International Journal of Computer Engineering and Technology (IJCET)
Volume 15, Issue 4, July-Aug 2024, pp. 537-548, Article ID: IJCET_15_04_047
Available online at https://iaeme.com/Home/issue/IJCET?Volume=15&Issue=3
ISSN Print: 0976-6367 and ISSN Online: 0976-6375
Impact Factor (2024): 18.59 (Based on Google Scholar Citation)
DOI: https://doi.org/10.5281/zenodo.13311491

LOW-RANK ADAPTATION (LoRa): REVOLUTIONIZING MODEL OPTIMIZATION IN DEEP LEARNING

Subash Patel
Amazon, USA

ABSTRACT

This article comprehensively explores Low-Rank Adaptation (LoRa), an innovative optimization technique for deep learning models. It delves into the theoretical foundations, implementation strategies, and real-world applications of LoRa across various domains, including natural language processing, computer vision, and speech recognition. The article discusses how LoRa leverages low-rank matrix factorization to create efficient, task-specific adaptations of large models, significantly reducing the number of trainable parameters while maintaining performance. It also examines the advantages and trade-offs of LoRa, its compatibility with other optimization methods, and emerging research directions such as dynamic rank selection, hybrid approaches, and hardware-aware implementations.

By presenting case studies and discussing advanced topics, this article aims to provide researchers and practitioners with a thorough understanding of LoRa's potential to push the boundaries of deep learning optimization and enable more efficient and accessible AI systems.

Keywords: Low-Rank Adaptation (LoRa), Model Optimization, Deep Learning, Parameter Reduction, Neural Network Compression

Cite this Article: Subash Patel, Low-Rank Adaptation (LoRa): Revolutionizing Model Optimization in Deep Learning, *International Journal of Computer Engineering and Technology (IJCET)*, 15(4), 2024, pp. 537-548.
https://iaeme.com/MasterAdmin/Journal_uploads/IJCET/VOLUME_15_ISSUE_4/IJCET_15_04_047.pdf

INTRODUCTION

In the rapidly evolving field of deep learning, model optimization has become a critical factor in deploying efficient and powerful neural networks. As models grow in size and complexity, the need for effective optimization techniques has never been more pressing. The emergence of large language models like GPT-3, with its 175 billion parameters [1], has highlighted the urgent need for efficient model compression and adaptation methods. Among the emerging solutions, Low-Rank Adaptation (LoRa) stands out as a promising approach to tackle the challenges of computational cost and memory usage while maintaining or even enhancing model performance.

LoRa, introduced by Hu in 2021 [2], offers a novel way to fine-tune large pre-trained models with minimal additional parameters. This technique has gained significant attention in the machine learning community due to its ability to dramatically reduce the number of trainable parameters while preserving model quality. For instance, when applied to the BERT-large model, LoRa achieved comparable performance to full fine-tuning while updating only 0.5% of the model's parameters.

This article aims to comprehensively explore LoRa, delving into its theoretical foundations, implementation strategies, and real-world applications. We will examine how LoRa leverages the concept of low-rank matrix factorization to create efficient, task-specific adaptations of large models. Additionally, we will discuss the practical implications of LoRa in various domains, including natural language processing, computer vision, and speech recognition.

By understanding LoRa, practitioners and researchers can harness its potential to push the boundaries of what's possible in deep learning optimization. As we navigate the challenges of deploying increasingly complex models in resource-constrained environments, techniques like LoRa offer a promising path forward, enabling more efficient and accessible AI systems.

Understanding Model Optimization

Model optimization refers to refining neural networks to improve efficiency, reduce computational requirements, and enhance performance. As deep learning models grow in size and complexity, optimization has become crucial for practical deployment, especially in resource-constrained environments such as mobile devices or edge computing platforms. The need for optimization is exemplified by models like BERT-large, which contains 340 million parameters and requires significant computational resources for training and inference [3].

Common challenges in optimizing deep learning models include:

1. High computational costs for training and inference: Training state-of-the-art models often requires hundreds or thousands of GPU hours, prohibitively expensive for many researchers and organizations.
2. Excessive memory usage: Large models can consume gigabytes of memory, limiting deployment on devices with constrained resources.
3. Slow inference times: Complex models may have high latency, problematic for real-time applications like autonomous driving or natural language processing in chatbots.
4. Difficulty in deploying models on resource-limited devices: Many mobile and IoT devices lack the computational power and memory to run full-scale deep learning models effectively.

