Proceedings of the 4th International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/2026.37399

Parameter-Efficient Fine-Tuning of BERT: A Study of Fixed-Rank and Layer-Wise LoRA on SST-2

Zijia Liu

Department of Computer Science, Ball State University, Muncie, USA
zijia.liu@bsu.edu

Abstract. Parameter-efficient fine-tuning (PEFT) has emerged as a crucial method for adapting pretrained Transformer models to downstream tasks without the need to fine-tune the full set of model parameters. Low-Rank Adaptation (LoRA) is a technique that reduces the number of trainable parameters by representing weight updates with low-rank matrices. This paper explores the impact of LoRA rank selection and layer-wise rank allocation on the trade-off between classification accuracy and the number of parameters to be fine-tuned through fine-tuning BERT-base-uncased on the sentiment classification task, SST-2. The three fixed-rank LoRA settings with ranks of 4, 8, and 16 are first investigated, as well as a manually-designed layer-wise LoRA setting that assigns different ranks to different subsets of BERT layers. In the fixed-rank experiments, increasing the rank is beneficial to the validation accuracy, but the magnitude of the improvement seems to diminish with the number of trainable parameters. The layer-wise setup is able to attain 92.20% validation accuracy with 345,602 trainable parameters. It decreases the number of trainable parameters by approximately 41.6% and improves the validation accuracy by 0.57 percentage point, compared with LoRA at r = 16. It also can achieves performance close to full fine-tuning with 109,483,778 trainable parameters and 92.32% accuracy. The results indicate that with a non-uniform rank allocation it is possible to achieve an effective compromise between the efficiency of the parameters and the goodness of the downstream classification.

Keywords: parameter-efficient fine-tuning, LoRA, BERT, layer-wise rank allocation, sentiment classification

1. Introduction

In recent years, a general-purpose model called the Transformer has been introduced for natural language processing based on self-attention mechanism is used to learn context [1]. BERT has also demonstrated the benefits of the Transformer pretraining paradigm by pretraining BERT on a large amount of unlabeled text and fine-tuning it for downstream language understanding tasks [2]. For each target task, the traditional full fine-tuning updates all the parameters of a pretrained model. Large models are relatively expensive in terms of computational cost, memory and storage requirements. The above deficiencies can be more pronounced when the same pretrained model is used for more than one downstream task, and different downstream tasks may require different parameters to be updated.

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.37399

To minimize the cost of downstream adaptation, parameter-efficient fine-tuning (PEFT) methods have been proposed to retain most of the knowledge in pretrained models. Adapter-based approaches append small trainable units on top of a fixed backbone and only fine-tune the new units [3]. Prefix Tuning focuses on keeping the pretrained language model unchanged, and learning continuous task-specific vectors, which adjust the language model using the attention mechanism [4]. Low-Rank Adaptation (LoRA) adapts the pretrained weights by adding trainable low-rank matrices for the specific task without modifying the original pretrained weights [5]. This approach decreases the number of trainable parameters as well as the amount of memory required for optimization. In addition, the learned update can be incorporated into the original weight matrix after training and the adapted model will be the same as the original model in the structure of computation during the inference.

Recently, some limitations in standard LoRA have been identified. A fixed-parameter budget may not be suitable for distributing the adaptation capacity of different weight matrices. DyLoRA solves the problem of selecting a fixed rank by using multiple rank settings without repeating the training for each rank [6]. AdaLoRA addresses this problem by allocating the available parameter budget according to the importance scores and reducing the capacity of less important updates [7]. QLoRA reduces the memory requirement for fine-tuning large language models by combining low-rank adaptation with a frozen four-bit quantized backbone [8]. DoRA decomposes the pretrained weights into magnitude and direction components, and then optimizes magnitude and direction adaptation separately [9]. Based on the above studies, parameter-efficient fine-tuning is not solely determined by the total number of trainable parameters but also by how adaptation capability is expressed and distributed.

This paper studies rank allocation in LoRA-based fine-tuning for BERT sentiment classification. Standard LoRA generally uses a fixed rank for the selected projection matrices in all Transformer layers. This design assumes that the same adaptation capability is suitable for all the selected layers. The current study tests this assumption by first evaluating fixed ranks of 4, 8 and 16, and then, using a layer-wise rank allocation strategy, assigns different ranks to different groups of BERT layers. Full fine-tuning is taken as the performance reference. Validation accuracy and the number of trainable parameters are used to determine whether a more selective distribution of low-rank capacity can improve parameter efficiency without a significant drop in classification performance.

