Proceedings of the 2023 International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/42/20230688

A concise analysis of low-rank adaptation

Yanran Chen

School of Computer Science and Technology, Xi’an Jiaotong University, Xi’an, 710049, China

2213311088@stu.xjtu.edu.cn

Abstract. Recent years the pre-trained language models have been proved to be a transformative technology within the domain of Natural Language Processing (NLP). From early word embeddings to modern transformer-based architectures, the success of models like BERT, GPT-3, and their variants has led to remarkable advancements in various NLP tasks. This paper is based on the Transformer model and explores and summarizes the application of the lightweight fine-tuning technique LoRA in pretrained language models, as well as improvements and derived technologies based on LoRA. Moreover, this paper categorizes these techniques into two main directions according to the advancements: enhancing training efficiency and improving training performance. Under these two major directions, several representative optimization and derived techniques are summarized and analyzed. Furthermore, this paper offers a perspective on the hot topics and future prospects of this research subject, and summarizes and proposes several directions that hold exploration value for the future, such as the possible avenues for further optimization and integration with other lightweight technologies.

Keywords: NLP, Pretrained Language Model, Transformer, Low-Rank Adaptation.

1. Introduction

NLP is a crucial branch of AI dedicated to enabling computers to comprehend, generate, and manipulate human language [1-3]. The development of NLP has evolved over time, from early rule-based language processing to statistical and machine learning methods, ultimately culminating in modern NLP techniques consisting of deep learning [4] and neural networks. In earlier times, researchers mainly relied on rule-based or statistical methods to process language. With the advancement of machine learning, techniques like SVM and Random Forest were introduced to NLP for various tasks, for example, text classification and sentiment analysis. As deep learning technologies emerged, particularly the development of neural networks, NLP witnessed revolutionary progress. Models like CNN and RNN were employed to process textual data, leading to achievements in word embeddings, language models and machine translation.

Nowadays, the emergence of pretrained language models has driven the forefront of NLP research. These models undergo large-scale pretraining on extensive text corpora and are fine-tuned for downstream tasks, resulting in significant performance improvements across multiple tasks. However, with the growing scale of pretrained language models, adapting them to downstream tasks becomes a critical concern. Full fine-tuning requires maintaining a large parameter backup for each task, which becomes expensive as the number of downstream tasks increases. This problem is exacerbated as pretrained models approach sizes of hundreds of billions or even trillions of parameters. Consequently, Parameter-Efficient Fine-Tuning (PEFT) [5] arises. PEFT such as Prompt Tuning [6] and Prefix-Tuning [7] primarily focuses on fine-tuning a tiny subset of trainable parameters and aims to achieve performance comparable to full fine-tuning. This approach also addresses potential structural mismatches between inputs and outputs of upstream and downstream tasks. Low-Rank Adaptation (LoRA) is a groundbreaking technique introduced by Edward Hu et al. in 2021 within the domain of PEFT. It efficiently fine-tunes pretrained models by introducing low-rank approximations to weight matrices, reducing the number of parameters, which needed to be regenerated in the fine-tuning process.

Following the introduction of LoRA, numerous researchers have further optimized and improved the technique to make it more adaptable to diverse scenarios and enhance its performance. But the summary analysis and comparative literature are insufficient. Thus, this paper reviews and analyzes the researches related to LoRA and summarize the advancements in lightweight fine-tuning techniques based on LoRA over the past two years. This encompasses improvements to the LoRA technique itself and novel fine-tuning methods that have emerged based on the foundational concept of LoRA.

2. Backgrounds

2.1. Large language model

Large language models (LLMs) are designed to handle human language. They are trained on a massive quantity of textual data, which make them to be capable of performing a various scope of missions, including text generation, summarization and more. The defining characteristic of LLMs is their massive scale, with billions of parameters that enable them to study complex patterns. These models typically base on the architectures of deep learning like transformers, which contribute to their fantastic capability across many NLP tasks.