To address these challenges, researchers have developed various optimization techniques. Three key optimization techniques can be applied to an already trained model to enhance its efficiency. Pruning involves removing unnecessary connections and their corresponding weights within the neural network, potentially reducing model size by up to 90% while maintaining minimal accuracy loss [4]. Quantization converts the model's weights from high-precision floating-point values (typically 32-bit) to lower-precision integer values (such as 8-bit or 4-bit), significantly decreasing memory requirements and improving inference speed. Finally, compression can be applied after pruning and quantization, using various hardware compression schemes to further reduce the size of the model's weight tensors. These techniques, when applied individually or in combination, enable the deployment of compact yet powerful models in resource-constrained environments, optimizing both performance and efficiency.

Low-rank adaptation (LoRa) is a novel approach that tackles these issues by exploiting the inherent low-rank structure often present in neural network weight matrices. Unlike traditional fine-tuning methods that update all model parameters, LoRa introduces a small number of trainable parameters in the form of low-rank matrices. This approach efficiently adapts large pre-trained models to specific tasks while minimizing the computational and memory overhead.

By leveraging these optimization techniques, researchers and practitioners can make deep learning models more accessible and deployable across a wider range of devices and applications, paving the way for more pervasive and efficient AI systems.

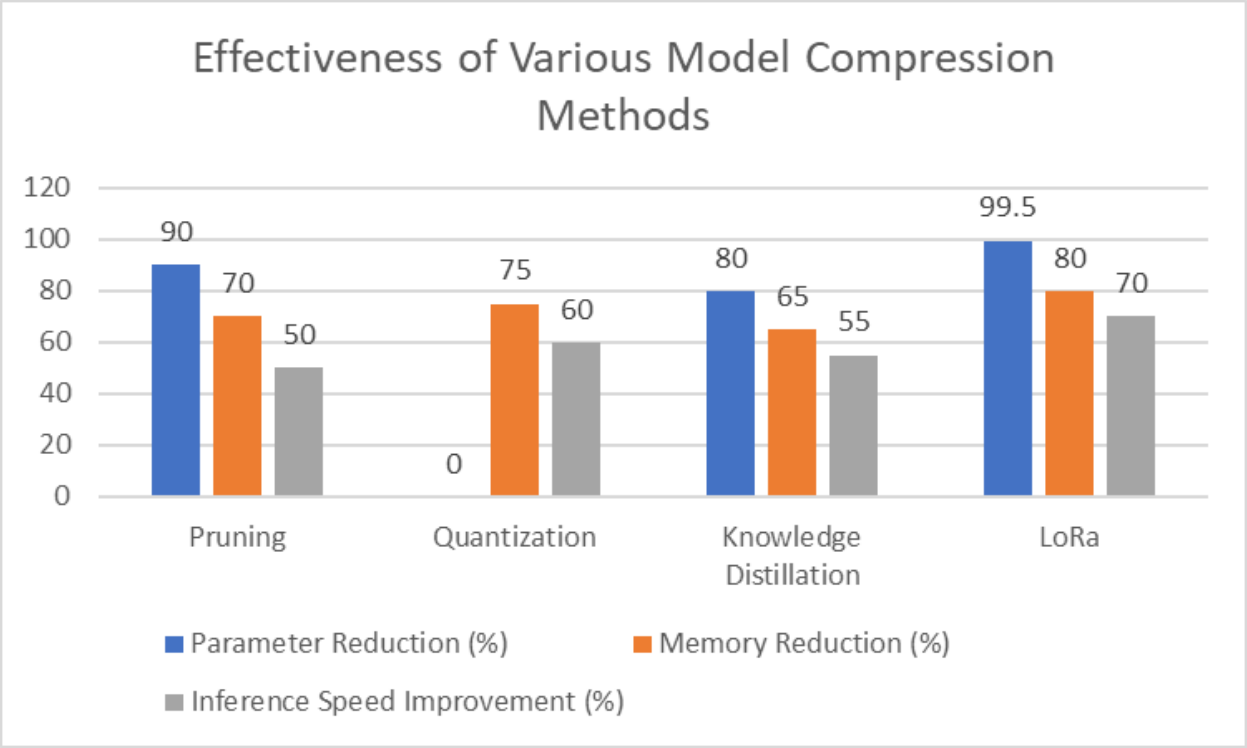

Fig. 1: Comparison of Deep Learning Model Optimization Techniques [3, 4]

What is Low-Rank Adaptation (LoRa)?