The rest of this paper is organized as follows. Section 2 introduces the Transformer architecture, BERT and the mathematical formulation of standard LoRA. Section 3 investigates improvements in LoRA rank selection and parameter allocation, introduces the layer-wise rank allocation strategy, and reports experimental results for the SST-2 sentiment classification task. Section 4 concludes the paper by summarizing the main findings, discussing the limitations, and outlining future research directions.

2. Background and standard LoRA

2.1. Transformer architecture

The Transformer replaces recurrent computation with attention-based representation learning and enables token representations within a sequence to be processed in parallel [1]. Its central component is the self-attention mechanism, which allows the representation of each token to incorporate information from other positions in the same sequence. This mechanism provides an effective way to model contextual dependencies without relying on recurrent connections. The

DOI: 10.54254/2755-2721/2026.37399

Transformer therefore provides the architectural basis for many pretrained language models, including BERT.

Let the input representation of a sequence be denoted by $X \in \mathbb{R}^{n \times d}$, where n represents the sequence length and d represents the hidden dimension. Self-attention first maps the input into three different representations called query, key, and value. These representations are obtained through three trainable linear transformations:

$$Q = XW_Q \tag{1}$$

$$K = XW_K$$

$$V = XW_V$$

where W_Q, W_K,and W_V denote the corresponding projection matrices. Scaled dot-product attention is given by:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2}$$

Here, d_k denotes the dimensionality of the key vectors [1]. The product QK^T represents the similarity between the query of one token and the keys associated with other tokens. The scaling term d_k controls the magnitude of these dot products before the softmax operation. Softmax converts the resulting scores into normalized attention weights. These weights determine how strongly the corresponding value vectors contribute to the contextual representation of each token.

The Transformer further employs self-attention through multi-head attention. Different attention heads use separate learned projections and can therefore capture different relationships in the input sequence. For the ith attention head,

$$head_i = Attention(Q_i, K_i, V_i) \tag{3}$$

The outputs produced by the individual attention heads are concatenated and transformed through an output projection:

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)W_O \tag{4}$$

where W_O denotes the output projection matrix [1]. Each Transformer encoder layer further includes a feed-forward sublayer, residual connections, and normalization. The repeated application of these components allows the model to construct increasingly contextual representations of the input sequence.

BERT uses the Transformer encoder and applies bidirectional pretraining to obtain contextual representations [2]. BERT-base consists of 12 Transformer encoder layers, with a hidden size of 768 and 12 self-attention heads. The pretrained representations can be adapted to classification and other language understanding tasks through the addition of a task-specific output layer [2]. The present study uses BERT-base-uncased as the pretrained backbone for sentiment classification. The self-attention projection matrices are particularly relevant because the query and value transformations are selected as the LoRA adaptation targets in the experimental implementation.

DOI: 10.54254/2755-2721/2026.37399

2.2. Low rank adaptation

In full fine-tuning, all parameters of the pretrained model are updated for the target downstream task. Let $W_o \in \mathbb{R}^{d\times k}$ denote a pretrained weight matrix. The adapted matrix can be expressed as:

$$W' = W_0 + \Delta W \tag{5}$$

where ΔW is the task specific update. In full fine-tuning, this update is unconstrained, and has the same dimensionality as the original weight matrix. Therefore, a dense update matrix will have dk parameters. Once this is done to the large number of dense transformations in a pretrained Transformer model, gradients and optimizer states need to be stored for many parameters. This requirement adds to the cost of full model adaptation in terms of computation and memory [3, 5].

To mitigate this cost, LoRA was proposed that represents the task specific update in a low dimensional form [5]. The approach is driven by the fact that the transformations needed for downstream adaptation might be of low intrinsic rank. Rather than learning the full matrix ΔW, LoRA factorizes the change into two smaller matrices that can be learned:

$$\Delta W = BA \tag{6}$$

where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$. The rank r is selected such that

$$r \ll \min(d, k) \tag{7}$$

The adapted weight matrix can therefore be written as:

$$W' = W_0 + BA \tag{8}$$

In LoRA training, W_0 is not updated and only A and B are trained for the downstream task [5]. The forward computation of a LoRA adapted linear transformation can be represented as:

$$h = W_0 x + \frac{\alpha}{r} BAx \tag{9}$$

where x is the input vector and α is a hyperparameter that scales the values. The contribution of the low rank update with regard to the pretrained transformation is controlled by the term α/r. The original LoRA formulation randomly initializes one of the low rank matrices and the other matrix to zero. This initialization ensures that ΔW is zero at the first iteration, and enables training to start with the performance of the original pretrained model [5].

