Master Thesis

A Study on PEFT Methods Applied to Quantized Large Language Models

Sharan Shyamsundar
(Matriculation Number 1870221)

September 3, 2024

Submitted to

Data and Web Science Group
Prof. Dr. Simone Paolo Ponzetto
University of Mannheim

Abstract

This thesis explores the integration of Quantization and Parameter-Efficient Fine-Tuning (PEFT) methods in optimizing Large Language Models (LLMs). While Quantization and PEFT have independently advanced the efficiency of LLMs, their combined application remains underexplored. This thesis seeks to optimize LLMs through applying PEFT methods to quantized models, aiming to replicate QLoRA's (Dettmers et al. 2024) success and assess its effectiveness and performance impact across different PEFT strategies and LLMs. The experiments examine various combinations of these techniques, focusing on their impact on model performance, memory efficiency, and computational overhead. Using the LLaMa series as the base models, experiments were conducted across diverse Natural Language Generation tasks to assess these combinations. The results reveal that certain Quantization and PEFT pairings offer substantial improvements in efficiency without significant loss of performance, highlighting the potential for these methods to enable more practical deployment of large models in resource-constrained environments.

Contents

Abstract ii

1. Introduction 1
- 1.1. Problem Statement . . . 2
- 1.2. Contribution . . . 3
- 1.3. Thesis Structure . . . 3

2. Background 4
- 2.1. Language Models . . . 4
- 2.2. Transformers . . . 7
- 2.3. Large Language Models . . . 9
- 2.4. Transfer Learning . . . 10
- 2.5. Parameter-Efficient Fine-Tuning (PEFT) . . . 11
  - 2.5.1. Adapter-Based Methods . . . 12
  - 2.5.2. Token-Based Methods . . . 20
  - 2.5.3. Other PEFT Methods . . . 23
- 2.6. Quantization . . . 27
  - 2.6.1. Quantization Aware Training . . . 28
  - 2.6.2. Post Training Quantization . . . 32
- 2.7. Summary . . . 37

3. Efficiently Fine-Tuning Quantized Models 38
- 3.1. QLoRA . . . 41
  - 3.1.1. Quantization in QLoRA . . . 42
  - 3.1.2. Low Rank Adapters . . . 44
  - 3.1.3. Combination of Quantization and LoRA . . . 45
  - 3.1.4. Advantages of QLoRA . . . 46
  - 3.1.5. Implementation Considerations . . . 46
- 3.2. Rank Stabilized LoRA . . . 46
- 3.3. LoftQ . . . 47
- 3.4. IA3 . . . 48
- 3.5. Weight-Decomposed Low-Rank Adaptation . . . 49
- 3.6. Half Quadratic Quantization . . . 51
- 3.7. Summary . . . 51

4. Experimental Study **52**
4.1. Goal of the Experiments . . . 52
4.2. Datasets used for Fine-tuning . . . 53
4.3. LLMs used in the experiments . . . 54
4.4. Implementation Details . . . 54
4.5. Evaluation Criteria . . . 55
4.6. Experimental Setup . . . 56
4.6.1. Combination 1 - QLoRA & variations . . . 56
4.6.2. Combination 2 - QIA3 . . . 58
4.6.3. Combination 3 - QDoRA . . . 59
4.6.4. Combination 4 - HQQ-LoRA . . . 60
4.7. Results . . . 61
4.7.1. Quantization Effect . . . 61
4.7.2. Trainable Parameters . . . 63
4.7.3. PEFT Memory Overhead . . . 64
4.7.4. Evaluation Perplexity . . . 66
4.7.5. Downstream Task - MMLU . . . 68
4.7.6. Memory Footprint . . . 71
4.7.7. Average Runtime . . . 72
4.7.8. Summary . . . 73

