Proceedings of the 4th International Conference on Applied Physics and Mathematical Modeling
DOI: 10.54254/2753-8818/2026.36279

Low-Rank Adaptation for Parameter-Efficient Fine-Tuning of Large Language Models

Shicheng Wei

South Forsyth High School, Cumming, USA
ctwilliamwei@gmail.com

Abstract. Modern natural language systems rely on large language models, whose sheer size makes full fine-tuning costly in computation, graphics processing unit (GPU) memory, and storage. Low-rank adaptation (LoRA) sidesteps most of that cost. It keeps the pre-trained weights frozen and captures each task-specific change as the product of two smaller matrices, so adapting a model reduces to a low-rank decomposition. This review covers LoRA and its main variants and pays particular attention to the linear algebra behind them. It first explains why the low intrinsic dimension of fine-tuning makes low-rank updates effective, then compares the major variants: quantized LoRA (QLoRA), quantization-aware LoRA (QA-LoRA), adaptive low-rank adaptation (AdaLoRA), sparse low-rank adaptation (SoRA), and weight-decomposed low-rank adaptation (DoRA). Across published studies, these methods come close to full fine-tuning accuracy while updating well under one percent of a model's parameters in some settings. For reference, LoRA cuts the trainable parameter count of Generative Pre-trained Transformer 3 (GPT-3) by four orders of magnitude, and QLoRA brings a 65-billion-parameter model within the memory of one 48 GB card. Open problems remain in choosing the rank, comparing results across studies, limiting quantization loss, and combining multiple adapters without interference. Ultimately, an established piece of linear algebra, approximating high-dimensional objects in low-dimensional subspaces, is what keeps the adaptation of very large models affordable.

Keywords: low-rank adaptation, parameter-efficient fine-tuning, large language models, matrix rank, singular value decomposition

1. Introduction

Large language models (LLMs) are central to modern natural language processing. Encoder models such as Bidirectional Encoder Representations from Transformers (BERT) demonstrated that a single pre-trained model could be adapted to a broad range of natural language understanding (NLU) tasks, while decoder-only models such as Generative Pre-trained Transformer 3 (GPT-3) later demonstrated that a sufficiently large model can answer questions, generate text, and learn from a few examples without gradient updates. Later, open releases such as Large Language Model Meta AI (LLaMA) put models of this kind in the hands of ordinary research groups [1]. However, fully fine-tuning these models demands substantial memory. As models grew from a few hundred million weights to several hundred billion, full fine-tuning stopped being practical on ordinary hardware [2].

DOI: 10.54254/2753-8818/2026.36279

Adapting GPT-3 means updating roughly 175 billion parameters, storing one full copy of the model per task, and holding gradient and optimizer states beyond what ordinary graphics processing units (GPUs) can hold [3]. Task-specific adaptation often remains necessary because pre-trained models have limited task-specific knowledge, and applications in dialogue, medicine, and finance still depend on domain-specific data. Parameter-efficient fine-tuning (PEFT) offers a compromise by reducing memory and computational requirements while retaining much of the performance of full fine-tuning. PEFT leaves the original weights untouched and trains only a thin layer of extra or chosen parameters. This cuts the computation, memory, and storage costs of adaptation while retaining much of its performance benefit [2]. Among the PEFT methods, low-rank adaptation (LoRA) has a particularly clean linear-algebraic foundation. It writes the task-specific weight update as the product of two low-rank matrices, so adapting a model becomes an exercise in matrix decomposition [3].