The parameter efficiency of LoRA follows directly from this factorized representation. A full update of a matrix W_0 requires dk trainable parameters, whereas the LoRA update requires $r(d + k)$ trainable parameters. The relative size of the LoRA update can therefore be expressed as

DOI: 10.54254/2755-2721/2026.37399

$r(d+k)/(dk)$. If r is significantly less than d, k, the total number of trainable parameters is significantly reduced as compared to a full matrix update. The number of parameters with gradients and optimizer states required during training is therefore also reduced. After fine-tuning, the learned low-rank update BA can be incorporated into pretrained weight matrix. Thus, LoRA can avoids introducing additional inference computation that needs to be run sequentially during inference once it has combined the weights [5].

LoRA can be injected into multiple dense matrices of a Transformer model such as the query, key, value and output projection matrices of the self-attention mechanism. The first LoRA study focuses on adapting only the query and value projection matrices in many of its Transformer experiments [5]. This structure has been followed in the present study, and LoRA modules have been added to the query and value transformations of BERT. The original BERT parameters are kept unchanged in LoRA fine-tuning, and only low-rank adaptation parameters and the task classification module are optimized.

The dimension of the update space and the number of trainable parameters are determined by a design parameter, the rank r. A smaller rank requires fewer parameters and provides a more compact representation of the task-specific update. A higher rank is more parameter intensive and is also good at adapting. With standard LoRA, the chosen low-rank matrices (LoRAs) typically have a fixed rank, which means that the low-rank capacity is evenly split between the selected LoRAs. DyLoRA considers fixed-rank selection to be a limitation because altering the effective rank of standard LoRA generally necessitates extra rank selection and training steps [6]. AdaLoRA assumes that it is not efficient to allocate equal parameters when different weight matrices are differently important in downstream adaptation [7].

The above observations are the theory behind the experimental study in Section 3. The experiments begin by comparing several fixed LoRA ranks in terms of trainable parameter count and classification accuracy. The experiments then evaluates a rank allocation strategy which is a layer-wise rank allocation array that is not uniform. This experiment aims to identify a suitable balance between the performance of the model and its parameter efficiency by strategically allocating the LoRA capacity among the BERT layers.

3. Layer-wise rank allocation and experiments

Recent LoRA extensions have explored different approaches to improving parameter efficiency. QA-LoRA combines quantization-aware training with low-rank adaptation to support more memory-efficient fine-tuning and deployment [10]. VeRA further reduces the number of trainable parameters by sharing frozen random low-rank matrices across layers and optimizing a smaller set of scaling parameters [11]. The present study focuses specifically on rank selection and layer-wise rank allocation.

3.1. Fixed-rank LoRA baselines

In LoRA, the rank r controls the size of the low-rank update space and it also determines how many parameters are trainable. Standard LoRA generally applies the same rank to all selected target modules. A larger rank increases the capacity of the low-rank update, but it also increases the number of trainable parameters. The relationship between rank and downstream performance therefore needs to be examined before introducing a more selective parameter allocation strategy. Previous studies such as DyLoRA have also identified fixed-rank selection as an important limitation of conventional LoRA [6].

DOI: 10.54254/2755-2721/2026.37399

This study first evaluates three fixed-rank LoRA configurations with $r = 4$, $r = 8$, and $r = 16$. All configurations use BERT-base-uncased as the pretrained backbone and apply LoRA to the query and value projection matrices in the self-attention layer. The dataset, training procedure, target modules, number of epochs and size of batch are the same in all three configurations. This design separates the effect of rank on trainable parameter count and validation accuracy. The experimental setup uses the same query and value adaptation settings as those in the project.

Higher ranks require more trainable parameters. LoRA with $r = 4$ contains 148,994 trainable parameters, while $r = 8$ contains 296,450. Increasing the rank to 16 results in a total of 591,362. Validation accuracy also increases with rank, but the speed of improvement gradually decreases. The accuracy at $r = 4$ is 91.06%, and it increases to 91.40% at $r = 8$; thus, an improvement of 0.34 percentage points has been achieved. The rise in rank from 8 to 16 is only an increase of 0.23 percent, or 91.63 percent.

Based on the above results, increasing the rank can improve classification accuracy, although it also leads to a substantial increase in the trainable parameter count. At $r = 4$ and $r = 16$, the number of trainable parameters is 148,994 and 591,362, respectively, and the improvement in validation accuracy is only 0.57 percent points. Therefore, it is not necessary to give a higher rank to all the selected layers; otherwise, the available parameter budget may not be used efficiently.

