Parameter-Efficient Sparsity Crafting from Dense to Mixture-of-Experts for Instruction Tuning on General Tasks

Haoyuan Wu[♠], **Haisheng Zheng**[♡], **Zhuolun He**[♠,♣], **Bei Yu**[♠]

[♠] The Chinese University of Hong Kong, Hong Kong SAR
[♡] Shanghai Artificial Intelligent Laboratory, China
[♣] ChatEDA Tech, China

Outline

Introduction

- Instruction tuning[1] is a prominent method for training LLMs, which utilizes large-scale, well-formatted instruction data, enabling LLMs to refine their pre-trained representations to comply with human instructions.
- Instruction tuning often encounters performance limitations across multiple tasks due to constrained model capacity.

[1] **Jason Wei et al. (2021). "Finetuned language models are zero-shot learners".** In: *arXiv preprint arXiv:2109.01652*.

- The scaling law[2] suggests that expanding the model's capacity can also improve instruction tuning effectiveness for general tasks[3].
- Nonetheless, most LLMs are pre-trained dense models designed based on transformer architecture, which limits scalability during instruction tuning.
- Sparse Upcycling[4] is proposed for upcycling dense models into sparse activated MoE models, which boast greater capacity.

[2] **Hyung Won Chung et al. (2022). "Scaling instruction-finetuned language models".** In: *arXiv preprint arXiv:2210.11416*.

[3] **Jared Kaplan et al. (2020). "Scaling laws for neural language models".** In: *arXiv preprint arXiv:2001.08361*.

[4] **Aran Komatsuzaki et al. (2023). "Sparse Upcycling: Training mixture-of-experts from dense checkpoints".** In: *International Conference on Learning Representations*.

Challenges

- Given the parameter scale of current LLMs, training such giant models requires updating the weights of experts in the MoE layer, which is constrained by GPU memory resources and computational costs.

Contribution

- We propose an approach, parameter-efficient sparsity crafting (PESC), for the extension of the model capacity efficiently.
- We implement the PESC method for instruction tuning across general tasks, achieving significant performance improvements on various benchmarks.
- We develop Camelidae models, sparse models trained with the PESC method, achieving the best performance across open-source sparse models and demonstrating superior general capabilities compared to GPT-3.5.

Method

An adapter consists of two matrices, $\boldsymbol{W}_{\text{down}} \in \mathbb{R}^{d_1 \times d_2}$ and $\boldsymbol{W}_{\text{up}} \in \mathbb{R}^{d_2 \times d_1}$, coupled with a non-linear function $\sigma(\cdot)$. Here, d_1 and d_2 denote the feature dimensions in the pre-trained models and the adapter's hidden dimension, respectively, with $d_2 < d_1$ typically. Given a feature $\boldsymbol{U} \in \mathbb{R}^{N \times d_1}$ in the pre-trained model, the output of the Adapter module is expressed as:

$$\boldsymbol{U}' = \sigma(\boldsymbol{U}\boldsymbol{W}_{\text{down}})\boldsymbol{W}_{\text{up}} + \boldsymbol{U}. \tag{1}$$

Neil Houlsby et al. (2019). "Parameter-efficient transfer learning for NLP". In: *International Conference on Machine Learning*.

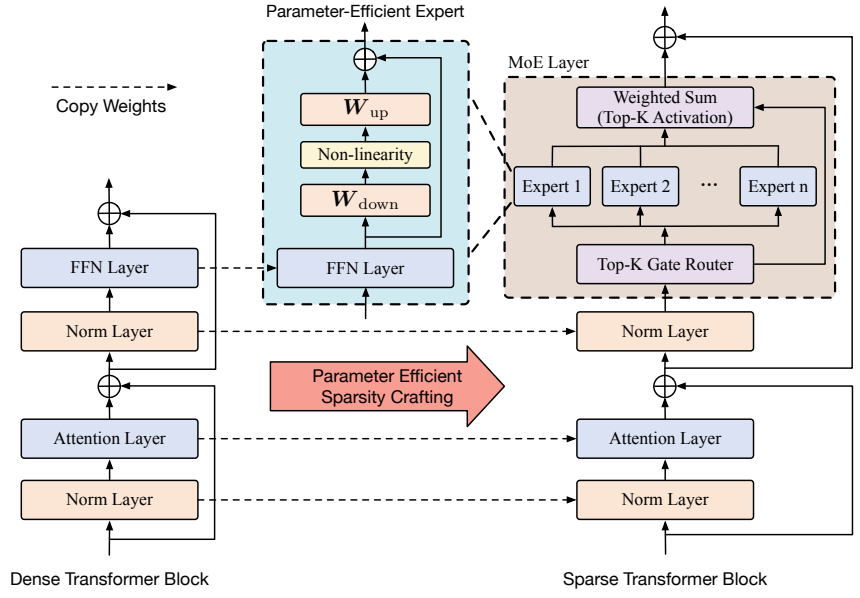
An MoE layer comprises n experts, $\{E_i\}_{i=1}^{n}$, and a router R. The output $\boldsymbol{y}$ for an input $\boldsymbol{x}$ in the MoE layer is computed as:

$$\boldsymbol{y} = \sum_{i=1}^{n} R(\boldsymbol{x})_i E_i(\boldsymbol{x}), \tag{2}$$

where $R(\boldsymbol{x})_i$ represents the output of the gating network for the i-th expert, and $E_i(\boldsymbol{x})$ is the output of the i-th expert.