Due to the continuous expansion of dataset sizes, the cost of training large language models like GPT-3 175B has become extremely substantial. In order to make these massive pretrained models more practical for downstream tasks, the concept of PEFT has emerged. PEFT aims to effectively fine-tune pretrained language models with minimal required parameters and computational resources. It constitutes a set of techniques in NLP tasks, enabling pretrained language models to fit in downstream missions with fewer computational resources compared to traditional fine-tuning methods. These techniques hold great significance for researchers and developers who might not have access to powerful hardware or need to perform model fine-tuning on resource-constrained devices. LoRA is a pivotal fine-tuning technique in the PEFT realm introduced by Microsoft.

2.2. Low rank adaption

LoRA is a fine-tuning method designed to simplify large models by approximating their complex high-dimensional structures with lower-dimensional ones. Specifically applied to language models, LoRA involves creating a smaller base model that can be effectively tailored for various downstream tasks. Extensive research has revealed that large models contain a significant amount of unnecessary and redundant information in their high-dimensional structure. In contrast, there exists a more compact intrinsic dimension that captures the essential information.

Therefore, during the training phase, it is possible to focus on learning these low-rank intrinsic dimensions independently, resulting in a more efficient and resource-friendly model. To illustrate, consider the case of fine-tuning a massive model like GPT-3 175B with the Adam optimizer. With LoRA, it becomes feasible to decrease the amount of trainable parameters by 10,000 times and reduce GPU memory requirements by threefold.

The primary components of LoRA comprise:

Pretrained Language Model: This refers to a large-scale language model, such as GPT-3 or BERT, that serves as the foundation for the adaptation.

Low-Rank Adaptation Layer: Low-rank matrices are introduced, which are attached to the weight matrices of the pretrained model. They play a significant role in approximating the model's high-dimensional structure.

Fine-Tuning Process: This step involves training the low-rank matrices, allowing them to adapt to the specific tasks and data. This fine-tuning process is a key element in making the model more efficient and task-specific.

LoRA parameterizes two small matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and a low-rank matrix $\Delta\boldsymbol{W}$.-s the product of them:

$$\boldsymbol{h} = \boldsymbol{W_0 x} + \Delta\boldsymbol{Wx} = \boldsymbol{W_0 x} + \boldsymbol{BAx} \tag{1}$$

where $\boldsymbol{W_0}, \Delta\boldsymbol{W} \in \mathbb{R}^{\boldsymbol{d1}\times\boldsymbol{d2}}$, $\boldsymbol{A} \in \mathbb{R}^{\boldsymbol{r}\times\boldsymbol{d2}}$ and $\boldsymbol{B} \in \mathbb{R}^{\boldsymbol{d1}\times\boldsymbol{r}}$ with $\boldsymbol{r} \ll \{\boldsymbol{d1}, \boldsymbol{d2}\}$. In the fine-tuning process, $\boldsymbol{W_0}$ is frozen and only $\boldsymbol{A}$ and $\boldsymbol{B}$ are trained. The rank $\boldsymbol{r}$ is much smaller than the dimension of $\boldsymbol{W_0}$. The detail of the structure of LoRA is shown in figure 1.

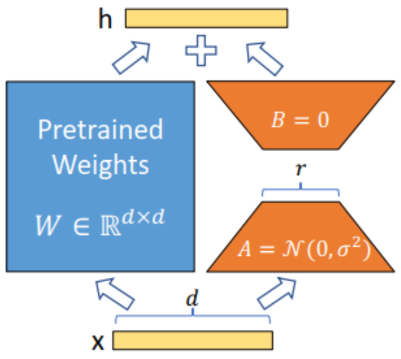
h
Pretrained Weights
$W \in \mathbb{R}^{d\times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

Figure1. Structure of LoRA.