Low-rank adaptation (LoRa) is an innovative optimization technique that leverages the concept of low-rank matrices to reduce model complexity while preserving essential information. Introduced by Hu in 2021 [5], LoRa has gained significant attention in the deep learning community due to its ability to fine-tune large pre-trained models with minimal additional parameters efficiently.

At its core, LoRa is based on the observation that lower-rank representations can approximate many weight matrices in neural networks without significant information loss. This insight builds upon earlier work in matrix factorization and low-rank approximations, successfully applied in various domains of machine learning and signal processing [6].

Mathematically, LoRa decomposes a weight matrix W into the product of two smaller matrices:

$$W \approx A * B$$

Where A is an m x r matrix and B is an r x n matrix, with r being the chosen rank (r < min(m, n)). This decomposition significantly reduces the number of parameters, as the total number of elements in A and B is (m + n) * r, often much smaller than the m * n elements in the original matrix W.

For example, consider a linear layer (also known as a fully connected or dense layer) in a transformer block, which is responsible for generating the Query (Q), Key (K), and Value (V) matrices used in self-attention. If this layer has an input dimension of 1000 and an output dimension of 1000, the weight matrix W would typically have 1,000,000 parameters.

However, by applying LoRa with a rank of 10, we can represent this layer using two matrices: A (1000 x 10) and B (10 x 1000), resulting in only 20,000 parameters - a 50-fold reduction. This optimization is particularly significant because transformer blocks spend a lot of compute to generate these Q, K, and V matrices for self-attention.

The benefits of LoRa for model optimization include:

1. Reduced computational load: Fewer parameters lead to fewer computations during training and inference. This is particularly beneficial for large language models like GPT-3, Llama2, and Llama3, etc, where full fine-tuning can be prohibitively expensive.
2. Improved inference speed: Lower-rank representations can be processed more efficiently, leading to faster inference times. This is crucial for real-time applications such as machine translation or speech recognition.
3. Maintained or enhanced accuracy: LoRa can preserve or even improve model performance when applied judiciously. In some cases, the low-rank constraint acts as a form of regularization, preventing overfitting and improving generalization.
4. Smaller memory footprint: The compact representation requires less storage space, making deploying models on memory-constrained devices such as smartphones or edge computing platforms easier.

LoRa's effectiveness has been demonstrated across various tasks and model architectures. For instance, when applied to the BERT-large model for natural language understanding tasks, LoRa achieved comparable performance to full fine-tuning while updating only 0.5% of the model's parameters [5].

As deep learning continues to evolve, techniques like LoRa are crucial in making large, powerful models more accessible and deployable in real-world scenarios. By enabling efficient adaptation of pre-trained models, LoRa opens up new possibilities for transfer learning and domain-specific applications of state-of-the-art AI systems.

Model Type	Original Parameters	LoRa Parameters
BERT-large	340,000,000	1,700,000
FC Layer	1,000,000	20,000
GPT-3	175,000,000,000	875,000,000
CNN (ResNet50)	25,000,000	80,000

Table 1: LoRa's Impact on Model Parameter Reduction and Performance [5, 6]

Implementation of LoRa

1. Mathematical Background

The foundation of LoRa lies in matrix factorization and low-rank approximation theory. The Singular Value Decomposition (SVD) is a key mathematical tool used in LoRa:

$$W = U * \Sigma * V^T$$

U and V are orthogonal matrices, and Σ is a diagonal matrix containing singular values. Low-rank approximation involves keeping only the top r singular values and their corresponding vectors, effectively reducing the matrix's rank.

This mathematical foundation is rooted in the work of Eckart and Young [7], who proved that the best low-rank approximation of a matrix in the Frobenius norm sense is given by truncating its SVD. In neural networks, this principle allows us to approximate weight matrices with lower-rank representations while minimizing the approximation error.