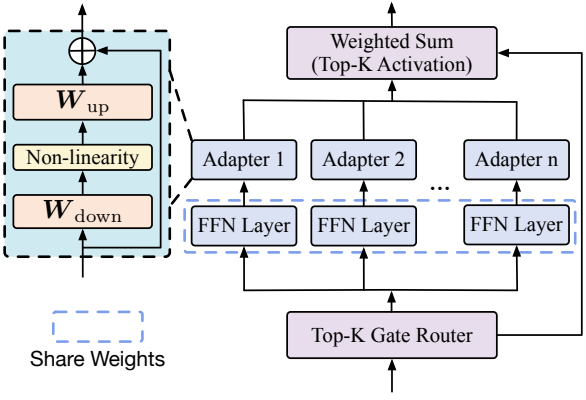
- Sparsity crafting involves a transformative process: substituting the FFN layer F within each block of the dense transformer model with an MoE layer.
- During the initialization phase of sparsity crafting, each expert E_i within the MoE layer is initialized with the FFN layer F.
- To ensure structural coherence, other components, such as the normalization and attention layers, are replicated directly from the dense transformer block.

Parameter-Efficient Sparse Crafting



Overview of the parameter-efficient sparsity crafting with parameter-efficient experts.

Parameter-Efficient Sparse Crafting



Detailed design of the MoE layer for PESC. All the FFN layers share the same weights.

- We update the experi of n inserted adapters to differentiate between experts without altering each expert's original weights replicated from the original FFN layer.
- We also update the shared weights of other components in transformer blocks.

For a given input $\boldsymbol{x}$ to the MoE layer, Equation (2) can be reformulated as:

$$\boldsymbol{y} = \sum_{i=0}^{n} R(\boldsymbol{x})_i A_i(E(\boldsymbol{x})), \tag{3}$$

where $A_i(\boldsymbol{x})$ construct the parameter-efficient expert as follows:

$$A_i(\boldsymbol{x}) = \sigma(\boldsymbol{x}\boldsymbol{W}_{i\text{down}})\boldsymbol{W}_{i\text{up}} + \boldsymbol{x}. \tag{4}$$

We integrated the expert loading balance loss[6] during training for each sparse transformer block to promote uniform expert utilization. With n experts and a batch B containing T tokens, this auxiliary loss $\mathcal{L}$ for experts loading balance is calculated as the scaled dot-product of vectors $\boldsymbol{f}$ and $\boldsymbol{p}$,

$$\mathcal{L} = \alpha \cdot n \cdot \sum_{i=1}^{n} \boldsymbol{f}_i \cdot \boldsymbol{p}_i, \tag{5}$$

where f_i denotes the fraction of tokens dispatched to expert i and p_i represents the fraction of router probability allocated to expert i. α is a multiplicative coefficient for the auxiliary losses.

[6] **William Fedus, Barret Zoph, and Noam Shazeer (2022). "Switch Transformers: Scaling to trillion parameter models with simple and efficient sparsity".** In: *The Journal of Machine Learning Research*.

Experiments

	SlimOrca	Magicoder	MetaMathQA
IDAE-500K	300K	100K	100K
IDAE-720K	360K	180K	180K

The training involved integrating three distinct datasets from varied domains during the instruction tuning phase: SlimOrca[7], Magicoder[8], and MetaMathQA[9] datasets. After filtration and sampling, we can get two instruction datasets including IDAE-500K and IDAE-720K finally.

[7] **Wing Lian et al. (2023).** ***SlimOrca: An Open Dataset of GPT-4 Augmented FLAN Reasoning Traces, with Verification.*** URL: https://huggingface.co/datasets/Open-Orca/SlimOrca.

[8] **Yuxiang Wei et al. (2023). "Magicoder: Source code is all you need".** In: *arXiv preprint arXiv:2312.02120*.

[9] **Longhui Yu et al. (2023). "MetaMath: Bootstrap your own mathematical questions for large language models".** In: *arXiv preprint arXiv:2309.12284*.

Experiments Setting

- We fine-tuned Camel and Camelidae models using IDAE-500K to ensure fair comparisons between dense and sparse models. Specifically, Camel models are dense models while Camelidae models are sparse models with MoE architecture.
- Notably, to further enhance the capabilities of the sparse models, we also utilize IDAE-720K for the instruction-tuning of the Camelidae-pro model.
- All Camelidae models utilize the top-2 gate router.