5. Conclusion **74**
5.1. Summary . . . 74
5.2. Future Work . . . 75

Bibliography **77**

A. Program Code / Resources **87**

B. Additional Experimental Results **88**
B.1. Hyperparameter Settings . . . 88
B.2. Detailed Results . . . 89

Ehrenwörtliche Erklärung **93**

1. Introduction

Recently, Large Language Models (LLMs), such as the GPT (Achiam et al. 2023), LLaMa (Touvron et al. 2023), and Gemma (Team et al. 2024) series, have achieved significant prominence and have become deeply integrated into various aspects of human life, representing a major advancement in Artificial Intelligence. LLMs approach any Natural Language Processing (NLP) task as a text generation challenge, producing the desired output by conditioning on both the input and the text generated so far. LLMs have set new benchmarks in task generalization across a wide range of applications, including question answering, language translation, and summarization. These models demonstrate the ability to learn and adapt through in-context task descriptions or few-shot examples (Min et al. 2022), making them versatile across different domains. This efficiency makes LLMs valuable tools in various domains, enhancing productivity and decision-making. These models, rooted in Deep Learning (DL), are built on the transformer architecture (Vaswani et al. 2017) and have been trained on vast amounts of text data. This training of LLMs on large corpora of text is known as pre-training. It enables the ability of LLMs to understand context and nuances and enables them to proficiently handle diverse tasks as mentioned above.

Earlier studies have suggested that LLMs exhibit high levels of generalization, enabling them to apply their acquired knowledge to new tasks not included in their original training (Brown et al. 2020). Nevertheless, to focus on more niche domains, LLMs can again be trained on domain-specific datasets. This is known as fine-tuning. Fine-tuning plays a crucial role in refining LLMs, enabling them to align more closely with desired behaviors, such as following instructions accurately (Ouyang et al. 2022) or adhering to specific principles. By systematically exposing LLMs to extensive collections of task-specific instruction datasets, fine-tuning not only sharpens their instruction-following capabilities but also enhances their scaling curve, leading to notable performance improvements on various unseen downstream tasks (Wei et al. 2022) (Sanh et al. 2022) (Scialom et al. 2022).

The development and popularity of LLMs have made it crucial to fine-tune them for different tasks. However, fine-tuning LLMs presents several significant challenges, particularly concerning computational resources, memory requirements, and scalability. LLMs consist of billions or trillions of parameters, making the process of fine-tuning both computationally expensive and memory-intensive. This high resource demand often exceeds the capabilities of standard hardware, necessitating the use of specialized

infrastructure. Lastly, fine-tuning often involves updating a large portion of the model's parameters, which can create difficulties in maintaining efficiency during deployment, particularly in memory-constrained environments. These challenges underscore the need for more efficient fine-tuning methods, such as Parameter-Efficient Fine-Tuning (PEFT) and quantization, which aim to mitigate these issues while preserving or enhancing model performance (Ding et al. 2023) (Fu et al. 2023).

Several PEFT methods have been developed that update only a small fraction of the total parameters in LLMs compared to the full set of pre-trained weights (Houlsby et al. 2019) (Hu et al. 2021). These approaches significantly reduce the number of learnable parameters, making the fine-tuning of LLMs more practical by minimizing the memory footprint of the optimizer states. This reduction leads to lower memory usage during training and facilitates the efficient storage and transfer of task-specific fine-tuned parameters during deployment.

Nevertheless, LLMs still need a significant amount of memory even with PEFT, and model compression techniques allow for even further optimizations. For example, as highlighted by Hu et al. (Hu et al. 2021), the LoRA method can decrease the memory required for fine-tuning GPT-3 175B from 1.2TB to 350GB. Despite this reduction, the model still demands around 350GB of memory for storing parameters in half-precision format, indicating the need for additional strategies to further reduce memory consumption. Quantization has become a key strategy to address memory issues, especially while using parameter-efficient fine-tuning techniques (Dettmers et al. 2022a). Quantization is an effective technique for both compressing and accelerating neural networks by converting parameters into low-bit integers while preserving a shared high-precision scale within each parameter group. This approach reduces the model's memory footprint and computational demands, enabling more efficient deployment without significantly compromising performance (Jacob et al. 2018).

1.1. Problem Statement

The rapid advancements in LLMs have been transformative, enabling breakthroughs across various domains, from natural language understanding to content generation. However, the sheer size of these models, often comprising billions of parameters, poses significant challenges in terms of computational resources, storage, and deployment. To address these issues, researchers have developed two key strategies: Parameter-Efficient Fine-Tuning (PEFT) methods and quantization techniques.

