Block-wise Optimization for Lottery-Ticket Adaptation in Large Language Models

Anonymous ACL submission

Abstract

We propose **B**lock-wise **O**ptimization for **L**ottery-Ticket **A**daptation (**BOLA**), a sparse-PEFT framework for efficient adaptation of large language models to new domains. In contrast to parameter-efficient fine-tuning (PEFT) methods such as LoRA and DoRA, which apply dense updates within target modules, BOLA adopts a sparse adapter that updates only a domain-relevant subset of parameters. Motivated by the Lottery Ticket Hypothesis, BOLA identifies and optimizes important parameters in a block-wise sparse manner. Experiments on math and commonsense reasoning benchmarks show that BOLA achieves performance comparable to or better than LoRA and DoRA, while effectively mitigating catastrophic forgetting by updating only a subset of important parameters. Moreover, its block-sparse structure enables more interpretable and memory-efficient adaptation, making BOLA a practical alternative to dense PEFT methods. Our experiment code is available at `https://anonymous.4open.science/r/peft-B728`.

1 Introduction

With the advent of large language models (LLMs), it is no longer necessary to train separate models for individual NLP tasks. Instead, a single general-purpose model can perform a wide range of tasks given appropriate instructions. It is well known that model performance follows empirical scaling laws, where increasing the number of parameters improves quality according to a power-law relationship (Kaplan et al., 2020). However, training such LLMs requires substantial computational resources, imposing significant hardware and energy demands. For example, training an 8B-parameter model typically requires over 128 GB of GPU memory (Suhoi, 2024), posing a significant challenge for adapting such models to specific domains or tasks.

To address this issue, parameter-efficient fine-tuning (PEFT) methods (Houlsby et al., 2019) have

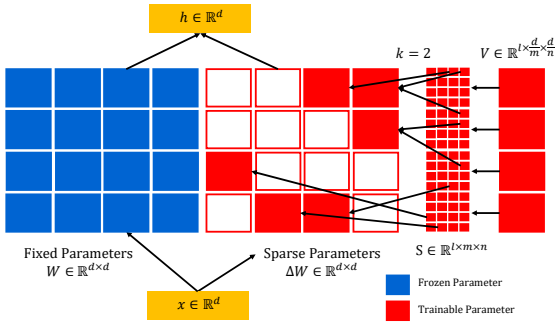

Figure 1: Overview of the proposed BOLA, which constructs trainable sparse weights $\Delta W \in \mathbb{R}^{d \times d}$, where $S \in \mathbb{R}^{l \times m \times n}$ are block-wise score weights and $V \in \mathbb{R}^{l \times d/m \times d/n}$ are block-wise value weights. Blue indicates frozen parameters, red indicates trainable parameters, and white indicates zero parameters.

been proposed to adapt pretrained models by updating only a small subset of parameters. Among these, reparameterized approaches such as LoRA (Hu et al., 2022) and DoRA (Liu et al., 2024a) employ low-rank decomposition, achieving performance comparable to full fine-tuning while significantly reducing memory requirements. Notably, QLoRA (Dettmers et al., 2023) enables fine-tuning of 8B-scale LLMs with as little as 24 GB of GPU memory. These advances have greatly improved the accessibility of large-scale model adaptation. Despite their efficiency, many PEFT methods rely on dense parameter updates within target modules, which can lead to catastrophic forgetting (Ramasesh et al., 2022; Dong et al., 2024; Luo et al., 2025), where pretrained knowledge is partially overwritten during fine-tuning. This can become a major obstacle to developing models tailored to specific tasks or domains, particularly in scenarios requiring continual or multi-domain adaptation.

Recently, sparse-PEFT methods (Panda et al., 2024; Xu and Zhang, 2024; Rios et al., 2025) have

shown strong performance while offering several advantages:

- They update only a subset of parameters, reducing interference with pretrained weights and mitigating catastrophic forgetting.
- They provide improved interpretability by revealing which parameters encode task- or domain-specific knowledge.

However, most existing sparse methods follow a two-stage pipeline, where important parameters are first identified and then fine-tuned, introducing additional computational overhead and implementation complexity.

To overcome these limitations, we propose **B**lock-wise **O**ptimization for **L**ottery-Ticket **A**daptation (**BOLA**), as illustrated in Figure 1. BOLA simultaneously performs parameter selection and optimization in a block-wise sparse manner, inspired by the Lottery Ticket Hypothesis (LTH) (Frankle and Carbin, 2019; Ramanujan et al., 2020). We evaluate BOLA across multiple models and domains, demonstrating its ability to capture domain-specific structures through interpretable score maps. Our results show that BOLA is highly compatible with existing PEFT methods such as LoRA and DoRA, achieving competitive performance despite its sparsity. Moreover, the sparse structure of BOLA reduces interference with pretrained weights, helping mitigate catastrophic forgetting. The main contributions of this paper are as follows:

- We introduce BOLA, a sparse-PEFT method that jointly performs parameter selection and optimization in a block-wise manner.
- We show that BOLA achieves performance comparable to LoRA and DoRA while mitigating catastrophic forgetting.
- We demonstrate that BOLA improves interpretability by identifying blocks responsible for domain-specific knowledge.

2 Related Works

Lottery Ticket Hypothesis (LTH) (Frankle and Carbin, 2019) posits that within a randomly initialized neural network, there exist sparse subnetworks that can achieve accuracy comparable to the original network when trained in isolation. This perspective enables pruning pretrained models to arbitrary sizes. These subnetworks, known as winning tickets, are typically identified by pruning weights based on their magnitude after training, removing connections that are less important for the target task or domain (Frankle and Carbin, 2019; Ramanujan et al., 2020). Several more efficient methods have been proposed to identify winning tickets earlier during training (Chen et al., 2022; Yuan et al., 2025). Methods that jointly perform training and pruning have also been proposed (You et al., 2025).