Research on PEFT has expanded rapidly. Adapter tuning inserts small bottleneck networks between Transformer sublayers [4]. Prompt tuning learns continuous prompt vectors at the input [5]. Prefix tuning prepends trainable vectors to the hidden states of every layer [5]. LoRA and its descendants are now among the most widely used approaches in this family [2, 3]. Quantized LoRA (QLoRA) compresses the frozen backbone to 4 bits, which lets models of up to 65 billion parameters train on one GPU [6]. Adaptive low-rank adaptation (AdaLoRA) gives each layer its own rank, guided by importance scores derived from singular value decomposition (SVD) [2]. Weight-decomposed low-rank adaptation (DoRA) first factors every pre-trained weight into a length and a direction, then updates only the direction [7]. These methods have been tested on a fairly stable set of backbones, including BERT, Robustly Optimized BERT Pretraining Approach (RoBERTa), and Decoding-enhanced BERT with disentangled attention (DeBERTa) among encoders, Text-to-Text Transfer Transformer (T5) among encoder–decoders, and the LLaMA family among decoder-only LLMs. The benchmarks are similarly standard, such as General Language Understanding Evaluation (GLUE) and SuperGLUE for language understanding, Massive Multitask Language Understanding (MMLU) and commonsense-reasoning suites for LLMs, and instruction-tuning datasets for chat-style evaluation. Individual studies vary the rank, the layers that receive adapters, the quantization of the backbone, and the resulting trainable-parameter ratio. They then report task performance, GPU memory, training time, and storage cost. Recent surveys capture both the wide variety of design choices in low-rank adaptation and the rapid growth of its literature [2]. Reported results commonly show that low-rank methods can approach full fine-tuning performance while updating well under one percent of the parameters, which helps explain the widespread use of LoRA in practice.

This paper reviews low-rank adaptation for parameter-efficient fine-tuning of LLMs, with particular attention to its linear-algebraic foundations. By showing how low-rank matrix representations translate into practical data science methods, this review addresses three central questions. First, why is low-rank structure enough to adapt very large models? Second, how do LoRA and its main variants relate to one another? Third, which problems currently limit the approach? The rest of the paper is organized as follows. Section 2 introduces matrix rank, low-rank decomposition, model fine-tuning, and parameter efficiency. Section 3 presents LoRA and its major variants. Section 4 reviews representative recent studies across models, tasks, and resource settings. Section 5 discusses limitations and prospects, and Section 6 concludes.

DOI: 10.54254/2753-8818/2026.36279

2. Basic concepts of matrix rank, model fine-tuning and parameter efficiency

A matrix's rank counts its linearly independent rows or columns, and it indicates how many dimensions the linear map spans. When a large matrix has low rank, two far smaller matrices, a tall narrow one and a short wide one, multiply together to reproduce it exactly, and their shared inner dimension equals the rank. Even a full-rank matrix whose information is mainly stored in a few directions can be approximated closely by such a product. SVD helps find the best of these approximations. It breaks any matrix into a set of directions and a list of singular values sorted from largest to smallest, and keeping only the largest few gives the best low-rank approximation of the original. This creates a huge saving in storage. A matrix with a thousand rows and a thousand columns holds a million numbers, but a rank-eight factorization of it stores only sixteen thousand.

Language models are often developed in two stages. Pre-training optimizes hundreds of millions to hundreds of billions of parameters on large unlabeled text corpora and produces weight matrices that encode general knowledge of natural language [1]. In the past, adapting to a specific task meant full fine-tuning, where every entry of every weight matrix receives gradient updates and training requires gradients and optimizer states for the model parameters. The cost of doing so grows with the model and eventually becomes infeasible [2, 3]. However, parameter-efficient fine-tuning takes the opposite approach. Training touches only a handful of new or chosen parameters while the original weights sit frozen. The fraction of parameters that receive gradients, the trainable-parameter ratio, is a common measure of parameter efficiency. Freezing the backbone removes most gradient and optimizer-state memory associated with its weights and can reduce direct modification of pre-trained representations.

Experimental evidence suggests that effective fine-tuning can often be constrained to a low-dimensional subspace [2, 3]. If the useful change to the weights lives in a low-dimensional subspace, then the update matrix is approximately low-rank, and thus the information can be carried by two small matrices.

PEFT methods differ mostly in where they put the trainable parameters. Adapter tuning inserts bottleneck modules between Transformer sublayers [4]. Prompt tuning learns soft prompt vectors at the input, while prefix tuning prepends trainable vectors to the hidden states of every layer [5]. LoRA instead reparametrizes the weight update as a low-rank product, directly connecting adaptation to the matrix algebra described above [3].