While both PEFT and quantization have been extensively studied individually, there is a noticeable gap in the research exploring the intersection of these two approaches. Most studies focus on the application of PEFT to uncompressed, full-precision models,

or on quantization techniques applied to fully fine-tuned models. However, the combined application of PEFT methods to quantized LLMs remains relatively unexplored. The integration of Quantization and PEFT techniques is very crucial in this domain of fine-tuning LLMs. QLoRA (Dettmers et al. 2024) is the state-of-the-art work in this domain, introducing a quantization approach to LoRA, significantly enhancing the efficiency of fine-tuning. This combination is crucial for enabling the efficient deployment of large models in resource-constrained environments, such as edge devices or smaller data centers. Despite the potential benefits, the existing literature lacks comprehensive evaluations of how PEFT methods interact with different quantization schemes and how this combination affects model performance across various tasks.

1.2. Contribution

This thesis seeks to optimize LLMs by applying PEFT to quantized models, aiming to replicate QLoRA's success and assess its effectiveness and performance impact across different strategies and LLMs. The motivation behind these experiments is rooted from the the growing need for efficient tuning and deployment of LLMs in resource-constrained environments. It aims to fill the gap in research of combined application by conducting a detailed study on the application of PEFT methods to quantized LLMs. The research will evaluate multiple PEFT techniques across different quantization configurations, assessing their impact on model performance, memory usage, and computational efficiency. The primary objective of the experiments in the thesis is to evaluate the performance and memory efficiency of different PEFT methods when applied to quantized LLMs. The experiments encompass a variety of configurations and methodologies to provide a comprehensive analysis of the capabilities and trade-offs associated with each approach. By addressing this unexplored intersection, the study seeks to provide valuable insights that can guide the development of more efficient and scalable LLMs, ultimately contributing to the broader field of AI research.

1.3. Thesis Structure

The thesis is organized into four main chapters. It begins with an overview of related concepts and a comprehensive review of existing Quantization and PEFT techniques. The following chapter explores the application of PEFT to quantized models, detailing the methods used in the experiments. These experiments involve fine-tuning LLaMa models with various Quantization and PEFT combinations, followed by a critical evaluation based on factors such as quantization effects, trainable parameters, memory overhead, memory footprint, runtime, perplexity, and performance on downstream tasks. The thesis concludes with a summary of findings and a discussion of future research directions.

2. Background

This chapter provides a comprehensive overview of key concepts related to the thesis. It begins with an exploration of language models and their various architectures in deep learning. The focus then shifts to the Transformer architecture and the development of large language models. The chapter also covers the concepts of pre-training and fine-tuning, setting the stage for a discussion on Parameter-Efficient Fine-Tuning (PEFT) methods. This is followed by an examination of quantization techniques. The chapter concludes with a summary.

2.1. Language Models

Language modeling (LM) employs a range of statistical and probabilistic methods to estimate the likelihood of specific word sequences within a sentence. By analyzing textual data, language models can predict the probability of subsequent tokens, allowing them to assess whether these sequences conform to typical grammatical patterns. At its core, it calculates the probability of a word or sequence of words appearing in a sentence. This is often achieved through the use of n-grams, which are continuous sequences of n words from a given sample of text. For example, in a bigram model (where n=2), the model would consider pairs of consecutive words to predict the likelihood of a word following another (Brown et al. 1992). By doing so, statistical language models can generate coherent text, perform tasks like text completion, and assist in various natural language processing applications (Chen and Goodman 1999).

The probabilistic language model computes a probability distribution of a sentence (s) of sequences of n words i.e.

$$\begin{aligned} p(s) &= p(w_1, w_2, ..., w_n) \\ &= p(w_1)p(w_2|w_1)p(w_3|w_1, w_2).....p(w_n|w_1, ...w_{n-1}) \\ &= \prod_{i=1}^{n} p(w_i|w_1, ..., w_{i-1}) \end{aligned} \tag{2.1}$$

or compute the probability of an upcoming word i.e.

$$p(w_t|w_1, w_2, ..., w_{t-1}) \tag{2.2}$$

Neural language models represent a significant advancement in natural language processing by utilizing neural networks to predict and generate text. Unlike traditional statistical models, neural language models leverage deep learning architectures, such as recurrent neural networks (RNNs) and transformers, to capture complex patterns and dependencies within language data (Mikolov et al. 2010) (Vaswani et al. 2017). Large text datasets are used to train these models, which enable them to acquire high-dimensional word representations called embeddings that capture syntactic and semantic links. Through the use of layers of nonlinear transformations, neural language models excel at generating coherent and contextually relevant text. This capability makes them exceptionally powerful for a range of applications, including machine translation, text summarization, and conversational agents (Devlin et al. 2018). While Feedforward neural networks (FNNs) are simpler, they are less effective at capturing long-range dependencies since they evaluate incoming data in a fixed-size context window (Bengio et al. 2000). To counter these limitations, Recurrent Neural Networks were introduced.

