AARHUS
UNIVERSITY
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Bridging the Gap

Integrating Linear Algebra in the Development and Understanding of Large Language Models for Software Engineering Applications.

by

Asger Poulsen

202106630

Bachelor's Thesis

in

Computer Engineering

Supervisor: Hugo Daniel Macedo

Aarhus University, Department of Electrical and Computer Engineering

8 June 2024

Preface

This bachelor's thesis was written for the Department of Electrical and Computer Engineering, Aarhus University. It is part of the Computer Engineering study program, and was written in the spring of 2024.
All source files associated with this thesis are found at: `https://github.com/asgersong/BSc`

Asger Poulsen, February 26, 2024

Abstract

Acknowledgements

Contents

Chapter 1

Introduction

Large Language Models (LLMs) have become a cornerstone in the field of natural language processing (NLP) and artificial intelligence (AI), driving significant advancements and innovations. These models are designed to understand, generate, and interpret human language at a level that is increasingly indistinguishable from that of a human being. The development and evolution of LLMs mark a pivotal shift in how machines can learn from and interact with textual data, enabling a plethora of applications ranging from automated text generation to sophisticated conversational agents.

Chapter 2

Literature Review

2.1 Overview of Large Language Models

The evolution of LLMs can be traced back to earlier models of machine learning that attempted to process and understand language. However, it was the introduction of models like Google's BERT (Bidirectional Encoder Representations from Transformers) and OpenAI's GPT (Generative Pre-trained Transformer) series that marked a significant leap in the capabilities of language models. Each iteration of these models has brought about improvements in understanding context, generating text, and general language comprehension, culminating in state-of-the-art models that are capable of writing essays, composing poetry, and even generating code.

2.2 Linear Algebra in Machine Learning

2.3 Previous Studies on LLM Compression and Optimization

Chapter 3

Theoretical Foundations

3.1 Linear Algebra in Deep Learning

3.2 Understanding LLMs

Chapter 4

Methodology

4.1 Educational Synergy Development

4.2 Software Engineering Application

4.3 Evaluation Method

Optimizing Large Language Models (LLMs) for efficiency and performance without compromising their effectiveness is a critical area of research in the field of artificial intelligence. Various techniques have been developed to address this challenge, each employing unique strategies to reduce computational resources, decrease model size, and maintain, if not improve, the model's performance. This section explores the general methodologies applied in the optimization of LLMs, focusing particularly on model compression techniques. Among these, Low-Rank Adaptation (LoRA) stands out as a significant innovation, offering a balance between model efficiency and task performance.

4.3.1 General Optimizations for LLMs

Optimization techniques for LLMs can be broadly categorized into two: model pruning and parameter sharing. Model pruning involves systematically removing parameters or connections within the model that contribute the least to its output, thereby reducing its size and complexity. Parameter sharing, on the other hand, reuses the model's parameters across different parts of the model or across different tasks, efficiently leveraging the model's capacity.

4.3.2 Low-Rank Adaptation (LoRA)

Low-Rank Adaptation (LoRA) introduces an efficient technique for adapting large pre-trained models like GPT-3 to specific tasks without the need for extensive retraining of all model parameters. This approach leverages the observation that despite the high parameter count in large neural models, their effective operational space often exhibits a significantly lower intrinsic dimensionality.

Theoretical Foundation At the core of LoRA is the adaptation of weight matrices $W \in \mathbb{R}^{d \times d}$ through the introduction of low-rank matrices $A \in \mathbb{R}^{d \times r}$ and

$B \in \mathbb{R}^{r \times d}$, where $r \ll d$. This results in the adapted weight matrix W' being represented as:

$$W' = W + BA \tag{4.1}$$

Here, r denotes the rank and serves as a crucial parameter that balances the efficiency and expressiveness of the adaptation. This formulation ensures that the pre-trained weights (W) remain unchanged, preserving the foundational knowledge acquired during pre-training, while A and B encapsulate the task-specific adjustments.

Advantages LoRA's methodology brings forth several advantages:

- **Parameter Efficiency:** By optimizing the low-rank matrices A and B, LoRA significantly reduces the number of trainable parameters, leading to efficient storage and faster adaptation processes.
- **Preservation of Pre-trained Knowledge:** The approach ensures that the valuable knowledge captured in the pre-trained model is retained, making it particularly beneficial for adapting computationally expensive models like GPT-3.
- **Flexibility and Scalability:** LoRA's adaptive process is highly scalable and flexible, making it possible to adapt large models to various tasks with minimal computational overhead.

Practical Implementation Implementing LoRA involves selecting the transformer layers most relevant to the target task and applying the low-rank adaptations. The process requires initializing A and B, choosing an appropriate rank r, and training these matrices while keeping the rest of the model parameters fixed. This strategy allows for the efficient adaptation of large-scale models to specific tasks, significantly reducing the need for computational resources compared to traditional full model retraining.

4.3.3 Metrics for Evaluation

Evaluating the effectiveness of optimization techniques like LoRA involves several key metrics:

Model Size Reduction: A primary metric is the reduction in the total number of parameters, indicating the efficiency of the compression technique.

Inference Speed: The impact on the model's inference latency is critical, especially for applications requiring real-time responses.

Performance Retention: The maintenance of task-specific performance, measured through metrics such as accuracy, F1 score, or ROUGE scores for language tasks, is essential to ensure the utility of the compressed model.

Computational Efficiency: The reduction in computational resources required for training and inference reflects the practical benefits of the optimization technique.

These metrics provide a comprehensive framework for assessing the trade-offs involved in LLM optimization, guiding the development and application of techniques like LoRA in enhancing the practical utility of state-of-the-art language models.

Chapter 5

Implementation

5.1 Curriculum Component Implementation

5.2 Chatbot Development

5.2.1 Tools and Libraries Used

5.2.2 Integration of LLM

Chapter 6

Evaluation and Results

6.1 Curriculum Effectiveness

6.2 Chatbot Performance and Usefulness

6.3 Compression and Optimization of LLM

6.3.1 Methodology Applied

6.3.2 Results and Analysis

Chapter 7

Discussion

7.1 Interpretation of Results

7.2 Theoretical and Practical Implications

7.3 Limitations and Challenges

Chapter 8

Conclusion and Future Work

8.1 Summary of Key Findings

8.2 Contributions to the Field

8.3 Recommendations for Future Research