3. Low-rank adaptation methods for LLM fine-tuning

LoRA [3] keeps the pre-trained weight matrix fixed and represents the task-specific update as the product of two much smaller matrices, whose shared inner dimension, the rank, is far smaller than either dimension of the original weight matrix. The adapted layer then produces its output in two parts. The frozen weights process the input exactly as before, and the two small matrices apply a correction to the result, scaled by a constant that controls the strength of the update [2, 3]. The first small matrix is initialized randomly and the second starts at zero, so the correction vanishes at the beginning of training and ensures training begins with the exact outputs of the original pre-trained model. Only the two small matrices receive gradients. In practice, performance depends on the rank, the scaling constant, optional dropout, and the target modules, with commonly tested ranks ranging from 4 to 64 and lower ranks also evaluated for GPT-3 [3]. In the original study, only the attention layer's query and value projections received the update, whereas later work shows that extending LoRA to the feed-forward multilayer perceptron (MLP) layers can improve performance further [2].

DOI: 10.54254/2753-8818/2026.36279

Once training finishes, the correction folds back into the frozen weights, so inference runs at full speed; adapters, by contrast, sit between the network's layers and stretch the forward pass [2, 3].

The variants push this scheme in different directions. QLoRA [6] uses less memory by storing the frozen pre-trained weights in a 4-bit NormalFloat (NF4) format. The quantization constants are themselves quantized (thus called "double quantization"), and the paged optimizers manage memory spikes. Together, these techniques enable a 65-billion-parameter model to be fine-tuned on one 48 GB GPU with no reported performance loss relative to 16-bit training. AdaLoRA [2] allows different layers of the model to have different ranks. It writes the update in an SVD-like factored form and prunes the least important singular values under a global budget, so the layers that matter most end up with the largest ranks. DoRA [7] takes the length-and-direction view of each weight described earlier and confines its low-rank update to the direction. This update more closely resembles full fine-tuning and raises commonsense-reasoning accuracy over LoRA by 3.7 points on LLaMA-7B and 1.0 points on LLaMA-13B with no inference overhead. Quantization-aware low-rank adaptation (QA-LoRA) addresses an inference limitation of QLoRA [2]. In QLoRA, the adapter weights are 16-bit while the base weights are 4-bit, which means the base weights must be dequantized first, adding to inference time. QA-LoRA uses group-wise operators to align low-rank updates with the quantized base model, allowing the trained adapter to be merged into a 4-bit integer model and reducing inference overhead.

Comparisons among these methods, and against adapter tuning [4] and prefix tuning [5], are usually made along the same axes: trainable parameters, GPU memory, training time, inference overhead, task accuracy, robustness, and cross-task transfer. Set against full fine-tuning of GPT-3, LoRA trains roughly one ten-thousandth as many parameters and needs about a third of the GPU memory [3]. On LLaMA 2-7B, LoRA brings fine-tuning memory down from about 60 GB to 23 GB, which is within reach of a single consumer GPU, and speeds up forward propagation by about 1.9 times [2].

4. State-of-the-art studies

Studies of low-rank fine-tuning usually differ by the base model, the task type, data scale, the rank, the layers that receive adapters, whether the backbone is quantized, the trainable-parameter ratio, the computational cost, and the metrics reported. These core variables are used to compare the experimental results.

For natural language understanding, comparative evidence includes Razuvayevskaya et al. [4], who evaluated LoRA, adapters, and full fine-tuning across multilingual text-classification tasks. Their results show that parameter-efficient methods can substantially reduce trainable parameters and memory use, but their performance depends on the task, language, and training setting. Adaptive-rank methods such as AdaLoRA and sparse low-rank adaptation (SoRA) allocate capacity unevenly across layers or rank components [2].

