

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

EMNLP
2024

Introduction

Background:

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

Challenge:

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

Contribution:

Consequently, we introduce parameter-efficient sparsity crafting (PESC), an approach that effectively expands model capacity while synergizing with parameter-efficient fine-tuning techniques. PESC involves inserting adapters into the MoE layers of sparse models, allowing differentiation between experts without altering each expert’s weights in the MoE layers. Our contributions are described as follows:

- We propose a novel approach, parameter-efficient sparsity crafting (PESC), for the efficient extension of the model capacity.
- We implement the PESC method for instruction tuning across general tasks, achieving significant performance 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.

Preliminaries

We provide a brief review of the preliminaries, encompassing adapters, mixture-of-experts architecture and sparse crafting.

Adapters

An adapter consists of two matrices, $\vec{W}_{\text{down}} \in \mathbb{R}^{d_1 \times d_2}$ and $\vec{W}_{\text{up}} \in \mathbb{R}^{d_2 \times d_1}$, 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 $\vec{U} \in \mathbb{R}^{N \times d_1}$ in the pre-trained model, the output of the Adapter module is expressed as:

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

Mixture-of-Experts

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

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

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

Sparse Crafting

Building on the concept of sparsity upcycling, sparsity crafting leverages the weights of dense models.

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

Parameter-Efficient Sparsity Crafting

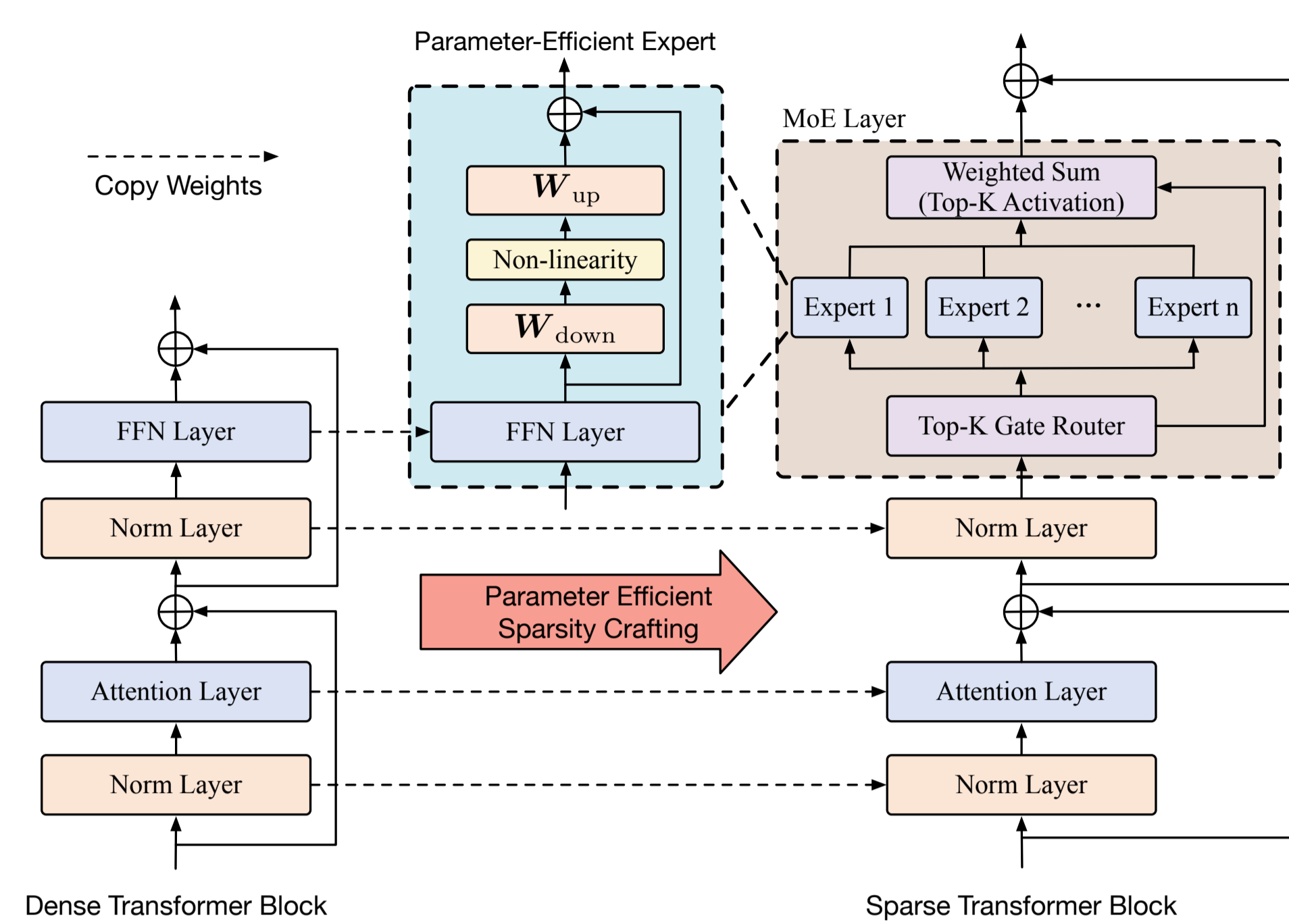


Figure 1. Overview of our parameter-efficient sparsity crafting method built with parameter-efficient experts.