3.2. Layer-wise rank allocation

Fixed-rank experiments motivate a more selective allocation of low-rank capacity. Standard LoRA assigns a uniform rank to selected projection matrices across Transformer layers. The different layers of a pretrained Transformer may have different requirements for adaptation to a particular downstream task. Therefore, assigning the same rank to all layers may allocate unnecessary capacity to some layers while limiting the representational capacity of others.

This paper studies a manually designed layer-wise rank allocation strategy. The 12 Transformer encoder layers of BERT-base are divided into three groups based on their depth. The first four layers are given a rank of 4, the middle four layers are given a rank of 8, and the last four layers are given a rank of 16:

$$r_l = \begin{cases} 4, & l = 0, 1, 2, 3, \\ 8, & l = 4, 5, 6, 7, \\ 16, & l = 8, 9, 10, 11. \end{cases} \tag{10}$$

The allocation has a smaller adaptation capacity for the earlier Transformer layers and gradually increases the rank for deeper layers. Based on the heuristic assumption that the deeper layers of BERT are better suited for the task, this design was put forward. The selected rank schedule is used as a simple experimental strategy and is not an automatically optimized or theoretically optimal allocation.

The two are not the same as those in the previous adaptive LoRA methods. AdaLoRA dynamically allocates a parameter budget based on the importance score [7]. The method in this paper does not calculate layer importance during training and does not automatically find an optimal rank. The objective is to determine whether a simple non-uniform allocation according to Transformer depth can achieve better performance and parameter efficiency than a uniform fixed-rank configuration.

The resulting layer-wise structure has 345,602 trainable parameters. This value is larger than the parameter count for fixed-rank LoRA with $r = 8$, but significantly smaller than that for LoRA with

DOI: 10.54254/2755-2721/2026.37399

$r = 16$. Thus, it offers an intermediate parameter budget and distributes adaptation ability differently across the network.

3.3. Experimental setup and results

The experiment was done on the SST-2 sentiment classification task in the GLUE benchmark which is a binary classification problem [12]. SST-2 is based on the Stanford Sentiment Treebank introduced by Socher et al. [13]. There are 67, 349 training examples, 872 validation examples and 1, 821 test examples in the dataset. Each inputs consists of a sentence in the movie review and a binary sentiment label, where 0 means negative sentiment and 1 means positive sentiment. The text is tokenized using BERT tokenizer and then converted to input_ids and attention_masks and then fed into the model.

All experiments are based on BERT-base-uncased as the pretrained model. Standard LoRA and layer-wise configuration are for query and value projection matrices. The number of epochs is set to 3 and the batch size is 16. The optimization is done with AdamW. The implementation is built using PyTorch, Hugging Face Transformers and PEFT, and training is done with CUDA-enabled support of GPU. A reference of full fine-tuning is provided, where all of the BERT parameters are updated during training.

Table 1. Validation accuracy and trainable parameter comparison

Method	Trainable Parameters	Validation Accuracy
LoRA r=4	148,994	91.06%
LoRA r=8	296,450	91.40%
LoRA r=16	591,362	91.63%
Layer-wise LoRA	345,602	92.20%
Full fine-tuning	109,483,778	92.32%

The results shown in Table 1 demonstrate that the layer-wise configuration is the most performant one when compared to the other LoRA-based approaches. It achieves 92.20% of validation accuracy and 345,602 trainable parameters. Fixed-rank LoRA ($r = 16$), on the other hand, can achieve 91.63% accuracy with 591,362 trainable parameters. Thus, the layer-wise configuration boosts the validation accuracy by 0.57 percentage points and cuts down the number of trainable parameters by about 41.6%.

The comparison with full fine tuning further demonstrates the efficiency of the method with regard to the parameter. With the full fine-tuning, the highest validation accuracy of 92.32% is obtained but with 109,483,778 trainable parameters. The accuracy of full fine-tuning and layer-wise LoRA is about the same, differing by just 0.12 percentage points. Meanwhile, the number of the trainable parameters falls from over 109 million to 345,602, a reduction of about 99.68%.

