, and Ivan Vulić. Composable sparse fine-tuning for cross-lingual transfer. *arXiv preprint arXiv:2110.07560*, 2021.

[2] Lucas Caccia, Edoardo Ponti, Zhan Su, Matheus Pereira, Nicolas Le Roux, and Alessandro Sordoni. Multi-head adapter routing for cross-task generalization, 2023.

[3] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

[4] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

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

[6] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

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

[8] Suchin Gururangan, Margaret Li, Mike Lewis, Weijia Shi, Tim Althoff, Noah A Smith, and Luke Zettlemoyer. Scaling expert language models with unsupervised domain discovery. *arXiv preprint arXiv:2303.14177*, 2023.

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

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

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

[12] Ariel N Lee, Cole J Hunter, and Nataniel Ruiz. Platypus: Quick, cheap, and powerful refinement of llms. *arXiv preprint arXiv:2308.07317*, 2023.

[13] Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A Smith, and Luke Zettlemoyer. Branch-train-merge: Embarrassingly parallel training of expert language models. *arXiv preprint arXiv:2208.03306*, 2022.

[14] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.

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

[16] Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. The flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*, 2023.

[17] Mohammed Muqeeth, Haokun Liu, and Colin Raffel. Soft merging of experts with adaptive routing. *arXiv preprint arXiv:2306.03745*, 2023.

[18] Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. Mad-x: An adapter-based framework for multi-task cross-lingual transfer. *arXiv preprint arXiv:2005.00052*, 2020.

[19] Adam Roberts, Colin Raffel, Katherine Lee, Michael Matena, Noam Shazeer, Peter J Liu, Sharan Narang, Wei Li, and Yanqi Zhou. Exploring the limits of transfer learning with a unified text-to-text transformer. 2019.

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

[21] Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*, 2021.

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

[23] Sheng Shen, Le Hou, Yanqi Zhou, Nan Du, Shayne Longpre, Jason Wei, Hyung Won Chung, Barret Zoph, William Fedus, Xinyun Chen, Tu Vu, Yuexin Wu, Wuyang Chen, Albert Webson, Yunxuan Li, Vincent Zhao, Hongkun Yu, Kurt Keutzer, Trevor Darrell, and Denny Zhou. Mixture-of-experts meets instruction tuning:a winning combination for large language models, 2023.

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

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

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[27] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

[28] Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. Super-naturalinstructions:generalization via declarative instructions on 1600+ tasks. In *EMNLP*, 2022.

[29] Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*, 2022.

[30] Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, et al. How far can camels go? exploring the state of instruction tuning on open resources. *arXiv preprint arXiv:2306.04751*, 2023.

[31] Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

[32] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*, 2023.

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

[34] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

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

[36] Simiao Zuo, Xiaodong Liu, Jian Jiao, Young Jin Kim, Hany Hassan, Ruofei Zhang, Tuo Zhao, and Jianfeng Gao. Taming sparsely activated transformer with stochastic experts. *arXiv preprint arXiv:2110.04260*, 2021.

Appendix A Additional Results

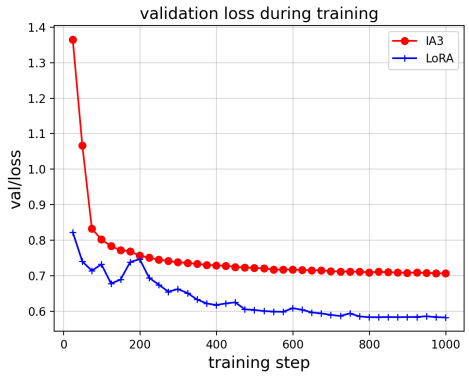


Figure 2: Validation loss converge for LoRA vs. IA3.