Parameter-Efficient Experts

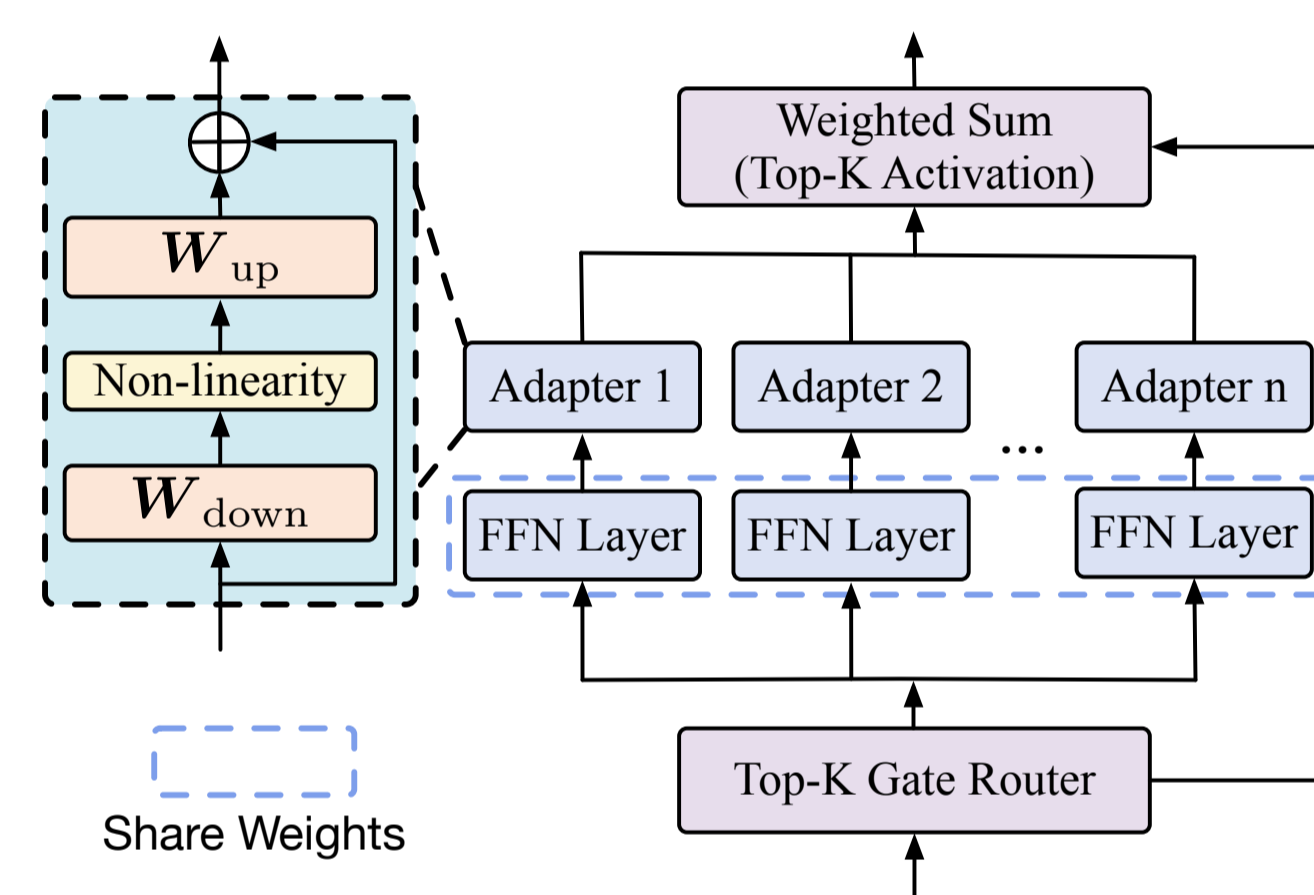


Figure 2. Detailed design of the MoE layer for PESC. All the FFN layers share the same weights.

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

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

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

$$A_i(\vec{x}) = \sigma(\vec{x}\vec{W}_{i,\text{down}})\vec{W}_{i,\text{up}} + \vec{x}. \quad (4)$$

Experiment Settings

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

Table 1. The proportion of various datasets in IDAE datasets.

The training involved integrating three distinct datasets from varied domains during the instruction tuning phase: SlimOrca, Magocoder, and MetaMathQA datasets. After filtration and sampling, we can get two instruction datasets including IDAE-500K and IDAE-720K for instruction tuning.

Camel and Camelidae Models.

1. We fine-tune Camel and Camelidae models using identical datasets, 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.
2. Notably, to further enhance the capabilities of the sparse models, we also utilize IDAE-720K for the instruction-tuning of the Camelidae-pro model.
3. All Camelidae models utilize the top-2 gate router.