Parameter-Efficient Fine-Tuning (PEFT) methods (Houlsby et al., 2019) are designed to reduce the computational and memory cost of fine-tuning large language models. Since the number of trainable parameters directly impacts GPU memory consumption during training (Suhoi, 2024), PEFT methods aim to minimize the number of trainable parameters through various techniques. These methods can be divided into several categories (Han et al., 2024), among which reparameterized PEFT is particularly notable for its strong performance and lack of additional inference overhead.

Reparameterized PEFT expresses the fine-tuning update as $W' = W_o + \Delta W$, where only ΔW is learned and stored. This formulation significantly reduces storage requirements compared to full fine-tuning. We further classify reparameterized PEFT into dense-PEFT and sparse-PEFT, depending on whether the trainable update ΔW is represented as a dense or sparse matrix.

Dense-PEFT employs a dense matrix for trainable weights $\Delta W \in \mathbb{R}^{d\times d}$. LoRA (Hu et al., 2022) applies low-rank matrix decomposition to approximate $\Delta W = BA$, where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times d}$. The rank r is much smaller than the dimension d, which leads to a reduction in the number of trainable parameters from d^2 to $2rd$. DoRA (Liu et al., 2024a) decomposes ΔW into its magnitude and direction as $\Delta W = m\Delta W'/\|\Delta W'\|$, thereby enhancing performance by bridging the gap between the learning patterns of full fine-tuning (Full-FT) and LoRA while maintaining a low computational cost. Variants of LoRA (Edalati et al., 2022; Zhang et al., 2023; Kopiczko et al., 2024; Hayou et al., 2024) have also been proposed. Additionally, methods to construct the trainable weights ΔW using Fourier transforms (Liu et al., 2024b; Gao et al., 2024), Kronecker product (Edalati et al., 2023), or specially designed operations have been proposed (Jiang et al., 2024; Tan et al., 2024).

These dense-PEFT methods update all parameters of the target module and lead to catastrophic forgetting (Ramasesh et al., 2022; Dong et al., 2024; Luo et al., 2025). Therefore, it is not suited for continuous learning or model merging that combines models from multiple domains.

Sparse-PEFT uses sparse updates for $\Delta W \in \mathbb{R}^{d\times d}$. Most entries of ΔW are zero, and only selected indices I with corresponding values V are updated. LoTA (Panda et al., 2024) identifies important indices via lottery-ticket scores in a pretraining stage, and has been shown to alleviate forgetting. SpaRTA (Rios et al., 2025) selects indices randomly, but still relies on a separate stage before fine-tuning. These sparse-PEFT methods require computational or design cost to determine which parameters to update before fine-tuning.

Our method simultaneously performs the search and update of trainable parameters, inspired by LTH. Therefore, it does not require a prior stage to identify indices and can be easily applied as a replacement for popular dense-PEFT methods such as LoRA and DoRA. To the best of our knowledge, this is the first PEFT framework that simultaneously searches for and updates target parameters. We validate the performance and effectiveness of our method through comprehensive experiments, and further show that it effectively mitigates knowledge forgetting during fine-tuning and model merging.

3 BOLA: Our Sparse PEFT

Figure 1 illustrates **Bl**ock-wise **L**ottery Ticket **A**daptation (**BOLA**), our proposed sparse-PEFT method. To mitigate the computational burden of identifying important parameters individually, BOLA partitions the weights into several blocks and performs exploration at the block level, as illustrated in Figure 1. BOLA searches for and updates block-wise target parameters, also referred to as winning tickets (Frankle and Carbin, 2019), based on block-wise weight scores. BOLA employs a reparameterized PEFT framework, where the fine-tuned weights $W' \in \mathbb{R}^{d\times d}$ can be represented as:

$$W' = W_o + \Delta W, \tag{1}$$

where $W_o \in \mathbb{R}^{d\times d}$ denotes the frozen pretrained weights, $\Delta W \in \mathbb{R}^{d\times d}$ denotes the trainable block sparse weights, and d denotes the input and output dimension.

3.1 Block-wise Optimization

As shown in Figure 1, the block sparse weights ΔW are structured such that nonzero elements are grouped into dense blocks, while the remaining regions are zero. To construct such block-sparse weights, both a value matrix V and a block assignment matrix T are required. The block assignment matrix T specifies the target block locations for the block-wise values in V. In this formulation, the sparse weights are constructed at the block level, and the trainable sparse weights $\Delta W \in \mathbb{R}^{d\times d}$ are defined as:

$$\Delta W = \sum_l T_l \otimes V_l, \tag{2}$$

where l denotes the number of trainable blocks, and m and n specify the number of divisions along the output and input dimensions, respectively. Here, each $T_l \in \mathbb{R}^{m\times n}$ is a block assignment matrix that selects target blocks, and each $V_l \in \mathbb{R}^{d/m\times d/n}$ represents the corresponding block values. $\otimes$ denotes the Kronecker product, which places each block in V_l at the corresponding location specified by T_l, forming a block-sparse weight matrix. T is a soft assignment matrix that represents relaxed block selection, rather than a hard binary mask. BOLA introduces two trainable parameters: a score matrix $S \in \mathbb{R}^{l\times m\times n}$ and a value matrix $V \in \mathbb{R}^{l\times d/m\times d/n}$. The block assignment matrix T is derived from S as follows:

