LoRA Lecture Notes

Sanjana Surapaneni, Nicholas Croteau, Kerui Wu

November 2024

1 Introduction

1.1 What is LoRA?

Fine tuning is used as an efficient method to retrain parameters of machine learning models. However, the process of fine tuning for optimal parameters is computationally expensive.

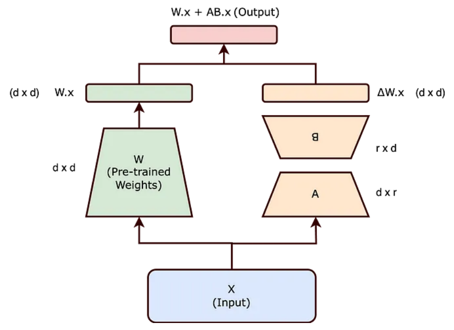

Figure 1: Introducing the process of LoRA

Low Rank Adaptation is a lightweight technique that allows models to be trained with better computational resources. Overall, this is done by freezing the pretrained model weights and performing rank decomposition to for efficient storage. This greatly reduces the number of trainable parameters.The model's parameters are updated in such a way that maintains low rank structure.

1.2 A review on rank

If you remember from Linear Algebra, the rank of a matrix refers to the maximum number of linearly independent rows or columns it possesses. Another way to envision rank is that linearly independent vectors within the matrix cannot be expressed as a linear combination of other vectors in the set.

This is useful because the matrices can now be represented by a small number of linearly independent columns. Essentially, we are narrowing down the

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 6 & 9 \\ 2 & 4 & 6 \end{bmatrix}$$

Figure 2: Matrix with rank 1

important information by removing the redundancy of linear combinations of other columns. A high dimensional matrix can have a small rank or be represented by a small number of columns.

2 Alternative Methods

Before looking into LoRa, there are multiple other techniques used to achieve similar results. This is only a surface level detailing of each technique, but it is worth to consider to realize the need and power of LoRa in contrast to its contemporaries.

2.1 Fine Tuning

Fine tuning is by far the most widely used method of those discussed, but also the most compute- and memory-intensive. Fine tuning is the full retraining of parameters for a specific task. This in effect creates a new model suited to only this new task. Therefore, if you had multiple tasks, you would have to fully retrain a given model for each given task.

Fine tuning comes with its own downsides other than just computational; a fine tuned model is prone to over fitting and forgetting general knowledge, a downside of adjusting all weights in a model.

2.2 Prefix-Tuning, Prompt Tuning

This group of techniques focuses on adding trainable tokens or learnable prefixes to a models input to steer output as desired. However, performance non-monotonically changes, that is to say, it is nebulous and unreliable to an extent, as tweaking prompts is more of an art than a science. There is also the issue of scalability, as a model at large cannot be changed just by tweaking its inputs for one task or output.

2.3 Adaptor Layers

This technique has the idea to add extra NN layers into a given architecture that are only trained on the new task desired. Expectantly, it has a low number of parameters compared to full fine tuning. The problem with adaptor layers is that it does not merge during the inference of a model, thereby creating latency for every new layer inserted. New layers similarly will increase computation and memory intensity.

2.4 Quantization

Quantization reduces the precision of parameters in a model. E.g, 32-bit floats to 8 bit or lower. This expectantly reduces the memory footprint and speeds up computation depending on the degree of quantization. Conversely, with each degree of precision lost, you are potentially losing accuracy. This technique also doesn't actually redeploy a given architecture for a new task, it is only used to help speed up and compress the size of models.

We will see now how LoRa addresses the above shortcomings and why it was such a breakthrough in the field of re-purposing models.

3 LoRa Operation

3.1 Standard Linear Transformation

Starting with a typical transformation, consider a weight matrix $W \in \mathbb{R}^{d\times k}$ that is part of a linear transformation in a layer of a neural network, where d is the input dimension and k is the output dimension. Given an input vector $x \in \mathbb{R}^d$, the output of the linear transformation is:

$$y = Wx$$

3.2 Adding a Low-Rank Adaptation

To adapt the model to a new task without modifying W directly, we introduce a low-rank matrix ΔW as a trainable update to W. This gives us the modified weight matrix:

$$W' = W + \Delta W$$

where W remains fixed, and only ΔW is updated during the tuning.

3.3 Low-Rank Decomposition of ΔW

The matrix $\Delta W \in \mathbb{R}^{d\times k}$ can be decomposed as a product of two smaller matrices:

$$\Delta W = BA$$

where $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$, and r is the rank of the adaptation, with $r \ll \min(d, k)$.

Thus, the adapted weight matrix becomes:

$$W' = W + BA$$

This decomposition reduces the number of trainable parameters from $d \times k$ (in the full-rank case) to $r(d+k)$ in the low-rank case, which is much smaller when r is small.

3.4 Forward Pass with the Adapted Weight Matrix

In the forward pass, we use the modified weight matrix W' instead of W:

$$y = W'x = (W + BA)x$$

Expanding this, we get:

$$y = Wx + B(Ax)$$

Here, Wx represents the original, pretrained transformation, $B(Ax)$ is the low-rank adaptation specific to the new task.

3.5 Training Only the Low-Rank Adaptation

During training on the new task, we freeze W and update only B and A. This low-rank adaptation captures the task-specific information while leaving the original model parameters unchanged, leading to a parameter-efficient and memory-efficient fine-tuning process.