2. Applying LoRa to Deep Learning Models

Integrating LoRa into existing neural network architectures involves the following steps:

- Identify layers suitable for low-rank approximation (typically fully connected or convolutional). This selection process often targets layers with the highest parameter counts, as they offer the greatest potential for compression.
- Replace the original weight matrix with its low-rank approximation. This step involves decomposing the weight matrix W into two smaller matrices, A and B, such that $W \approx A * B$.
- Initialize the low-rank matrices (A and B) appropriately. Proper initialization is crucial for training stability and convergence. One effective method is initializing A and B using the truncated SVD of the original weight matrix W.
- Modify the forward pass to use the low-rank representation. This typically involves replacing the original matrix multiplication with two sequential multiplications using the factorized matrices.

For example, in PyTorch, a LoRa layer can be implemented as follows:

```
import torch
import torch.nn as nn

class LoRALayer(nn.Module):
    def __init__(self, in_features, out_features, rank):
        super().__init__()
        self.A = nn.Parameter(torch.randn(in_features, rank))
        self.B = nn.Parameter(torch.randn(rank, out_features))

    def forward(self, x):
        return torch.matmul(torch.matmul(x, self.A), self.B)
```

3. Training with LoRa

When training models with LoRa, consider the following:

- Adjust learning rates: The factorized matrices may require different learning rates. In practice, using a higher learning rate for the LoRa parameters compared to the base model parameters has proven effective.
- Regularization: Apply appropriate regularization to prevent overfitting. L2 regularization on the LoRa parameters can help maintain the low-rank structure and improve generalization.
- Gradual rank increase: Start with a low rank and gradually increase it during training. This approach, known as "rank-relaxation," allows the model to find a good low-rank approximation before fine-tuning with higher ranks.
- Monitor performance: Track accuracy and adjust the rank as needed. The optimal rank often depends on the task and dataset, requiring empirical tuning.

Implementing LoRa effectively requires careful consideration of these factors and an understanding of the trade-offs between model compression and performance. By following these guidelines and adapting them to specific use cases, practitioners can leverage LoRa to significantly reduce model size and computational requirements while maintaining or improving model performance.

Step	Complexity (1-10)	Time Requirement (hours)
Layer Identification	3	2
Matrix Decomposition	7	5
Initialization	5	3
Forward Pass Modification	4	4
Learning Rate Adjustment	6	8
Regularization	5	6
Gradual Rank Increase	8	12
Performance Monitoring	4	10

Table 2: Breakdown of Low-Rank Adaptation Process in Deep Learning Models [7]

Advantages and Trade-Offs of LoRa

Low-rank adaptation (LoRa) has emerged as a powerful optimization technique in deep learning, offering several unique advantages over other methods. However, like any optimization approach, it also comes with certain trade-offs that practitioners should consider.

Advantages:

1. Fine-grained control: One of LoRa's key strengths is its ability to provide fine-grained control over the optimization process. The rank can be adjusted per-layer basis, allowing for targeted optimization of specific parts of the neural network. This granular approach enables researchers and engineers to focus on the layers that benefit most from low-rank approximation, potentially leading to more efficient and effective model compression [8].
2. Compatibility: LoRa demonstrates excellent synergy with various model optimization techniques. It can be effectively combined with model compression methods such as pruning, quantization, and weight compression. When applied in conjunction with LoRa, these techniques can lead to even greater model efficiency without significant performance loss. Furthermore, LoRa enhances the process of knowledge distillation, where a smaller model (e.g., Llama 7B) is fine-tuned using prompt/response pairs from a larger foundational model. LoRa's unique advantage lies in its ability to facilitate knowledge distillation while simultaneously allowing for model compression through the techniques above. This versatility enables the development of highly optimized models that leverage multiple efficiency strategies concurrently, resulting in compact yet powerful AI systems.

3. Preservation of model structure: Unlike pruning techniques that remove connections or neurons, LoRa maintains the overall architecture of the model. This structure preservation can be particularly beneficial when working with pre-trained models or when model interpretability is important. It allows for easier integration of the optimized model into existing pipelines and facilitates comparison with the original, uncompressed model.