$$M_l = \mathbb{I}\big((i,j) \in \mathcal{T}(S_l)\big), \tag{3}$$

$$\tilde{S} = S \odot M, \tag{4}$$

$$T = S + \mathrm{stopgrad}(\tilde{S} - S), \tag{5}$$

where $\mathcal{T}$ denotes a selection operator (e.g., top-k, top-p, or sampling), $\mathbb{I}$ is an indicator function, $\odot$ denotes the element-wise product, and $\mathrm{stopgrad}$ prevents gradient propagation. The score matrix S determines the selected block indices, while the value matrix V provides the corresponding block parameters. The same block may be selected multiple times, in which case the corresponding values are accumulated. The parameters S are initialized using a uniform Kaiming initialization (He et al., 2015), while V is initialized to zero. This ensures that ΔW starts from zero, similar to LoRA-based methods (Hu et al., 2022; Edalati et al., 2022; Zhang et al., 2023; Kopiczko et al., 2024; Hayou et al., 2024; Liu et al., 2024a), which is important for stable training (Liao et al., 2023). Since only

the selected blocks are used in constructing ΔW during the forward pass, gradients are not propagated to score parameters associated with unselected blocks in Equation (4).

To address this, we adopt a straight-through estimator (STE) (Bengio et al., 2013), as shown in Equation (5), enabling gradient propagation through all score parameters. This design is inspired by the Edge-popup algorithm (Ramanujan et al., 2020) and allows dynamic block selection during training. The total number of trainable parameters is given by $\phi = l\,(mn + dd/mn)$, which accounts for both S and V. Setting $mn = d$ simplifies this to $\phi = 2ld$. This formulation provides a drop-in alternative to LoRA, making BOLA a compatible and efficient method for parameter-efficient fine-tuning.

3.2 Block-wise Sinkhorn Optimization

In the block-wise selection process, a block may be selected multiple times, in which case its contributions are accumulated. While this accumulation increases expressiveness, it can also introduce redundancy and lead to inefficient utilization of available blocks. To mitigate this issue, we encourage diversity in block selection by reducing duplicate assignments across the $m \times n$ blocks.

In particular, we impose a constraint that promotes near-orthogonal (i.e., one-to-one) block assignments by normalizing the score matrix S into a doubly stochastic form. Prior to Equation (3), we apply the Sinkhorn-Knopp operator to S, which approximates a soft permutation over block indices. This is achieved by enforcing both row-wise and column-wise normalization, which encourages near one-to-one assignments between candidate blocks and target positions while discouraging repeated selection of the same block. This procedure is implemented using the Sinkhorn-Knopp operator, given by

$$S = \text{Sinkhorn-Knopp}(S), \tag{6}$$

where the operator first enforces positivity via an element-wise exponential transformation and then performs an iterative normalization that alternately rescales rows and columns so that each sums to one, yielding a doubly stochastic matrix.

4 Experiments

We conduct a series of experiments to evaluate the effectiveness of our method across various language models and tasks. Specifically, we use RoBERTa as a small-scale model and LLaMA3 as a large-scale model. For evaluation, we consider datasets from the GLUE benchmark, along with benchmarks targeting mathematical and commonsense reasoning. We adopt LoRA (Hu et al., 2022), DoRA (Liu et al., 2024a), and SpaRTA (Rios et al., 2025) as strong baselines. To ensure a fair comparison, we set hyperparameters such that the number of trainable parameters matches those of LoRA and SpaRTA. However, in DoRA, the number of trainable parameters increases by an additional d due to the magnitude component. Furthermore, we keep hyperparameters consistent across methods unless explicitly stated otherwise to ensure reproducibility (see Appendix A). For model selection, we choose the model that achieves the best validation performance among five different learning rates. All experiments are conducted on NVIDIA H100 80GB GPUs. We use four GPUs for RoBERTa-125M and eight GPUs for LLaMA3-8B. RoBERTa-125M is trained in float32 precision due to its smaller size, while LLaMA3-8B is trained in bfloat16 precision to improve efficiency on H100 GPUs. Implementation is based on PyTorch, Hugging Face Transformers (Wolf et al., 2020), and the PEFT (Mangrulkar et al., 2022) library. In BOLA, we adopt top-p sampling for the selection operator $\mathcal{T}$, and use ten iterations for the Sinkhorn variant.

4.1 GLUE Benchmark

We evaluate BOLA in comparison with LoRA, DoRA, and SpaRTA using the relatively small language model RoBERTa-125M (Liu et al., 2019) on the GLUE benchmark (Wang et al., 2018). Although GLUE consists of eight downstream tasks, for consistency with previous work (Hu et al., 2022), we report the results on all tasks as well as their average, computed over three random seeds. We also report the number of trainable parameters, excluding the classification head, to ensure a fair comparison of adaptation modules.