This approach enables effective task-specific adaptation with a small number of trainable parameters $r(d+k)$, thus making it an efficient alternative to full fine-tuning and the other methods discussed.

4 Potential Applications

LoRA's strong adaptability and efficiency have significantly improved various applications, targeting transfer learning and improving model performance, including accuracy, scalability, and robustness. In this section, we will briefly introduce some of its applications in Natural Language Processing(NLP) and Computer Vision(CV) fields based on the survey [8].

4.1 Natural Language Processing

The rapid advancements in pre-trained language models, especially large language models (LLMs), are transforming approaches to language tasks due to their impressive performance. Despite being trained on vast amounts of general data, these models still need further fine-tuning on task-specific data to perform well in downstream tasks. Consequently, using LoRA for fine-tuning these pre-trained models is a logical choice, as it helps lower the computational resources needed.

LoRA is widely adopted in LLaMA for diverse traditional NLP tasks, such as emotion recognition [12], text classification [7], and role recognition [1]. [13]

proposed a customized LoRA structure to improve the pre-trained language model's performance for Human-Centered Text Understanding(HCTU), a type of domain adaptation to transfer the pre-trained model to match the user's preference.

In addition to traditional NLP tasks, several methods have been proposed to apply LoRA to improve model performance in various code-related tasks. [10] uses LoRA to fine-tune Llama for automated program repair(APR); [9] fine-tuned with LoRA for Text-to-SQL task.

4.2 Computer Vision

In vision tasks, LoRA is primarily applied to image generation and image segmentation. In dealing with image generation tasks, LoRA is widely used in diffusion models to address various image generation tasks while reducing computational resources. [6, 3] use LoRA to fine-tune diffusion models for image style transfer, while [5] apply LoRA to text-to-image generation. Similar to [13], which personalizes LoRA to improve the model's performance, [4] proposes Smooth Diffusion, which uses LoRA to achieve smoothness in the latent space, leading to better performance in various image generation tasks.

The image segmentation task aims to divide an image into multiple meaningful regions or objects. SAM has been proposed as a foundational model for image segmentation and demonstrated superior generalization ability. [2] propose SamLP, which utilizes LoRA to adapt SAM for efficient segmentation of license plates. [11] fine-tunes SAM's encoder using LoRA for instance segmentation task, which can in turn be used in structural damage detection.

5 Empirical Results

Here we will display side by side comparisons of computational expenses between contemporary LLM's and Image Generation with and without the addition of LoRA.

References

[1] Tobias Bornheim, Niklas Grieger, Patrick Gustav Blaneck, and Stephan Bialonski. Speaker attribution in german parliamentary debates with qlora-adapted large language models. *arXiv preprint arXiv:2309.09902*, 2023.

[2] Haoxuan Ding, Junyu Gao, Yuan Yuan, and Qi Wang. Samlp: A customized segment anything model for license plate detection. *arXiv preprint arXiv:2401.06374*, 2024.

[3] Yarden Frenkel, Yael Vinker, Ariel Shamir, and Daniel Cohen-Or. Implicit style-content separation using b-lora. *arXiv preprint arXiv:2403.14572*, 2024.

[4] Jiayi Guo, Xingqian Xu, Yifan Pu, Zanlin Ni, Chaofei Wang, Manushree Vasu, Shiji Song, Gao Huang, and Humphrey Shi. Smooth diffusion: Crafting smooth latent spaces in diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7548–7558, 2024.

[5] Likun Li, Haoqi Zeng, Changpeng Yang, Haozhe Jia, and Di Xu. Block-wise lora: Revisiting fine-grained lora for effective personalization and stylization in text-to-image generation. *arXiv preprint arXiv:2403.07500*, 2024.

[6] Shaoxu Li. Diffstyler: Diffusion-based localized image style transfer. *arXiv preprint arXiv:2403.18461*, 2024.

[7] Zongxi Li, Xianming Li, Yuzhang Liu, Haoran Xie, Jing Li, Fu-lee Wang, Qing Li, and Xiaoqin Zhong. Label supervised llama finetuning. *arXiv preprint arXiv:2310.01208*, 2023.

[8] Yuren Mao, Yuhang Ge, Yijiang Fan, Wenyi Xu, Yu Mi, Zhonghao Hu, and Yunjun Gao. A survey on lora of large language models. *arXiv preprint arXiv:2407.11046*, 2024.

[9] Richard Roberson, Gowtham Kaki, and Ashutosh Trivedi. Analyzing the effectiveness of large language models on text-to-sql synthesis. *arXiv preprint arXiv:2401.12379*, 2024.

[10] André Silva, Sen Fang, and Martin Monperrus. Repairllama: Efficient representations and fine-tuned adapters for program repair. *arXiv preprint arXiv:2312.15698*, 2023.

[11] Zehao Ye, Lucy Lovell, Asaad Faramarzi, and Jelena Ninic. Sam-based instance segmentation models for the automation of masonry crack detection. *arXiv preprint arXiv:2401.15266*, 2024.

[12] Yazhou Zhang, Mengyao Wang, Prayag Tiwari, Qiuchi Li, Benyou Wang, and Jing Qin. Dialoguellm: Context and emotion knowledge-tuned llama models for emotion recognition in conversations. *arXiv preprint arXiv:2310.11374*, 2023.

[13] You Zhang, Jin Wang, Liang-Chih Yu, Dan Xu, and Xuejie Zhang. Personalized lora for human-centered text understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19588–19596, 2024.