For large-scale instruction tuning, QLoRA reports particularly strong results among the methods discussed here. Fine-tuning a 4-bit 65B LLaMA with LoRA adapters on one 48 GB GPU produced the Guanaco family, and after a single day of training its strongest member scored 99.3% of ChatGPT's result on the Vicuna benchmark [6]. In the reported experiments, NF4 with double quantization achieved MMLU performance comparable to 16-bit LoRA [6]. On commonsense reasoning with LLaMA backbones, DoRA beat LoRA by 3.7 points on LLaMA-7B and 1.0 on LLaMA-13B, when averaging over eight datasets. This margin still holds on newer backbones [7]. LoRA and QLoRA have also spread well beyond standard benchmarks into domain-specific

DOI: 10.54254/2753-8818/2026.36279

applications [2]. In medical evidence summarization, for example, LoRA fine-tuning improved the performance of several open-source models across automatic and human evaluations [8].

Three factors help explain the reported performance differences. First is model scale, because the effectiveness of low-rank adaptation can vary with backbone size and task requirements [2]. Second is data quality, because QLoRA's instruction-tuning outcomes depend on small but carefully curated datasets [6]. Third is task complexity, because knowledge- and skill-heavy domains such as mathematics and code can retain a larger gap to full fine-tuning [2]. Naturally, different methods are specialized to address different challenges. QLoRA and QA-LoRA combine quantization with low-rank adaptation [2, 6]. AdaLoRA and SoRA make the rank adaptive [2]. Finally, DoRA decomposes weights into magnitude and direction and applies the low-rank update specifically to the directional component [7].

5. Limitations and prospects

Low-rank adaptation also has documented weaknesses. First, the low-rank assumption does not hold uniformly. The importance of weight matrices varies across Transformer layers, so one rank shared by all modules is rarely optimal [2]. In addition, knowledge-intensive domains such as mathematics and code may require higher-rank updates, leaving a gap to full fine-tuning [2]. This creates a tradeoff between parameter efficiency and adaptation capacity [2]. Second, there is no accepted standard for choosing the rank, the scaling factor, or the target modules. Studies select them using different heuristics, which makes results hard to reproduce and methods hard to tune [2]. Third, cross-study comparison is genuinely difficult, since experiments differ in backbone scale, datasets, budgets, and metrics [2, 4]. Comparative evidence also shows that no single PEFT method dominates every setting [4]. Fourth, quantization is not free. Low-precision backbones can lose accuracy unless the data type is designed carefully, and QLoRA has to dequantize at inference time, although QA-LoRA was built to remove this overhead [2, 6]. Fifth, LoRA modules combined for multi-task or continual learning can interfere with each other, and naively merged adapters may lose performance or forget earlier tasks [2, 9]. Finally, fewer trainable parameters do not necessarily mean proportionally faster training or inference. They primarily save memory used for gradients and optimizer states, and convergence can be slower than for full fine-tuning [2]. In addition, inference is slower unless the factors are first merged back into the frozen weights [2, 3].

Several LoRA variants have been proposed to address specific limitations of the original method. Hydra widens what the adapter can express by running parallel and sequential branches together [10]. A fixed uniform rank can waste parameters, so AdaLoRA and SoRA assign rank automatically under explicit budgets [2]. Quantization-aware variants combine quantization and adaptation so that fine-tuning and inference can use the same low precision [2, 6]. Adapters can interfere with each other when combined across tasks, but mixture-of-experts (MoE) arrangements in which a router combines multiple LoRA modules can reduce task interference in multi-task adaptation [2]. Cross-modal use is also already common. DoRA yields performance gains on vision-language architectures, and LoRA is widely used for fine-tuning diffusion and multimodal models [2, 7]. From a theoretical perspective, the intrinsic-dimensionality hypothesis remains a major explanation for why low-rank updates are efficient, and sharper analyses of their expressive power and optimization behavior are needed to predict where they fail [2, 3]. The field would also benefit from evaluation benchmarks that control variables such as backbone, parameter budget, and training data, which would improve cross-study comparability [2, 4]. Ultimately, the overall trajectory is clear. As models continue to scale, mapping high-dimensional weight updates into efficient, low-dimensional subspaces will remain a core design principle.