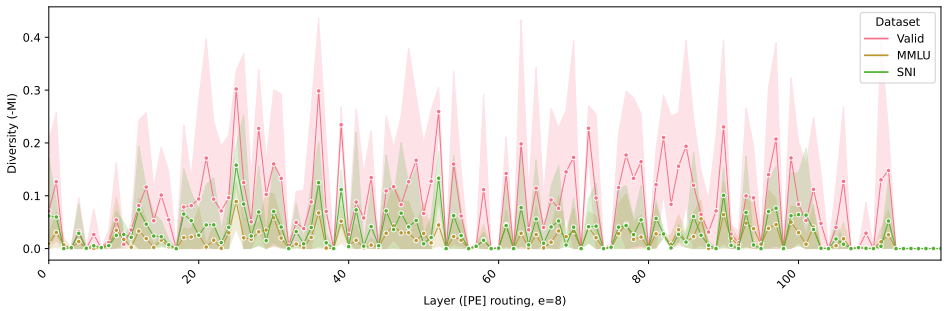


Figure 3: Routing diversity per layer for [PE] routing with 8 experts.

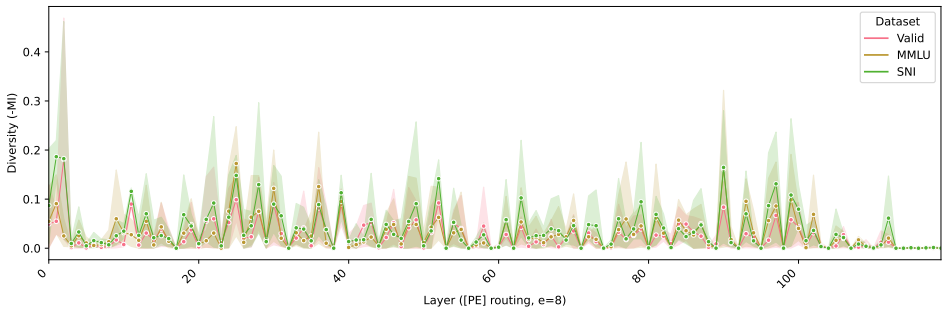


Figure 4: Routing entropy per layer for [PE] routing with 8 experts.

	LLaMA1 13B (base)		LLaMA1 7B (base)	
	MMLU	0-shot SNI	MMLU	0-shot SNI
Base	42.914	8.966	33.16	7.41
LoRA (r4)	50.94 ±0.17	42.57 ±2.0	39.69 ±1.5	38.04 ±0.8
R[PE] (e8)	50.86 ±1.0	37.29 ±3.4	40.43 ±1.4	37.73 ±1.1
R[PE, MA] (e8)	50.62 ±0.5	40.07 ±4.1	40.21 ±1.8	36.69 ±1.1
R[PT] (e8)	51.08 ±0.8	40.92 ±3.3	41.03 ±1.6	37.62 ±0.4
R[PT,MA] (e8)	50.52 ±0.7	40.43 ±2.8 ±0.2	39.28 ±1.2	36.9 ±2.9

Table 5: Result of instruction tuning with different PEFT-MoE flavours and Platypus dataset on top of LLaMA1 13B and 7B models. We report 0-shot SNI performance (Rouge-L) and 5-shot MMLU performance (accuracy).

	MMLU	0-sh. SNI	2-sh. SNI
IA3	55.37	6.74	30.54
R[PE](E4)	55.63	6.69	30.15
R[PE](E12)	55.41	6.71	31.18
R[PE](E20)	55.48	6.71	30.71

Table 6: Results with IA3 adapter and Open Platypus dataset with LLaMA2-13b as base.

	0-SNI	MMLU	0-ARC	25-ARC	TQA	10-HS
LLaMA2 13B	7.13	55.5	49.2	59.6	36.9	82.2
LoRA	17.2	**55.3**	49.9	59.8	47.1	82.4
R[PE] (e4,r=4)	20.6	54.3	50.9	60.3	45.0	82.4
R[PE] (e4,r=8)	24.2	54.7	51.4	60.6	47.0	82.3
R[PE] (e4,r=12)	21.7	54.9	50.3	**60.7**	**47.6**	82.2
R[PT,MA] (e4,r=4)	13.9	55.0	50.8	60.2	43.7	82.3
R[PT,MA] (e4,r=8)	15.9	55.1	51.7	60.2	45.0	82.3
R[PT,MA] (e4,r=12)		54.7	**51.6**	60.6	45.7	82.3

Table 7: Results on Evol Instruct [32] fine-tuned using LLAMA2 13B as a base model. We fine-tuned for 1 epoch with max. sequence length of 1024.