Recurrent neural networks (RNNs) have been pivotal in advancing language modelling due to their ability to handle sequential data and capture temporal dependencies. Because RNNs keep track of a hidden state that is updated every time step, they are especially well-suited for language applications where words' meanings are greatly influenced by their context. However, the limitations of ordinary RNNs include vanishing and exploding gradients, which make it difficult for them to learn long-term dependencies (Bengio et al. 1994). To address these issues, architectures like Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) were developed, offering methods to preserve information over longer periods (Hochreiter and Schmidhuber 1997) (Cho et al. 2014).

Convolutional neural networks (CNNs), traditionally used in image processing, have also been adapted for language modelling. CNNs utilize convolutional filters to extract hierarchical structures and local patterns from textual input. This approach allows CNNs to effectively handle varying lengths of input and capture n-gram-like features, making them useful for tasks such as text classification (Kim 2014). By stacking multiple convolutional layers, CNNs can capture increasingly complex patterns, providing a powerful alternative for certain natural language processing tasks (Grefenstette et al. 2014). Although CNNs have done a decent job in NLP tasks, they have a couple of drawbacks and hence have been unpopular in the domain. CNNs are designed to capture local spatial relationships, but they have difficulty modeling long-range dependencies in data. This can make them less effective for tasks that require an understanding of global context or relationships between distant parts of an input.

Although RNNs were the state-of-the-art architecture for NLP-related tasks. These models excelled at handling sequential data by maintaining a hidden state that could capture dependencies across time steps. However, RNNs struggled with long-range dependencies and required significant computational resources for training. The introduction of Pretrained Language Models (PLMs) marked a significant shift in NLP.

Pretrained language models have revolutionized natural language processing by leveraging large-scale corpora to learn rich linguistic representations before being fine-tuned for specific tasks. These models, such as BERT (Devlin et al. 2018), GPT (Radford et al. 2018), and RoBERTa (Liu et al. 2019), undergo a two-stage training process: pretraining on a diverse dataset to capture general language patterns, followed by fine-tuning on task-specific data. These models are based on transformers (Vaswani et al. 2017), self-supervised learning, and transfer learning, have been pre-trained on a plethora of text, and can now be adopted for any task even with a limited amount of data. By capturing contextual information and semantic nuances, pre-trained language models significantly reduce the need for task-specific labelled data and computational resources, making them highly efficient and versatile tools in the field of NLP. Masked Language Models (MLMs) and Causal Language Models (CLMs) are types of pre-trained language models.

Masked language Models (MLMs) are a type of Pre-trained Language Model which are trained by masking out certain words in a sentence and then predicting them based on the surrounding context.The advantage of MLMs is their ability to capture deep contextual relationships between words, enabling tasks like sentiment analysis, question answering, and named entity recognition with high accuracy. However, MLMs require large datasets and extensive computational resources for training, which may not be feasible for smaller organizations. Additionally, their bidirectional nature makes them unsuitable for text-generation tasks. One of the most famous examples is BERT. BERT's training approach is an advantage for understanding context, but it limits MLMs in terms of versatility in generating sequential text, where a unidirectional model might be more appropriate (Devlin et al. 2018) (Liu et al. 2019).

Causal Language Models (CLMs) or autoregressive models generate text sequentially by predicting the next word in a sentence based on the previous words. Unlike MLMs, CLMs generate text in a natural flow, making them ideal for tasks like text completion, dialogue generation, and creative writing. The main advantage of CLMs lies in their generative capabilities, which enable them to produce coherent and contextually relevant text. However, because they generate text based only on preceding words, they may not capture the full context as effectively as MLMs. Additionally, their propensity to produce plausible-sounding but incorrect information is a significant disadvantage, especially in applications requiring high accuracy. GPT series is a prominent example of CLM (Radford et al. 2019) (Brown et al. 2020).

Having covered the fundamentals of language models and their various types, the following section delves into the core component that underpins the current state-of-the-art in Language Models: Transformers. This section will explore the architecture, key mechanisms, and advancements that have made Transformers the foundation for many of today's most powerful models.