However, practitioners should be aware of potential trade-offs:

1. Increased training complexity: Training models with low-rank representations may require careful tuning of hyperparameters. The introduction of additional matrices (A and B in the LoRa formulation) can increase the complexity of the optimization landscape. Researchers have found that techniques such as gradual rank increase or specialized learning rate schedules may be necessary to achieve optimal results. This increased complexity can lead to longer training times and require more expertise to implement effectively.
2. Possible accuracy loss: While LoRa often maintains model performance, aggressive rank reduction can lead to performance degradation. The relationship between rank reduction and accuracy is not always linear, and finding the optimal rank for each layer can be challenging. In some cases, the low-rank approximation may not capture all the important features learned by the original model, leading to a drop in accuracy on certain tasks or datasets.
3. Limited applicability: Not all layers in a neural network benefit equally from low-rank approximation. Layers with inherently low-rank structure or those that capture critical high-dimensional features may not be suitable candidates for LoRa. For example, early convolutional layers in image classification networks often learn low-level features that are difficult to approximate with low-rank matrices without significant information loss [9]. Therefore, the effectiveness of LoRa can vary depending on the specific architecture and task at hand.

Despite these trade-offs, LoRa has demonstrated significant potential in optimizing large neural networks, particularly in natural language processing. Its ability to dramatically reduce the number of parameters while maintaining performance makes it a valuable tool in the ongoing efforts to deploy powerful AI models in resource-constrained environments.

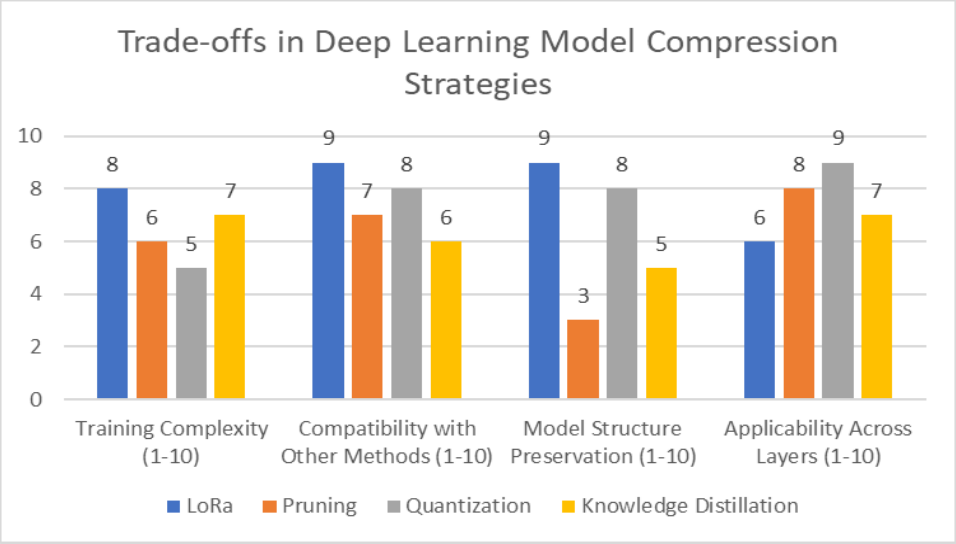

Fig. 2: Comparison of Model Optimization Techniques: LoRa vs Traditional Methods [8, 9]

CASE STUDIES AND APPLICATIONS

Low-rank adaptation (LoRa) has demonstrated remarkable versatility and effectiveness across various domains in deep learning, showcasing its potential to optimize complex models while maintaining performance. Here, we explore some notable case studies and applications of LoRa in different fields:

1. Natural Language Processing (NLP):

LoRa has shown auspicious results in optimizing large language models, which are at the forefront of NLP research. In a groundbreaking study by Hu [5], LoRa was applied to the BERT-large model, a transformer-based architecture widely used for various NLP tasks. The researchers found that LoRa could reduce the number of trainable parameters by 99.5% while achieving performance comparable to full fine-tuning on tasks such as question answering and natural language inference.

For machine translation, LoRa has been used to adapt pre-trained models to specific language pairs efficiently. This approach allows for rapid deployment of translation systems for low-resource languages without extensive retraining of the entire model.