LoRA presents a range of significant advantages. It substantially reduces the number of trainable parameters, leading to quicker and more resource-efficient fine-tuning process of LLMs. This efficiency translates into savings in computational resources. Furthermore, LoRA's adaptability shines through by allowing seamless switching between different sets of A and B matrices, making it versatile for various downstream tasks. Its linear structure ensures that it doesn't introduce inference latency, setting it apart from fully fine-tuned models. Moreover, LoRA is not exclusive and can be combined with other lightweight fine-tuning methods for added benefits.

However, it's important to acknowledge LoRA's limitations. The use of low-rank matrix approximation to represent the original high-dimensional structure may introduce approximation errors, potentially affecting the model's performance on specific tasks. Additionally, the initial LoRA application primarily focuses on weight matrices in transformers, whereas in many scenarios, FFNs play a more substantial role. Moreover, the intrinsic dimensions of a model may not remain constant across different training processes for a given task, which adds a layer of complexity to its application.

3. Research Based on the LoRA Method

3.1. Improve the efficiency of training process

Although LoRA effectively reduces the number of trainable parameters, when applied to larger models like GPT-4 1.8T, the simplified trainable parameter count through LoRA remains substantial. Therefore, it is necessary to further optimize beyond the original LoRA framework to enhance training efficiency, decrease training overhead, and ensure the accuracy of prediction results.

One effective method for reducing the number of parameters in LoRA involves freezing the matrix A, leading to a substantial reduction in trainable parameters and an enhancement in training efficiency. This variant is known as LoRA with Frozen-A (LoRA-FA) [8,9], which excels at significantly minimizing the activation memory footprint of LoRA without introducing any additional computational overhead. Specifically, the strategy is to freeze both the pretrained weight matrix $\boldsymbol{W}$ and $\boldsymbol{A}$, while only updating the projection-up weight matrix $\boldsymbol{B}$. By adopting this approach, the training process focuses solely on computing the gradient of $\boldsymbol{B}$, which requires storing a much smaller input derived from AX during the feed-forward pass. Given that $r \ll d$ ($\boldsymbol{W} \in \mathbb{R}^{d\times d}$, $\boldsymbol{A} \in \mathbb{R}^{d\times r}$, and $\boldsymbol{B} \in \mathbb{R}^{r\times d}$), the memory requirement for activations in LoRA-FA is significantly reduced. Furthermore, LoRA-FA achieves a remarkable reduction in the number of trainable parameters, reducing them from d^2 to dr, which amounts to a reduction by a factor of 2048. Initializations are critical in this context, with $\boldsymbol{A}$ being randomly initialized from a normal distribution and $\boldsymbol{B}$ initialized as zero. These initializations ensure that pretrained models with LoRA-FA modules maintain their model predictions before fine-tuning.

It's worth noting that LoRA-FA doesn't alter the feed-forward and back-propagation computations of LoRA, and thus, it doesn't introduce additional computational overhead during the fine-tuning phase. During inference, similar to LoRA, it can merge low-rank weights by adding $\boldsymbol{A}, \boldsymbol{B}$ into $\boldsymbol{W}$, which means that it doesn't introduce any extra inference latency compared to a fully fine-tuned model.

The advantage of LoRA-FA lies in maintaining a certain level of accuracy while significantly decreasing computational requirements and reducing GPU consumption compared to the original LoRA. However, due to the condition where $r \ll d$, while the improvements of LoRA-FA over Full Fine-Tuning are evident, its enhancement compared to LoRA is not substantial enough. Additionally, due to limited training parameters, it may exhibit slightly poorer performance on certain downstream tasks compared to LoRA.

Another approach in the realm of model efficiency is model quantization. Model quantization involves approximating the continuous-valued floating-point model weights or tensors used in a model with fixed-point approximations, often using a limited set of discrete values, such as int8. This approximation method significantly reduces the precision of inference while keeping the model's inputs and outputs in floating-point format. This technique serves various purposes, including shrinking model size, reducing model memory consumption, and speeding up model inference.