2.2. Transformers

Transformers (Vaswani et al. 2017) have fundamentally transformed the landscape of natural language processing (NLP) and are the core architecture behind many state-of-the-art pre-trained language models. Transformers eliminate the need for RNNs by employing a mechanism known as self-attention, which allows the model to weigh the importance of different words in a sentence relative to one another. With this method, transformers are better equipped to extract long-range dependencies and relationships from the data.

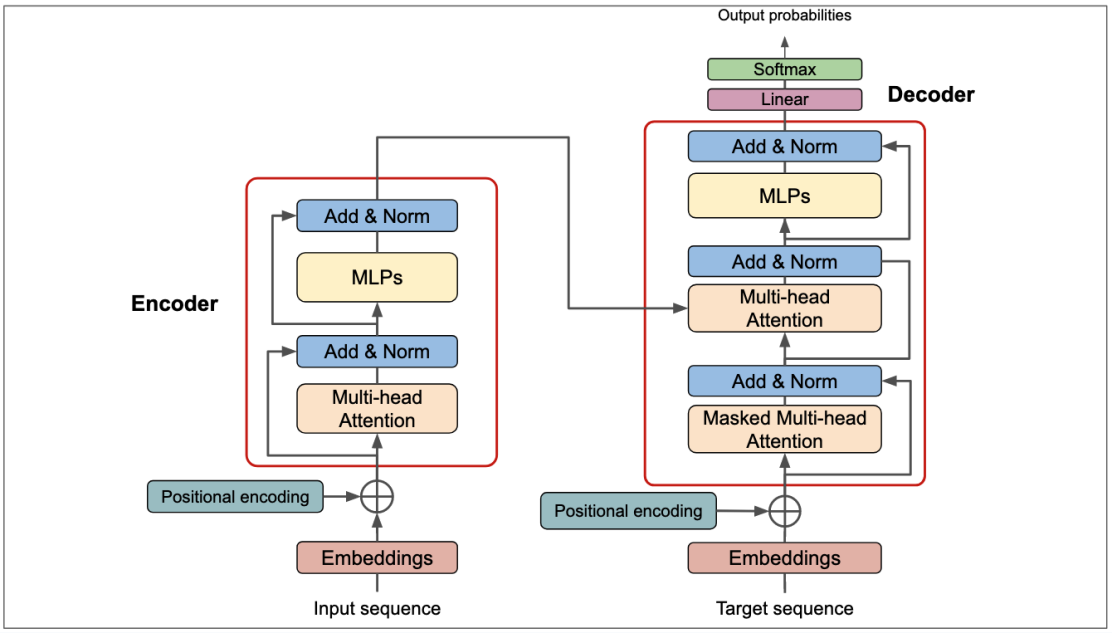

Figure 2.1.: The Transformer model architecture. Image adapted from (Vaswani et al. 2017)

The transformer architecture's self-attention mechanism is a key element. This process is essential for capturing contextual relationships within the data, enabling more effective language understanding. Each word in the input sequence is first converted into a vector representation. For each word in the sequence, three vectors are calculated through linear transformations of the input embeddings: the query vector (Q), the key vector (K), and the value vector (V).

$$Q = XW_Q, K = XW_K, V = XW_V \tag{2.3}$$

where X is the input embedding matrix and W_Q, W_K, W_V are the weight matrices for

query, key, and value, respectively.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.4}$$

where $\sqrt{d_k}$ is the dimension of the key vector k and query vector q .

By taking the dot product of the query vector of a single word and the key vectors of every word in the sequence, the attention scores are computed. This operation indicates how much focus each word should have on every other word. The attention scores are then used to compute a weighted sum of the value vectors that represent the output for each word, taking into account the importance of other words in the sequence.

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \tag{2.5}$$

where W^O is the weight matrix for the output and

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{2.6}$$

The transformer shown in Figure 2.1 consists mainly of two components: the encoder and the decoder. A series of token plus positional embeddings, with each element in the sequence representing a token, is fed into the encoder. The encoder then processes the input sequence using a stack of identical encoder layers to produce a contextualized vector representation for each input token. Each layer consists of multiple self-attention heads, followed by a feedforward neural network. Transformers employ multi-head attention to improve the model's capacity to focus on several portions of the sequence at once. After the multi-head attention mechanism, the output is passed through a feedforward neural network. Transformers also use layer normalization and residual connections to stabilize and improve training.