In text classification tasks, LoRa has enabled fine-tuning of large language models for specific domains with minimal additional parameters. This has proven especially valuable in limited labeled data scenarios, as the low-rank adaptation can effectively capture domain-specific features without overfitting.

2. Computer Vision:

While initially developed for NLP applications, LoRa has also shown potential in optimizing Convolutional Neural Networks (CNNs) for image classification tasks. The technique has been particularly effective in compressing the fully connected layers, which are present in most vision models based on Convolutional Neural Networks (CNNs), such as ResNet, MobileNet, and others. These layers often contain a significant portion of the model's parameters, making them prime targets for optimization.

Recent studies have applied LoRa to various CNN architectures for image classification tasks. Researchers have found that LoRa can significantly reduce model size while maintaining accuracy close to that of the original model. This compression is achieved by applying low-rank approximations to both the convolutional and fully connected layers, with the latter often benefiting the most from the technique.

3. Speech Recognition:

LoRa has been used in speech recognition to compress Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, commonly employed for sequence modeling tasks. These optimizations have significant implications for deploying speech recognition systems with limited computational resources on mobile and edge devices.

A major tech company demonstrated a notable application of LoRa in speech recognition in a project. They used the technique to adapt a large, pre-trained acoustic model to specific accents and dialects. This approach allowed for rapid customization of the speech recognition system without extensive retraining, resulting in improved accuracy for diverse user groups.

Furthermore, LoRa has been applied to compress LSTM-based language models used in speech recognition systems. By reducing the parameter count in these models, researchers have achieved faster inference times on mobile devices without sacrificing recognition accuracy.

These case studies highlight the broad applicability and effectiveness of LoRa across different domains of deep learning. The technique's ability to significantly reduce model size while maintaining performance makes it valuable for deploying sophisticated AI models in resource-constrained environments, such as mobile devices, edge computing platforms, and IoT devices.

As research in this area continues to evolve, we expect to see further refinements and novel applications of LoRa, potentially opening up new possibilities for efficient, adaptive AI systems across a wide range of industries and use cases. For instance, in mobile computing, techniques like LoRa can be combined with efficient network architectures such as ShuffleNet [10] to create highly optimized models for deployment on resource-constrained devices.

ADVANCED TOPICS AND EXTENSIONS

As Low-Rank Adaptation (LoRa) continues to gain traction in the deep learning community, researchers are exploring several exciting directions to enhance its capabilities and broaden its applicability. These advanced topics and extensions promise to push the boundaries of model optimization and efficiency even further.

1. Dynamic rank selection:

One of the most promising avenues of research is the development of algorithms for automatically determining the optimal rank for each layer during training. Currently, the rank in LoRa is typically set manually or through extensive hyperparameter tuning. Dynamic rank selection aims to automate this process, potentially leading to more efficient and effective model compression.

Recent work by Cai [11] has proposed an adaptive rank selection method that dynamically adjusts the rank of each layer based on its importance to the overall model performance. Their approach, called AdaRank, uses a differentiable rank selection mechanism that allows the model to learn the optimal rank for each layer during training. Initial results showed that this dynamic approach could achieve better compression ratios than static rank selection while maintaining comparable performance across various tasks, including machine translation and image classification.

2. Hybrid approaches:

Combining LoRa with other optimization techniques such as quantization or pruning has shown great promise for achieving even greater model compression. These hybrid approaches leverage the strengths of multiple methods to create highly efficient models suitable for deployment in resource-constrained environments.

For instance, a study by Liu [12] introduced a framework that integrates LoRa with structured pruning and quantization. Their method, called "LoRaP," first applies LoRa to reduce the rank of weight matrices, then uses structured pruning to remove less important neurons or filters, and finally quantizes the remaining parameters.

The researchers demonstrated that this hybrid approach could achieve up to 98% compression on large language models while maintaining performance within 1% of the original model on various NLP tasks.

3. Hardware-aware LoRa:

Another area of active research is tailoring low-rank representations to specific hardware architectures. This approach aims to optimize the performance of LoRa-compressed models on particular hardware platforms, such as mobile devices, FPGAs, or specialized AI accelerators.