One specific implementation of model quantization is QLoRA [10], which offers an efficient fine-tuning approach that effectively reduces memory usage. This reduction is substantial enough to fine-tune a model on a 48GB GPU while preserving full 16-bit fine-tuning task performance. In QLoRA, gradients are backpropagated through a frozen, 4-bit quantized pretrained language model into LoRA.

Several innovative methods are introduced in QLoRA to save memory without compromising performance:

(a) 4-bit NormalFloat (NF4): This is a new type of data, which is information theoretically optimal for weights that follow a normal distribution.

(b) Double Quantization: This technique quantizes the quantization constants to reduce the average memory footprint.

(c) Paged Optimizers: It manages memory spikes effectively.

In summary, both of these approaches share a common idea of enhancing training efficiency by reducing the training load. However, their implementations differ. LoRA-FA achieves this by freezing LoRA layers to further reduce trainable parameters, while QLoRA optimizes parameter models by quantizing the pretrained model into 4-bit values.

3.2. Enhance the accuracy of prediction

Because LoRA uses low-rank matrix approximation of the original high-dimensional structure, it might introduce some approximation errors that could impact the model's performance on certain tasks. As a result, certain studies have proposed improvements to LoRA aimed at enhancing its accuracy across various downstream tasks, thus increasing the reliability and effectiveness during training.

A cutting-edge approach named Generalized LoRA (GLoRA) [11] achieve universal and PEFT across various tasks. Building upon the LoRA framework, GLoRA introduces a generalized prompt module that optimizes pretrained model weights and adjusts intermediate activations. This enhancement provides GLoRA with greater flexibility and adaptability when applied to diverse tasks and datasets.

To be specific, GLoRA streamlines the parameter adaptation process by merging an expandable and modular layer-wise structure search [12]. This innovative approach learns unique adapters for every layer, contributing to improved model performance. Thorough experimentation provides compelling evidence that GLoRA surpasses all prior techniques on various benchmarks encompassing natural, domain-specific, and structured tasks. It attains remarkable accuracy while making more efficient use of parameters and computational resources across a diverse set of datasets.

The consolidated formulation to represent all tunable spaces can be represented as follows,and the detailed illustration is shown in figure 2.

$$f(x) = (\boldsymbol{W_0} + \boldsymbol{W_0 A} + \mathbf{B})x + \boldsymbol{CW_0} + \boldsymbol{Db_0} + \boldsymbol{E} + \boldsymbol{b_0} \tag{2}$$

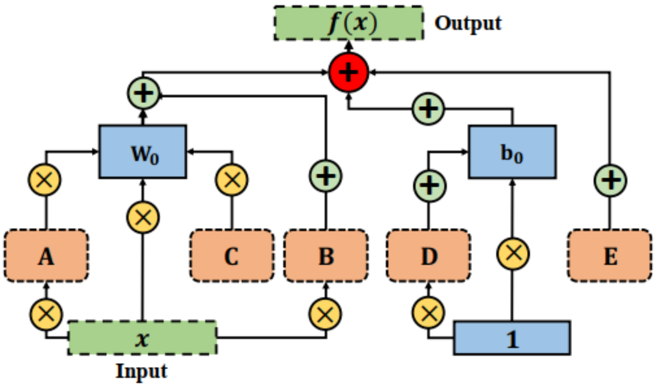

Figure 2. Process of GloRA.

Here $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}, \boldsymbol{E}$ are the trainable support tensors for downstream tasks in our GLoRA, $\boldsymbol{W_0}$ and $\boldsymbol{b_0}$ are frozen during whole fine-tuning. $\boldsymbol{A}$ is utilized to scale the weight. $\boldsymbol{B}$ has the role to scale the input and shift the weight. $\boldsymbol{C}$ is the layer-wise prompt serving a similar function of VPT-Deep, $\boldsymbol{D}$ and $\boldsymbol{E}$ are used to scale and shift the bias, respectively.