The last encoder layer's output is fed into the decoder as input. Like the encoder, the decoder is made up of a stack of layers that work together to produce complex, context-aware tokens and input representations. Masked multi-head self-attention, encoder-decoder attention, feed-forward neural networks, residual connections, and layer normalizations make up each decoder layer. The masked multiple attention heads allow them to attend to the previously generated tokens and not over the future tokens. The encoder-decoder attention mechanism enables the decoder to concentrate on relevant parts of the input sequence produced by the encoder layer to produce the output sequence. At the last layer of the decoder, a softmax activation plus a linear transformation provide the probability distribution of each token becoming the next word in the sequence.

2.3. Large Language Models

Large Language Models (LLMs) have emerged as pivotal components in the field of NLP, revolutionizing how machines understand and generate human language. LLMs are deep learning models trained on vast amounts of text data, enabling them to capture intricate patterns, semantics, and contextual nuances in language (Brown et al. 2020) (Radford et al. 2019). These models, such as OpenAI's GPT-4 (Achiam et al. 2023) and Google's BERT (Devlin et al. 2018), have set new benchmarks in various NLP tasks, including text generation, translation, summarization, and question answering.

The development of LLMs is grounded in the transformer architecture that allows LLMs to process entire sentences or documents at once, rather than sequentially, leading to more coherent and contextually accurate language understanding and generation.

The primary benefit of LLMs lies in their ability to generalize across a wide range of NLP tasks with minimal task-specific training. This generalization capability stems from the extensive pretraining phase, where models are exposed to diverse text corpora, learning a broad spectrum of language patterns and structures. As a result, LLMs can be fine-tuned for specific tasks with relatively small datasets, significantly reducing the time and resources required for model development.

Moreover, LLMs have demonstrated exceptional performance in generating human-like text. This has enabled advancements in applications such as conversational agents, automated content creation, and personalized recommendations. For instance, GPT-4 has been employed to draft emails, write code, and create poetry, showcasing its versatility and creative potential. Additionally, LLMs contribute to improved machine translation systems, where their ability to capture context leads to more accurate translations across multiple languages.

LLMs stand out due to their large parameter counts, which usually range from billions to trillions. As an example, GPT-3 (Brown et al. 2020) contains 175 billion parameters, although more recent models scale up to over a trillion parameters (Kaplan et al. 2020). This scale allows LLMs to capture a wide array of linguistic nuances and world knowledge, contributing to their ability to perform well across various tasks without task-specific training.

Despite their numerous benefits, the development and deployment of LLMs come with significant challenges, particularly related to the high computational and financial costs of training these models. Training an LLM involves processing terabytes of text data using sophisticated hardware, often necessitating the use of powerful GPUs. This process can span several weeks or even months, depending on the model's size and the computational infrastructure available. For example, training GPT-3 was estimated to involve thousands of GPUs running for several weeks, consuming a huge amount

of computing power (Narayanan et al. 2021). The financial cost associated with such extensive training is substantial, with estimates suggesting that training GPT-3 might have cost several million dollars.

Efforts are being made to mitigate these impacts by improving the efficiency of training algorithms and developing more energy-efficient hardware. However, the trade-off between model performance and sustainability remains a pressing issue in the field.

2.4. Transfer Learning

One of the most important techniques used in deep learning is transfer learning. Transfer learning is a machine learning technique where a model developed for a particular task is reused as the starting point for a model on a second task. This approach is particularly useful when the second task has limited labeled data but is related to the first task, which typically has a large amount of labeled data (Pan and Yang 2009). It's made up of two components, pretraining and fine-tuning.

The core idea behind transfer learning is that certain features learned by a model on a large dataset can be transferred and used for different but related tasks. For instance, a model trained on a large image dataset to recognize everyday objects can be adapted to identify specific types of objects, such as medical anomalies in radiographs, with relatively few additional training examples. This reduces the need for extensive labeled data and computational resources for the new task.

LLMs have revolutionized NLP through transfer learning. These models are pretrained on vast corpora of text data, learning to predict words in a sentence, which captures a wide range of linguistic features and knowledge. Then, these models can be fine-tuned on specific downstream tasks such as sentiment analysis, question answering, or named entity recognition, often with a small amount of task-specific data.