Recent work in this direction has focused on developing hardware-aware LoRa implementations that consider target hardware's specific constraints and capabilities. For example, researchers have explored techniques to align the low-rank matrices with the memory access patterns and computational units of specific processors, leading to improved inference speed and energy efficiency.

Additionally, efforts are being made to integrate LoRa with neural architecture search (NAS) techniques to automatically discover low-rank and hardware-efficient model architectures. This combination of LoRa and hardware-aware NAS holds the potential to create highly optimized models tailored for specific deployment scenarios.

These advanced topics and extensions demonstrate the ongoing evolution and potential of LoRa as a powerful tool for model optimization. As research in these areas progresses, we can expect to see even more efficient and versatile applications of LoRa across various domains of deep learning and artificial intelligence.

CONCLUSION

In conclusion, Low-Rank Adaptation (LoRa) presents a powerful and versatile approach to optimizing deep learning models, addressing critical challenges in computational efficiency and memory usage. Through its ability to dramatically reduce the number of trainable parameters while maintaining or even enhancing performance, LoRa has demonstrated significant potential across various domains, particularly in natural language processing. The technique's compatibility with other optimization methods and its preservation of model structure make it a valuable tool in the ongoing efforts to deploy sophisticated AI models in resource-constrained environments. As research in this area continues to evolve, with promising developments in dynamic rank selection, hybrid approaches, and hardware-aware implementations, LoRa is poised to play an increasingly important role in making large, powerful models more accessible and deployable in real-world scenarios. By enabling efficient adaptation of pre-trained models and opening up new possibilities for transfer learning and domain-specific applications, LoRa contributes significantly to the advancement of more efficient and accessible AI systems, paving the way for broader adoption and innovation in the field of deep learning.

REFERENCES

[1] T. B. Brown, "Language Models are Few-Shot Learners," arXiv:2005.14165 [cs.CL], 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[2] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-Rank Adaptation of Large Language Models," arXiv:2106.09685 [cs.CL], 2021. [Online]. Available: https://arxiv.org/abs/2106.09685

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language

Technologies, Volume 1 (Long and Short Papers), 2019, pp. 4171-4186. [Online]. Available: https://www.aclweb.org/anthology/N19-1423/

[4] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," arXiv:1510.00149 [cs.CV], 2015. [Online]. Available: https://arxiv.org/abs/1510.00149

[5] E. J. Hu, "LoRA: Low-Rank Adaptation of Large Language Models," arXiv:2106.09685 [cs.CL], 2021. [Online]. Available: https://arxiv.org/abs/2106.09685

[6] M. Udell and A. Townsend, "Why Are Big Data Matrices Approximately Low Rank?," SIAM Journal on Mathematics of Data Science, vol. 1, no. 1, pp. 144-160, 2019. [Online]. Available: https://epubs.siam.org/doi/10.1137/18M1183480

[7] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," Psychometrika, vol. 1, no. 3, pp. 211-218, 1936. [Online]. Available: https://link.springer.com/article/10.1007/BF02288367

[8] J. Frankle and M. Carbin, "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks," in International Conference on Learning Representations, 2019. [Online]. Available: https://openreview.net/forum?id=rJl-b3RcF7

[9] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal Brain Damage," in Advances in Neural Information Processing Systems, 1990, pp. 598-605. [Online]. Available: https://proceedings.neurips.cc/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf

[10] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 6848-6856. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2018/html/Zhang_ShuffleNet_An_Extremely_CVPR_2018_paper.html

[11] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-All: Train One Network and Specialize it for Efficient Deployment," in International Conference on Learning Representations, 2020. [Online]. Available: https://openreview.net/forum?id=HylxE1HKwS

[12] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the Value of Network Pruning," in International Conference on Learning Representations, 2019. [Online]. Available: https://openreview.net/forum?id=rJlnB3C5Ym

Citation: Subash Patel, Low-Rank Adaptation (LoRa): Revolutionizing Model Optimization in Deep Learning, International Journal of Computer Engineering and Technology (IJCET), 15(4), 2024, pp. 537-548

Abstract Link: https://iaeme.com/Home/article_id/IJCET_15_04_047

Article Link:
https://iaeme.com/MasterAdmin/Journal_uploads/IJCET/VOLUME_15_ISSUE_4/IJCET_15_04_047.pdf

✉ **editor@iaeme.com**