Experiment Results

	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 2. 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.

	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 3. Overall performance on all the evaluation benchmarks of dense models (Camel) and sparse (Camelidae) models across 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 4. Evaluation on different numbers of experts in the MoE layers.

Routing Analysis

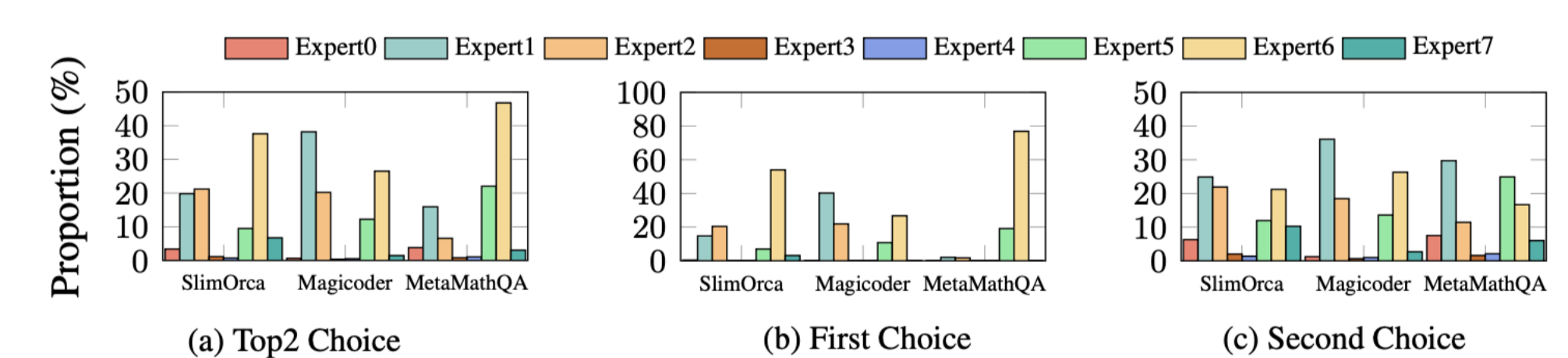


Figure 3. Proportion of tokens assigned to each expert on different dataset subsets.

Conclusion

This paper introduces Parameter-Efficient Sparsity Crafting (PESC) which upcycles dense models into sparse models utilizing the MoE architecture.

1. PESC incorporates adapters within the MoE layers of sparse models, enabling the differentiation of experts without modifying the individual weights of each expert.
2. This technique significantly reduces computational costs and GPU memory requirements compared to sparse upcycling, which facilitates the expansion of model capacity with a minimal parameter.
3. Extensive experiment results on various benchmarks demonstrate the effectiveness of our PESC for instruction tuning on general tasks.