Comparison with Chat LLMs

	Sparse Chat Models			Dense Chat Models			
	Camelidae 8×34B-pro	Mixtral 8×7B Inst.	DeepSeekMoE 16B Chat	Yi 34B Chat	Llama2 70B Chat	Qwen 72B Chat	GPT-3.5
MMLU (Acc.)	**75.7%** (5-shot)	68.7% (5-shot)	47.2% (5-shot)	74.8% (5-shot)	63.8% (5-shot)	75.0% (5-shot)	70.0% (5-shot)
GSM8K (Acc.)	**79.4%** (5-shot)	71.7% (5-shot)	62.2% (5-shot)	67.6% (5-shot)	59.3% (5-shot)	67.4% (5-shot)	57.1% (5-shot)
MATH (Acc.)	24.0% (4-shot)	22.1% (4-shot)	15.2% (4-shot)	17.3% (4-shot)	10.4% (4-shot)	26.8% (4-shot)	**34.1%** (4-shot)
HumanEval (Pass@1)	**48.8%** (0-shot)	25.6% (0-shot)	42.7% (0-shot)	20.1% (0-shot)	32.3% (0-shot)	47.0% (0-shot)	48.1% (0-shot)
MBPP (Pass@1)	**43.2%** (4-shot)	40.6% (4-shot)	42.2% (4-shot)	41.0% (4-shot)	35.6% (4-shot)	41.8% (4-shot)	-
HellaSwag (Acc.)	85.2% (10-shot)	**86.5%** (10-shot)	72.2% (10-shot)	83.9% (10-shot)	84.8% (10-shot)	85.9% (10-shot)	85.5% (10-shot)
NaturalQuestions (EM)	**31.2%** (0-shot)	22.5% (0-shot)	30.7% (0-shot)	23.7% (0-shot)	30.6% (0-shot)	29.3% (0-shot)	-

Table: Performance of Camelidae-8×34B-pro on academic benchmarks. We present a detailed comparison of the Camelidae-8×34B-pro model with the various open-source sparse chat models and dense chat models. We bold the highest scores among all models.

Dense models vs. Sparse Models

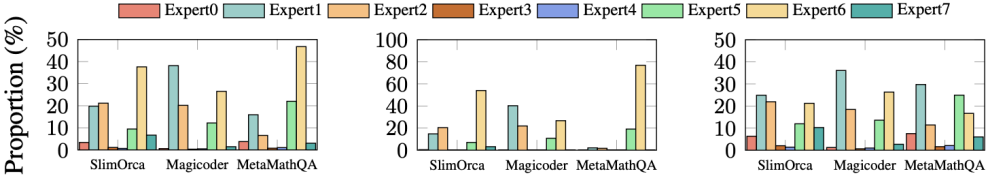
EMNLP 2024

	Camel-7B	Camelidae 8×7B	Camel-13B	Camelidae 8×13B	Camel-34B	Camelidae 8×34B	Camelidae 8×34B-pro
# Total Params	7B	8B	13B	15B	34B	38B	38B
# Activated Params	7B	7B	13B	14B	34B	35B	35B
# Training Instructions	500K	500K	500K	500K	500K	500K	720K
MMLU (Acc.)	47.7	**48.3**	**54.4**	**54.4**	75.3	75.6	**75.7**
HumanEval (Pass@1)	17.7	**18.3**	28.7	**30.6**	42.1	43.9	**48.8**
MBPP (Pass@1)	21.0	**23.4**	30.3	**30.4**	40.6	41.4	**43.2**
GSM8K (Acc.)	40.7	**44.0**	50.2	**52.6**	76.1	78.3	**79.4**
MATH (Acc.)	4.8	**5.8**	8.4	**9.8**	18.2	22.6	**24.0**
PIQA (Acc.)	79.7	**79.9**	**80.9**	**80.9**	82.3	82.7	**83.6**
HellaSwag (Acc.)	**76.8**	**76.8**	79.8	**80.1**	82.6	**83.2**	82.5
Winogrande (Acc.)	71.3	**72.1**	74.6	**74.7**	80.0	**80.9**	80.1
ARC-easy (Acc.)	**75.0**	**75.0**	77.7	**78.8**	86.1	86.2	**86.6**
ARC-challenge (Acc.)	47.9	**49.6**	**54.3**	54.2	63.6	**65.2**	63.3
NaturalQuestions (EM)	17.6	**17.8**	24.7	**26.8**	31.6	**32.2**	31.2
TriviaQA (EM)	**51.0**	**51.0**	57.5	**59.4**	63.3	**63.4**	62.5

Table: Overall performance on all the evaluation benchmarks of dense models (Camel) and sparse (Camelidae) models across different model sizes. We bold the highest scores separately for different model sizes.

Model	# Experts	Avg.	Code	Math	CR	WK	MMLU
Camelidae-4×7B	4	39.6	20.7	24.3	70.2	33.3	49.3
Camelidae-8×7B	8	39.9	20.9	24.9	**70.7**	34.4	48.3
Camelidae-16×7B	16	**40.5**	**21.6**	**25.8**	**70.7**	**35.0**	**49.4**

Table: Evaluation on different numbers of experts in the MoE layers. We bold the highest scores for each grouped benchmark.



(a) Top2 Choice (b) First Choice (c) Second Choice

Proportion of tokens assigned to each expert on different dataset subsets.

THANK YOU!