The results are presented in Table 1. For each run, we report the best-performing epoch. BOLA achieves strong performance across all tasks, outperforming the baselines on SST2 and QNLI. Compared with same sparse-PEFT SpaRTA, BOLA consistently achieves higher performance, with an average improvement of 0.5 points. When applying the Sinkhorn algorithm, BOLA further improves performance by 0.2 points on average, resulting in performance that is competitive with dense-PEFT

Method	#Params.	MNL	SST2	MRPC	CoLA	QNLI	QQP	RTE	SST-B	Avg.
Full-FT	125M	86.3$_{\pm 0.0}$	93.4$_{\pm 0.3}$	**89.9**$_{\pm 0.4}$	63.8$_{\pm 0.6}$	92.2$_{\pm 0.1}$	**91.6**$_{\pm 0.1}$	**79.5**$_{\pm 1.2}$	90.0$_{\pm 0.1}$	**85.9**$_{\pm 0.3}$
LoRA	0.3M	**87.5**$_{\pm 0.0}$	95.0$_{\pm 0.1}$	89.2$_{\pm 0.2}$	63.1$_{\pm 0.4}$	92.7$_{\pm 0.0}$	90.9$_{\pm 0.1}$	77.1$_{\pm 1.3}$	90.9$_{\pm 0.0}$	85.8$_{\pm 0.2}$
DoRA	0.3M	84.3$_{\pm 0.2}$	94.2$_{\pm 0.1}$	89.3$_{\pm 0.6}$	**64.6**$_{\pm 0.5}$	91.6$_{\pm 0.3}$	87.7$_{\pm 0.0}$	79.1$_{\pm 1.4}$	91.0$_{\pm 0.1}$	85.2$_{\pm 0.1}$
SpaRTA	0.3M	85.1$_{\pm 0.1}$	94.8$_{\pm 0.1}$	88.8$_{\pm 0.8}$	60.2$_{\pm 1.4}$	92.9$_{\pm 0.1}$	90.0$_{\pm 0.0}$	76.2$_{\pm 1.1}$	90.5$_{\pm 0.1}$	85.0$_{\pm 0.1}$
BOLA	0.3M	87.0$_{\pm 0.0}$	**95.1**$_{\pm 0.2}$	88.2$_{\pm 0.5}$	63.3$_{\pm 0.9}$	**92.8**$_{\pm 0.2}$	90.0$_{\pm 0.0}$	77.1$_{\pm 0.2}$	90.8$_{\pm 0.2}$	85.5$_{\pm 0.1}$
BOLA†	0.3M	87.1$_{\pm 0.1}$	94.8$_{\pm 0.2}$	88.9$_{\pm 0.8}$	63.1$_{\pm 1.6}$	92.7$_{\pm 0.3}$	90.6$_{\pm 0.3}$	76.9$_{\pm 2.2}$	**91.1**$_{\pm 0.6}$	85.7$_{\pm 0.2}$

Table 1: RoBERTa-125M with different adaptation methods on the GLUE benchmark. #Params. indicates the number of trainable parameters excluding the classification head module. We report the overall (matched and mismatched) accuracy for MNLI, Matthews correlation for CoLA, Pearson correlation for STS-B, and accuracy for the other tasks. For all metrics, higher values indicate better performance. The highest values are indicated in bold. † indicates the BOLA variant using the Sinkhorn algorithm.

Method	#Params.	MultiArith	GSM8K	AddSub	AQuA	SingleEq	SVAMP	Avg.
LoRA	56.6M	89.3$_{\pm 2.3}$	53.6$_{\pm 1.5}$	82.0$_{\pm 1.4}$	27.8$_{\pm 3.5}$	83.7$_{\pm 1.1}$	**65.6**$_{\pm 2.2}$	67.0$_{\pm 0.8}$
DoRA	57.4M	**89.9**$_{\pm 2.3}$	**56.0**$_{\pm 1.8}$	78.7$_{\pm 1.2}$	24.7$_{\pm 0.9}$	84.6$_{\pm 1.0}$	63.9$_{\pm 2.5}$	66.2$_{\pm 1.3}$
SpaRTA	56.6M	44.4$_{\pm 5.8}$	35.9$_{\pm 4.2}$	79.2$_{\pm 2.6}$	29.1$_{\pm 2.3}$	72.8$_{\pm 2.2}$	51.8$_{\pm 2.3}$	52.2$_{\pm 2.9}$
BOLA	56.6M	83.2$_{\pm 2.1}$	54.6$_{\pm 1.8}$	81.8$_{\pm 1.1}$	33.1$_{\pm 2.8}$	84.3$_{\pm 1.0}$	64.2$_{\pm 2.0}$	66.8$_{\pm 0.8}$
BOLA†	56.6M	83.7$_{\pm 2.3}$	55.5$_{\pm 1.4}$	**83.3**$_{\pm 1.2}$	**33.9**$_{\pm 2.1}$	**84.8**$_{\pm 0.9}$	65.0$_{\pm 2.0}$	**67.7**$_{\pm 1.0}$

Table 2: Llama3-8B with different adaptation methods on the math benchmark. #Params. indicates the number of trainable parameters. We report the accuracy on each dataset and its higher value is better for all metrics. The highest values are indicated in bold. † indicates the BOLA variant using the Sinkhorn algorithm.