DOI: 10.54254/2753-8818/2026.36279

6. Conclusion

This paper has reviewed low-rank adaptation as a parameter-efficient way to fine-tune large language models. The starting point was linear algebra, including matrix rank, low-rank factorization, and the SVD. The empirical observation that fine-tuning has low intrinsic dimension motivates writing weight updates as products of two small matrices, which is central to LoRA's core functionality. LoRA leaves the pre-trained weights untouched and trains nothing but the two low-rank factors. The variants of LoRA aim to improve the method in different ways. QLoRA and QA-LoRA focus on quantizing the frozen backbone, AdaLoRA and SoRA focus on adaptive rank allocation, and DoRA focuses on a finer decomposition of the weight itself. Published results show accuracy near full fine-tuning while training well under one percent of the model parameters in some settings, though rank selection, comparability across studies, quantization losses, and multi-adapter interference remain open problems. Nevertheless, the practical value of each method depends on the base model, target task, available memory, quantization setting, and evaluation protocol. Reliable comparisons therefore require consistent backbones, datasets, parameter budgets, and reporting of training cost, inference overhead, and task performance. These considerations are essential when selecting a low-rank adaptation strategy for real-world deployment. Future progress on adaptive rank, quantized low-rank training, and multi-task adaptation is likely to narrow these gaps. What low-rank adaptation ultimately shows is that an established idea from linear algebra—approximating high-dimensional objects in low-dimensional subspaces—can cut the parameter, memory, and storage costs of adapting billion-parameter models far enough to keep LLM customization within reach of ordinary hardware.

References

[1] Yang, J., Jin, H., Tang, R., Han, X., Feng, Q., Jiang, H., Zhong, S., Yin, B., & Hu, X. (2024). Harnessing the power of LLMs in practice: A survey on ChatGPT and beyond. *ACM Transactions on Knowledge Discovery from Data, 18*(6), Article 160, 1-32. https://doi.org/10.1145/3649506

[2] Mao, Y., Ge, Y., Fan, Y., Xu, W., Mi, Y., Hu, Z., & Gao, Y. (2025). A survey on LoRA of large language models. *Frontiers of Computer Science, 19*, Article 197605. https://doi.org/10.1007/s11704-024-40663-9

[3] 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. In *The Tenth International Conference on Learning Representations (ICLR 2022)*. OpenReview.

[4] Razuvayevskaya, O., Wu, B., Leite, J. A., Heppell, F., Srba, I., Scarton, C., Bontcheva, K., & Song, X. (2024). Comparison between parameter-efficient techniques and full fine-tuning: A case study on multilingual news article classification. *PLOS ONE, 19*(5), e0301738. https://doi.org/10.1371/journal.pone.0301738

[5] Prottasha, N. J., Mahmud, A., Sobuj, M. S. I., Bhat, P., Kowsher, M., Yousefi, N., & Garibay, O. O. (2024). Parameter-efficient fine-tuning of large language models using semantic knowledge tuning. *Scientific Reports, 14*, Article 30667. https://doi.org/10.1038/s41598-024-75599-4

[6] Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems 36*, 10088-10115. Curran Associates, Inc. https://doi.org/10.52202/075280-0441

[7] 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. In *Proceedings of the 41st International Conference on Machine Learning (Vol. 235)*, pp. 32100-32121. PMLR.

[8] Zhang, G., Jin, Q., Zhou, Y., Wang, S., Idnay, B., Luo, Y., Park, E., Nestor, J. G., Spotnitz, M. E., Soroush, A., Campion, T. R., Jr., Lu, Z., Weng, C., & Peng, Y. (2024). Closing the gap between open source and commercial large language models for medical evidence summarization. *npj Digital Medicine, 7*, Article 239. https://doi.org/10.1038/s41746-024-01239-w

[9] Li, L., Wang, S., Li, C., Yuan, Y., & Wang, G. (2025). DC-LoRA: Domain correlation low-rank adaptation for domain incremental learning. *High-Confidence Computing, 5*(4), 100270.

DOI: 10.54254/2753-8818/2026.36279

https://doi.org/10.1016/j.hcc.2024.100270
[10] Kim, S., Yang, H., Kim, Y., Hong, Y., & Park, E. (2024). Hydra: Multi-head low-rank adaptation for parameter-efficient fine-tuning. *Neural Networks, 178*, 106414. https://doi.org/10.1016/j.neunet.2024.106414