The results suggest that the distribution of low-rank capacity can be as important as the total rank budget. A uniform increase in rank improves performance only gradually in the fixed-rank experiments, whereas the layer-wise configuration achieves higher accuracy with fewer parameters

DOI: 10.54254/2755-2721/2026.37399

than the $r = 16$ baseline. The current results have not shown that the selected 4, 8 and 16 schedules are optimal. However, they also provide evidence that a non-uniform rank distribution can be employed to improve the trade-off between classification performance and parameter efficiency in LoRA-based fine-tuning.

4. Conclusion

This study investigated the effects of LoRA rank allocation on parameter efficiency and classification accuracy for BERT-based sentiment analysis. Standard LoRA was first tested with fixed ranks of 4, 8 and 16 to see how increasing the rank affects the number of trainable parameters and validation accuracy. The results showed that higher ranks improved classification accuracy, but the improvement became progressively smaller as the parameter count increased. LoRA with $r = 4$, $r = 8$ and $r = 16$ achieved validation accuracies of 91.06%, 91.40 and 91.63. Based on the above results, increasing the rank of all the selected layers does not necessarily improve the performance of the downstream task.

Then, a layer-wise rank allocation strategy was used to determine if a non-uniform distribution of adaptation capacity could achieve a better trade-off between performance and parameter efficiency. The proposed configuration assigned ranks of 4, 8 and 16 to the early, middle and late groups of BERT encoder layers, respectively. The validation accuracy reached 92.20% and had 345,602 trainable parameters under this configuration. Compared with fixed-rank LoRA at $r = 16$, the layer-wise configuration reduced the number of trainable parameters by about 41.6% and increased the validation accuracy by 0.57 percentage points. Full fine-tuning achieved the highest accuracy of 92.32% but required 109,483,778 trainable parameters. The accuracy difference between full fine-tuning and the layer-wise configuration was only 0.12 percent points, and the number of trainable parameters was reduced by about 99.68 percent. Therefore, the efficiency of LoRA is also affected by rank allocation in addition to the total rank budget.

Some deficiencies in the results have not been addressed. Only BERT-base-uncased and the SST-2 sentiment classification task were used in the experiment. A layer-wise rank schedule was set manually and not determined by automatic rank search or layer-importance estimation. Only validation accuracy and the number of trainable parameters were evaluated. Training time, GPU memory usage, inference speed and variations under different random seeds were not systematically analyzed.

Future work can be done to develop an automatic or dynamic rank allocation method that determines a suitable rank based on layer importance, gradient information or task requirements. Further studies on large language models, numerous NLP datasets and difficult-to-solve downstream tasks may further verify the general effectiveness of layer-wise rank allocation in experiments. Future work will further examine memory usage, training time, and inference cost to better assess the overall efficiency of parameter-efficient fine-tuning approaches.

References

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems, 30*, 5998–6008.

[2] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional Transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 1*, 4171–4186.

[3] Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., & Gelly, S. (2019). Parameter-efficient transfer learning for NLP. *Proceedings of the 36th International Conference on*

DOI: 10.54254/2755-2721/2026.37399

Machine Learning, 97, 2790–2799.

[4] Li, X. L., & Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, 1*, 4582–4597.

[5] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2022). LoRA: Low-rank adaptation of large language models. *International Conference on Learning Representations*.

[6] Valipour, M., Rezagholizadeh, M., Kobyzev, I., & Ghodsi, A. (2023). DyLoRA: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, 3274–3287.

[7] Zhang, Q., Chen, M., Bukharin, A., He, P., Cheng, Y., Chen, W., & Zhao, T. (2023). Adaptive budget allocation for parameter-efficient fine-tuning. *International Conference on Learning Representations*.

[8] Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLoRA: Efficient finetuning of quantized LLMs. *Advances in Neural Information Processing Systems, 36*, 10088–10115.

[9] Liu, S. Y., Wang, C. Y., Yin, H., Molchanov, P., Wang, Y. C. F., Cheng, K. T., & Chen, M. H. (2024). DoRA: Weight-decomposed low-rank adaptation. *Proceedings of the 41st International Conference on Machine Learning, 235*, 32100–32121.

[10] Xu, Y., Xie, L., Gu, X., Chen, X., Chang, H., Zhang, H., Chen, Z., Zhang, X., & Tian, Q. (2024). QA-LoRA: Quantization-aware low-rank adaptation of large language models. *International Conference on Learning Representations*.

[11] Kopiczko, D. J., Blankevoort, T., & Asano, Y. M. (2024). VeRA: Vector-based random matrix adaptation. *International Conference on Learning Representations*.

[12] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2018). GLUE: A multi-task benchmark and analysis platform for natural language understanding. *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 353-355.

[13] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1631-1642.