Method	#Params.	BoolQ	PIQA	SIQA	HS	WG	ARC-e	ARC-c	OBQA	Avg.
LoRA	56.6M	74.6$_{\pm 0.2}$	88.4$_{\pm 0.4}$	80.0$_{\pm 0.5}$	93.3$_{\pm 0.7}$	85.3$_{\pm 0.7}$	90.1$_{\pm 0.3}$	79.3$_{\pm 0.4}$	84.9$_{\pm 0.4}$	84.5$_{\pm 0.1}$
DoRA	57.4M	74.9$_{\pm 0.4}$	88.4$_{\pm 0.5}$	79.8$_{\pm 0.5}$	93.2$_{\pm 0.9}$	85.1$_{\pm 0.7}$	90.0$_{\pm 0.0}$	79.0$_{\pm 0.1}$	**85.5**$_{\pm 1.1}$	84.4$_{\pm 0.2}$
SpaRTA	56.6M	73.9$_{\pm 0.6}$	86.8$_{\pm 0.6}$	79.0$_{\pm 0.4}$	91.6$_{\pm 0.8}$	84.6$_{\pm 1.1}$	88.6$_{\pm 0.6}$	76.4$_{\pm 0.3}$	83.8$_{\pm 0.8}$	83.1$_{\pm 0.1}$
BOLA	56.6M	75.2$_{\pm 0.2}$	88.6$_{\pm 0.7}$	80.4$_{\pm 0.1}$	93.1$_{\pm 0.1}$	**85.4**$_{\pm 0.3}$	90.6$_{\pm 0.2}$	79.9$_{\pm 0.7}$	85.1$_{\pm 0.1}$	84.9$_{\pm 0.1}$
BOLA†	56.6M	**75.6**$_{\pm 0.3}$	**89.0**$_{\pm 0.8}$	**80.6**$_{\pm 0.2}$	**93.5**$_{\pm 0.4}$	85.3$_{\pm 0.5}$	**90.7**$_{\pm 0.1}$	**80.4**$_{\pm 0.4}$	85.2$_{\pm 0.2}$	**85.0**$_{\pm 0.2}$

Table 3: Llama3-8B with different adaptation methods on the commonsense benchmark. #Params. indicates the number of trainable parameters. We report the accuracy on each dataset and its higher value is better for all metrics. The highest values are indicated in bold. Note that HS and WG are abbreviations for HellaSwag and WinoGrande, respectively. † indicates the BOLA variant using the Sinkhorn algorithm.

methods such as LoRA.

4.2 Math and Commonsense Benchmark

We evaluate BOLA in comparison with LoRA, DoRA, and SpaRTA using the LLM LLaMA3-8B (Llama-Team, 2024) and benchmarks for math and commonsense reasoning. The math and commonsense benchmarks consist of eight and six tasks, respectively, each with predefined training and testing sets. Following the experimental setup of (Hu et al., 2023), we merge the training data from all eight commonsense tasks to construct a unified training dataset and evaluate on the corresponding test set for each task. We use the datasets[1] `math10K.json` and `commonsense170K.json` to fine-tune the models. We use top-p sampling with $p = 0.5$.

The results of the math benchmark are shown in Table 2. BOLA demonstrates improved performance over the baselines on the AddSub, AQuA, and SingleEq tasks. On average, BOLA achieves performance comparable to LoRA and higher than the sparse-PEFT method SpaRTA. BOLA further improves its average performance by an additional 0.9 points when applying the Sinkhorn algorithm, thereby establishing a new best overall result. The results of the commonsense benchmark are shown in Table 3. BOLA demonstrates improved performance over the baselines on the all tasks without OBQA task. On average, BOLA achieves higher performance than LoRA, DoRA

[1]https://github.com/AGI-Edgerunners/LLM-Adapters

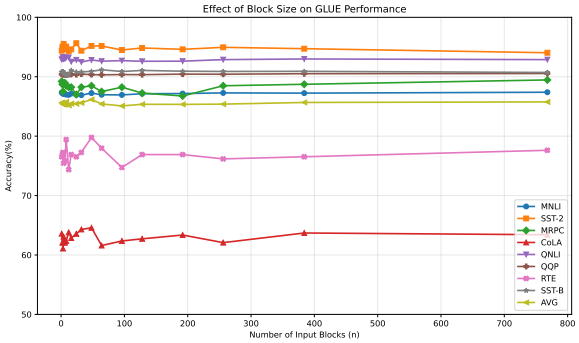

Figure 2: Evaluation results for various block sizes for RoBERTa-125M on GLUE tasks. The horizontal axis corresponds to size n, which is varied from 1 to $d = 768$ under the constraint that $mn = d = 768$ remains constant. Each marker represents the evaluation result for a downstream task included in the GLUE benchmark.

and SpaRTA. BOLA further improves its average performance by an additional 0.6 points when applying the Sinkhorn algorithm, thereby establishing a new best overall result.

Across all benchmarks and model configurations, BOLA consistently outperforms another sparse-PEFT method SpaRTA, and achieves the best performance when combined with the Sinkhorn algorithm. As discussed in Section 3.2, this result suggests that reducing overlap among selected blocks improves learning efficiency. However, incorporating the Sinkhorn algorithm increases the training time in the math domain by approximately 1.3 times, highlighting a trade-off between improved performance and additional computational cost.

5 Ablation Study

In this section, we introduce the impact of hyperparameter in BOLA.

5.1 Structured vs. Unstructured

The hyperparameters m and n determine the number of block divisions along the input and output dimensions, thereby defining the shape of the trainable block weights. Figure 2 presents the performance trends as n varies under the constraint $mn = d$. In this study, we define configurations as "structured" when either $m = d$ or $n = d$, such that the block structure corresponds to a full-row or full-column form, where each block spans the entire input or output dimension along one axis, while all other configurations are regarded as "unstructured". Structured configurations are more amenable to efficient GPU implementation due to regular memory access patterns and better compatibility with dense matrix multiplication kernels.

As shown in Figure 2, we evaluate the impact of block weight shapes on the performance of RoBERTa-125M using the GLUE benchmark. Although slight fluctuations are observed across different block sizes, the performance remains largely stable across all tasks. These results indicate that the choice of block structure does not critically affect performance and does not require careful tuning. This robustness simplifies practical deployment, as performance is largely insensitive to the specific block configuration. Instead, it is more effective to focus on tuning the number of base blocks l and the scaling factor α. This observation aligns with prior dense-PEFT methods such as LoRA and DoRA, where performance is primarily governed by the rank r and scaling factor α rather than the specific parameterization structure.