One notable advantage of GLoRA is its structural re-parameterization design, which ensures that it doesn't incur any extra inference cost. This makes GLoRA a practical and efficient solution for resource-limited applications.

AdaLoRA [13] presents an innovative improvement method to address certain limitations observed in LoRA. Unlike LoRA, which mandates the pre-specification of a consistent eigen-rank 'r' for each incremental matrix, AdaLoRA recognizes the substantial variations in the importance of weight matrices across different modules and layers during fine-tuning of pretrained models. Additionally, LoRA primarily focuses on training the attention mechanism and doesn't explicitly handle the FFN. To overcome these shortcomings, AdaLoRA introduces a set of enhancements.

One key feature of AdaLoRA is its dynamic allocation of parameter budgets to weight matrices based on importance scores. It achieves this by parameterizing incremental updates using singular value decomposition (SVD), allowing for the removal of unimportant singular values while preserving their corresponding singular vectors. This approach significantly accelerates computations by reducing the parameter budgets, all the while maintaining the potential for future recovery and ensuring training stability.

AdaLoRA comprises two essential components:

Adaptation based on SVD: This element constructs the incremental matrices using singular value decomposition, facilitating efficient updates.

Rank allocation with importance-aware: AdaLoRA trims unnecessary singular values using a freshly devised significance measure, guaranteeing the retention of only pertinent information.

Initially, representing the incremental updates of the pre-trained weight matrices through singular value decomposition., where $P \in \mathbb{R}^{d_1 \times r}$ and $Q \in \mathbb{R}^{r \times d_2}$ mean the different singular vectors of Δ, while the diagonal matrix $\Lambda \in \mathbb{R}^{r \times r}$ contains the singular values$\{\lambda i\}_{1 \le i \le r}$ with $r \ll \ min(d1, d2)$.

$$\boldsymbol{W} \ = \ \boldsymbol{W(0)} + \Delta = \ \boldsymbol{W(0)} \ + \ \boldsymbol{P\Lambda Q} \tag{3}$$

An additional penalty term is introduced in the training loss to keep the two singular matrices P and Q orthogonal, thereby avoiding extensive SVD computations and stabilizing the training process.

$$R(P,Q) = \left|\left|P^T P - I\right|\right|_F^2 + \left|\left|Q^T Q - I\right|\right|_F^2 \tag{4}$$

The incremental updates are parameterized using SVD, and unimportant singular values are pruned based on importance metrics, while retaining the singular vectors. Since performing an accurate SVD decomposition on a large matrix is computationally intensive, this approach accelerates calculations by reducing their parameter budget. Simultaneously, it preserves the potential for future recovery and stabilizes the training process.

Second, adjusting the allocation of incremental matrices. AdaLoRA assigns higher ranks to crucial incremental matrices to capture finer and task-specific information, while reducing the rank of less important matrices to prevent overfitting and save computational budget. The authors applied SVD-Based Adaptation to parameter matrices such as W_f, W_q, W_v, W_k. To control the budget of trainable parameters, it's necessary to dynamically allocate trainable parameters for each parameter during the training process.

In comparison to LoRA, AdaLoRA offers several notable advantages in its design. AdaLoRA prunes only the singular value matrix Λ while leaving the singular vectors untouched, which simplifies the recovery of inadvertently pruned singular values during training, contributing to enhanced training stability. Additionally, AdaLoRA employs orthogonal matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$, which differ from the non-orthogonal matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. The pruning operations training do not affect the singular vectors corresponding to other singular values, enhancing the model's stability and improving generalization performance.

In summary, AdaLoRA offers a valuable enhancement by dynamically allocating resources to weight matrices based on their importance and efficiently using SVD for incremental updates, thereby addressing limitations in LoRA. These methods achieve improvements in LoRA performance through distinct approaches. GloRA enhances the pretrained model's weights by introducing a generalized prompt module and adjusting intermediate activation functions. This enables greater flexibility and capability for various tasks and datasets. It employs an extensible, modular, and layer-wise structure search to learn a unique adapter for each layer. On the other hand, AdaLoRA optimizes and refines the original LoRA model by introducing importance scores to allocate budget for weight matrices. It also incorporates a penalty term to mitigate excessive computations in SVD and stabilize training.