6 Analysis of Sparse Updates

In this section, we present two complementary benefits of sparse-PEFT compared with dense-PEFT, namely enhanced interpretability and mitigated performance degradation during model merging.

6.1 Interpretability of Model Structure

The sparsity of BOLA improves interpretability by explicitly revealing which parameters are responsible for domain-specific knowledge. Figure 3 illustrates the trained weights ΔW of LoRA and BOLA in LLaMA3-8B (Llama-Team, 2024), using math and commonsense reasoning datasets. While LoRA updates all parameters, BOLA selectively identifies and updates only the most important ones. As a result, BOLA induces a structured separation between active and inactive parameters, where inactive weights remain zero.

BOLA partitions the weights into $m \times n$ blocks along the input and output dimensions, and updates only the blocks with high scores. Since the base model is not specialized for the reasoning domain and we employ top-p sampling, the updates are distributed more broadly across selected regions to compensate for the lack of prior domain-specific knowledge. Notably, in BOLA, there exist input dimensions that are not explicitly utilized in either the mathematical or commonsense domains (around input dimensions 1700 and 2700), suggesting the presence of unused or redundant representational capacity. In contrast, regions that are jointly uti-

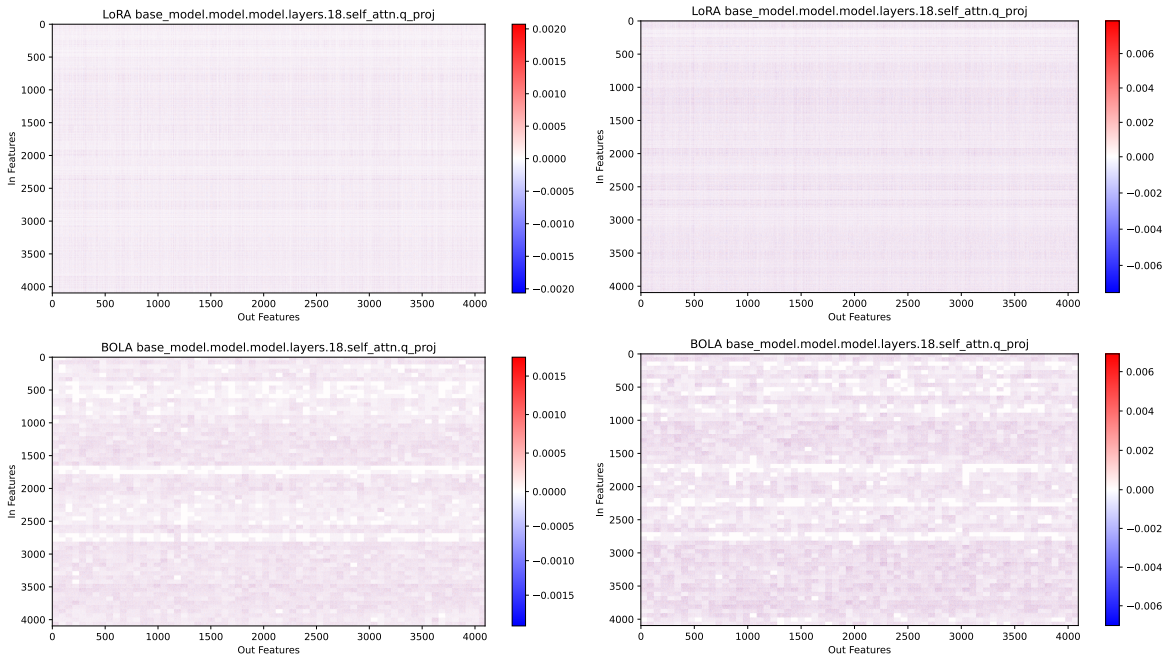

Figure 3: Visualization of the trained weights ΔW in the tenth layer of Llama3-8B. The top panel shows the weights obtained with LoRA, while the bottom panel shows those obtained with BOLA. The left and right panels correspond to models trained on mathematical and commonsense reasoning data, respectively. Color intensity represents the magnitude of the weights.

lized by both domains suggest the existence of shared representational capacity that supports general reasoning abilities. We further observe dimensions that are utilized exclusively for the mathematical domain (around input dimension 2300), indicating domain-specific specialization in the input space. Such structured and interpretable patterns are not readily observable in LoRA, where updates are densely distributed across all parameters. These observations suggest that BOLA implicitly learns a structured allocation of representational capacity, where different regions of the parameter space are specialized for different domains. This structured sparsity provides insights into how knowledge is organized within the model, enabling more transparent analysis of domain adaptation behavior.

As shown in Figure 3, although LoRA has access to all parameters, this suggests that effective domain adaptation does not require uniform updates across all parameters. This observation is consistent with prior work suggesting that certain parameters play specialized roles (Yu et al., 2025).

6.2 Sparse Model Merge

One notable property of sparse-PEFT is the reduced overlap between weights trained on different domains, which contributes to reduced interference during model merging. LoTA (Panda et al., 2024) and related sparse-PEFT methods have been shown to be robust to knowledge forgetting during model merging, leveraging this property of sparse updates. Catastrophic forgetting (Ramasesh et al., 2022; Dong et al., 2024; Luo et al., 2025) refers to the degradation in performance on previously learned tasks that occurs when LLMs are fine-tuned sequentially on multiple domains or tasks.

Figure 4 shows that the sparse-PEFT methods SpaRTA and BOLA mitigate catastrophic forgetting during model merging. We perform linear model merging (Wortsman et al., 2022) by merging only the parameter updates ΔW (with seed=0) of models trained in Section 4.2. For BOLA, we employ the variant without the Sinkhorn algorithm to isolate the effect of sparsity. In the math benchmark, LoRA exhibits significantly degraded performance across all datasets when merging weights trained on math with those trained on commonsense, resulting in an average drop of 17.7 points. Notably, MultiArith, GSM8K, and AQuA experience substantial performance degradation. In contrast, the sparse-PEFT methods SpaRTA and BOLA exhibit average performance reductions of only 5.4 and 6.4 points after merging, demonstrating their robustness to catastrophic forgetting. This behavior is consistent with the reduced overlap

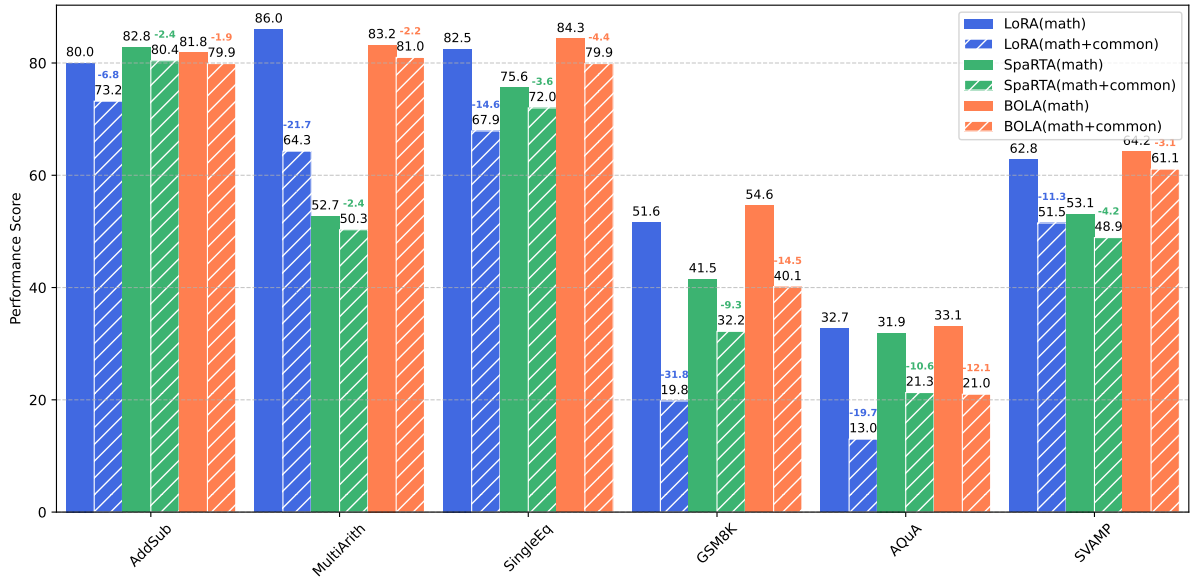

Figure 4: Performance comparison of model merging on the math benchmark. Bar plots show the performance of math domain LoRA, SpaRTA, and BOLA models (filled) and their counterparts after merging with commonsense domain models (hatched). The black numbers above each bar denote the absolute performance values, whereas the colored numbers indicate the performance differences induced by model merging.

between domain-specific parameters, which alleviates interference during model merging. These results suggest that sparse parameterization naturally enables modular composition of domain-specific knowledge, as independently trained components can be combined with minimal conflict. In particular, sparse-PEFT methods such as LoTA and BOLA are well-suited for continual learning and multi-domain model merging settings, where maintaining previously acquired knowledge is essential.

Sparse-PEFT models trained independently on the math and commonsense domains can therefore be merged with limited performance degradation. This sparsity enables efficient and modular fine-tuning, making sparse-PEFT a practical approach for developing domain-specialized models.

7 Conclusion

This work introduces **B**lock-wise **O**ptimization for **L**ottery Ticket **A**daptation (**BOLA**), a block-wise sparse parameter-efficient fine-tuning method that jointly performs parameter selection and optimization in a single stage. Unlike prior sparse-PEFT methods that rely on separate pruning or selection procedures before fine-tuning, BOLA learns importance scores and task-specific updates in an end-to-end manner during training.

BOLA induces adaptive sparsity that dynamically identifies task-relevant subnetworks throughout optimization, operating on block-structured parameters with learnable scores. This enables efficient adaptation while maintaining competitive performance with dense-PEFT methods such as LoRA and DoRA across various tasks and model scales. Moreover, its updates can be merged back into the pretrained model without additional inference overhead.

In addition, the resulting block-wise sparsity improves interpretability of learned representations and reduces interference between tasks, leading to more robust model merging with alleviated catastrophic forgetting. Finally, we plan to explore the generalization of BOLA to other modalities, including audio and vision.

Limitations

Our PEFT approach demonstrates the potential to alleviate catastrophic forgetting during model merging. However, when the characteristics of the target domains differ significantly, there is no guarantee that the trained parameters remain mutually exclusive. As a future direction, we plan to extend this method to enable users to explore and update critical parameters within user-specified regions, thereby providing more effective control over parameter redundancy.

References

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *Preprint*, arXiv:1308.3432.

Xiaohan Chen, Jason Zhang, and Zhangyang Wang. 2022. Peek-a-boo: What (more) is disguised in a randomly weighted neural network, and how to find it efficiently. In *International Conference on Learning Representations*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2024. How abilities in large language models are affected by supervised fine-tuning data composition. *Preprint*, arXiv:2310.05492.