4. Conclusion

Today, NLP has entered the era of modern NLP, characterized by the dominance of deep learning and neural networks. The development of PEFT techniques has also been rapidly progressing. These techniques alleviate the training costs associated with large pretrained models, enabling researchers to efficiently adapt these models to new tasks even under limited computational resources, facilitating effective transfer learning. This paper mainly categorizes, organizes, and summarizes the research landscape of novel lightweight fine-tuning techniques based on LoRA. The study classifies research results based on their objectives into categories such as "enhance the efficiency of the training process" and "enhance the accuracy of prediction",summarizing the technical characteristics and pros and cons of each category of methods. Building upon existing research achievements, the paper identifies several directions for future exploration in this field:

(1) Optimizing Intrinsic Dimensionality:

Despite existing research efforts in optimizing the intrinsic dimensionality of LoRA model weight matrices, there's still room for better understanding how to allocate intrinsic dimensions during the training of downstream task models to enhance training performance. Achieving this effectively remains a challenge.

(2) Combining and Optimizing Multiple Fine-Tuning Techniques:

Since LoRA is not in conflict with many other fine-tuning techniques, exploring how to rationally combine and optimize various techniques to achieve significantly better results offers a promising avenue for research.

References

[1] Ashish V, Noam S, Niki P, Jakob U, Llion J, Aidan N G, Łukasz K and Illia P June 2017 Attention is all you need. arXiv:1706.03762

[2] Jacob D, Ming-Wei C, Kenton L and Kristina T May 2019b BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805

[3] Edward J Hu, Yelong S, Phillip W, Zeyuan Allen-Z, Yuanzhi L, Lora: Low-rank adaptation of large language models. arXiv:2106.09685,2021.
Thijs V, Sai P K, and Martin Powersgd: Practical low-rank gradient compression for distributed optimization. J. Advances in Neural Information Processing Systems, 32, 2019

[4] Misha D, Babak S, Laurent D, Marc'A R, and Nando D F 2014 Predicting parameters in deep learning. International Conference on Neural Information Processing Systems 2148–2156, 2013

[5] Haoyu H, Jianfei C, Jing Z, Dacheng T and Bohan Z Sensitivity-aware visual parameter-efficient tuning. 2023

[6] Neil H, Andrei G, Stanislaw J Bruna M, Quentin D L, Andrea G, Mona A and Sylvain G Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, 2790–2799, 2019.

[7] Brian L, Rami Al-R and Noah C The Power of Scale for Parameter-Efficient Prompt Tuning. 2021, arXiv:2104.08691.

[8] Xiang L L and Percy L Prefix-Tuning: Optimizing Continuous Prompts for Generation. 2021, arXiv:2101.00190.

[9] Armen A, Luke Z and Sonal G December Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. 2020 arXiv:2012.13255

[10] Longteng Z, Lin Z, Shaohuai S, Xiaowen C and Bo L August 2023 LoRA-FA: memory-efficient Low-Rank Adaptation for large language models fine-tuning. arXiv:2308.03303 [cs], URL http://arxiv.org/abs/2308.03303.

[11] Tim D, Artidoro P, Ari H and Luke Z May QLoRA: efficient finetuning of quantized LLMs. 2023, arXiv:2305.14314.

[12] Arnav C, Zhuang L, Deepak G, Eric X and Zhiqiang S One-for-all: Generalized lora for parameter-efficient fine-tuning, 2023 arXiv:2306.07967.

[13] Qingru Z, Minshuo C, Alexander B, Pengcheng H, Yu C, Weizhu C and Tuo Z March 2023 Adaptive budget allocation for parameter efficient fine-tuning. arXiv:2303.10512