Ali Edalati, Marzieh Tahaei, Ivan Kobyzev, Vahid Partovi Nia, James J. Clark, and Mehdi Rezagholizadeh. 2022. Krona: Parameter efficient tuning with kronecker adapter. *Preprint*, arXiv:2212.10650.

Ali Edalati, Marzieh Tahaei, Ivan Kobyzev, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. 2023. Krona: Parameter efficient tuning with kronecker adapter. In *Third Workshop on Efficient Natural Language and Speech Processing (ENLSP-III)*.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *Preprint*, arXiv:1803.03635.

Ziqi Gao, Qichao Wang, Aochuan Chen, Zijing Liu, Bingzhe Wu, Liang Chen, and Jia Li. 2024. Parameter-efficient fine-tuning with discrete fourier transform. *Preprint*, arXiv:2405.03003.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *Preprint*, arXiv:2403.14608.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Preprint*, arXiv:1502.01852.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. 2023. LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5254–5276, Singapore. Association for Computational Linguistics.

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

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

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

Baohao Liao, Shaomu Tan, and Christof Monz. 2023. Make your pre-trained model reversible: From parameter to memory efficient fine-tuning. *Preprint*, arXiv:2306.00477.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024a. DoRA: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*.

Weiyang Liu, Zeju Qiu, Yao Feng, Yuliang Xiu, Yuxuan Xue, Longhui Yu, Haiwen Feng, Zhen Liu, Juyeon Heo, Songyou Peng, Yandong Wen, Michael J. Black, Adrian Weller, and Bernhard Schölkopf. 2024b. Parameter-efficient orthogonal finetuning via butterfly factorization. In *The Twelfth International Conference on Learning Representations*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *Preprint*, arXiv:1907.11692.

Llama-Team. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2025. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *Preprint*, arXiv:2308.08747.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft.

Ashwinee Panda, Berivan Isik, Xiangyu Qi, Sanmi Koyejo, Tsachy Weissman, and Prateek Mittal. 2024. Lottery ticket adaptation: Mitigating destructive interference in llms. *Preprint*, arXiv:2406.16797.

Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. 2020. What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Vinay Venkatesh Ramasesh, Aitor Lewkowycz, and Ethan Dyer. 2022. Effect of scale on catastrophic forgetting in neural networks. In *International Conference on Learning Representations*.

Jesus Rios, Pierre Dognin, Ronny Luss, and Karthikeyan N. Ramamurthy. 2025. Sparsity may be all you need: Sparse random parameter adaptation. *Preprint*, arXiv:2502.15975.

Daniil Suhoi. 2024. Efficient deep learning: A comprehensive overview of optimization techniques. Hugging Face Blog.

Wenxuan Tan, Nicholas Roberts, Tzu-Heng Huang, Jitian Zhao, John Cooper, Samuel Guo, Chengyu Duan, and Frederic Sala. 2024. More fine-tuning with 10x fewer parameters. In *ICML 2024 Workshop on Foundation Models in the Wild*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. *Preprint*, arXiv:2203.05482.

Jing Xu and Jingzhao Zhang. 2024. Random masking finds winning tickets for parameter efficient fine-tuning. *Preprint*, arXiv:2405.02596.

Haoran You, Baopu Li, Zhanyi Sun, Xu Ouyang, and Yingyan Celine Lin. 2025. Supertickets: Drawing task-agnostic lottery tickets from supernets via jointly architecture searching and parameter pruning. *Preprint*, arXiv:2207.03677.

Mengxia Yu, De Wang, Qi Shan, Colorado J Reed, and Alvin Wan. 2025. The super weight in large language models. *Preprint*, arXiv:2411.07191.

Fei Yuan, Chang Ma, Shuai Yuan, Qiushi Sun, and Lei Li. 2025. KS-lottery: Finding certified lottery tickets for multilingual transfer in large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 9077–9090, Albuquerque, New Mexico. Association for Computational Linguistics.

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

A Hyperparameters

For all experiments, we select the learning rate that achieves the highest validation performance from the range $\{1e{-}3, 5e{-}4, 1e{-}4, 5e{-}5, 1e{-}5\}$.

A.1 Configuration for the GLUE Benchmark

Parameter	Value
Epochs	30
Batch Size	64
Sequence Length	512
Optimizer	AdamW
Initial Learning Rate	1e-3
LR Scheduler	Linear
Warmup Ratio	0.06
Weight Decay	0.1
Dropout	0.1
Target Modules	Q, K

Table 4: Common training configuration for the GLUE benchmark.

Parameter	LoRA	DoRA	SpaRTA	BOLA
r	8	8	–	–
(m, n)	–	–	–	(24, 32)
l	–	–	12,288	8
α	16	16	4	2

Table 5: Method-specific hyperparameters for LoRA, DoRA, SpaRTA, and BOLA on the GLUE benchmark.

A.2 Configuration for the Math and Commonsense Benchmarks

Parameter	Value
Epochs	5
Batch Size	128
Sequence Length	512
Optimizer	AdamW
Initial Learning Rate	1e-4
LR Scheduler	Linear
Warmup Ratio	0.06
Weight Decay	0.1
Dropout	0.1
Target Modules	Q, K, V, U, D

Table 6: Common training configuration for the math and commonsense benchmarks.

Parameter	LoRA	DoRA	SpaRTA	BOLA
r	32	32	–	–
(m, n)	–	–	–	(64, 64)
l	–	–	353,894	32
α	64	64	4	2

Table 7: Method-specific hyperparameters for LoRA, DoRA, SpaRTA, and BOLA on the math and commonsense benchmarks.