Pre-training involves training a model on a large dataset to learn general features. In the context of LLMs, this usually means training on a diverse and extensive text corpus to understand language patterns, semantics, and syntax. The objective is to create a robust model that has captured the general structure and nuances of language. The starting point is the model architecture, and all of the weights of the parameters are random. The model has no knowledge of language, and you then pre-train the model. This pretraining piece is very resource-heavy. Loads of data are required, and this could include the entire Wikipedia corpus and a wide range of other corpora. Huge computing power is another requirement. This is normally several hundreds to thousands of hardware accelerators, depending on how quickly you want to train your model. And these hardware accelerators are usually Nvidia's GPUs or Google's TPUs. At the end of

this training, which can take days, weeks, or months, you have a model that has a very good understanding of the language you've trained it on.

For example, BERT is pretrained using a masked language model (MLM) and next sentence prediction (NSP) tasks. In MLM, random words in a sentence are masked, and the model attempts to predict these masked words. In NSP, the model predicts if a given pair of sentences follow each other in the original text, helping the model understand sentence relationships (Devlin et al. 2018) (Radford et al. 2018).

Fine-tuning is the process of taking a pretrained model and training it further on a specific task with a relatively smaller dataset. This step adjusts the model's parameters minimally to specialize it for the new task, leveraging the general knowledge obtained during pretraining.

In the context of LLMs, fine-tuning involves training the model on a specific NLP task, such as sentiment analysis or machine translation. During fine-tuning, the model parameters are updated to improve performance on the target task, but the changes are typically small compared to the pretraining phase. This allows the model to retain its broad linguistic knowledge while adapting to the specific nuances of the task at hand (Howard and Ruder 2018) (Sun et al. 2019). Transfer learning, through pretraining and fine-tuning, has become a cornerstone of modern machine learning, particularly in NLP with LLMs. This approach significantly reduces the time and computational resources required to develop high-performing models for a wide range of applications.

2.5. Parameter-Efficient Fine-Tuning (PEFT)

The rapid advancement of large language models (LLMs) has significantly impacted various domains in natural language processing (NLP), including machine translation, disease prediction, code generation for robotics, and chatbot assistance. LLMs, which have billions of parameters, present a lot of issues when it comes to memory usage, processing expenses, and environmental effects during the deployment and fine-tuning stages (Wei et al. 2022) (Ouyang et al. 2022) (Chang et al. 2024). In response to this need, the development of Parameter-Efficient Fine-Tuning (PEFT) approaches has become important, enabling significant reductions in the resources required for model fine-tuning while maintaining and sometimes even improving model performance.

Traditional fine-tuning involves updating all parameters of a model, which can be resource-intensive and impractical for very large models. PEFT methods address this challenge by selectively updating a small subset of parameters, thereby preserving the bulk of the pre-trained weights and drastically reducing the amount of memory and computational power required. These methods help overcome the practical difficulties

associated with dealing with large models that have billions of parameters, such as memory and processing limitations, overfitting risk, environmental effects, and the requirement for model customization (Ding et al. 2023) (Fu et al. 2023). Various methods under the umbrella of PEFT have been proposed, each with its unique mechanism for achieving parameter efficiency.

This section will explore various PEFT methods, focusing on those developed for LLMs but also extending to other domains such as computer vision. We will review innovative techniques like LoRA, Prompt Tuning, and DoRA, among others, highlighting their unique approaches to parameter efficiency and their impact on model performance. Through this comprehensive review, we aim to provide a broad understanding of the state-of-the-art in PEFT and its implications for future research and practical applications.

2.5.1. Adapter-Based Methods

Adapter-based PEFT methods enable efficient fine-tuning of large pre-trained models by introducing lightweight, task-specific modules while keeping most of the model parameters frozen. This subsection focuses on a selection of these techniques.

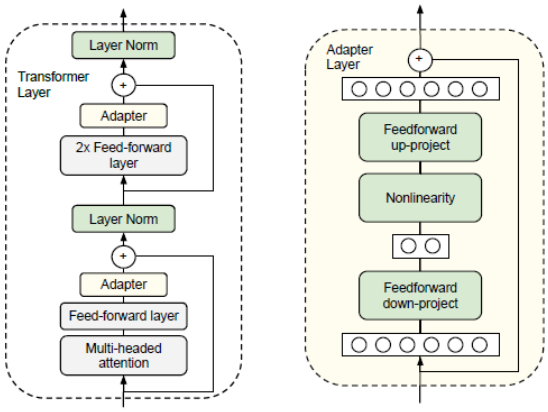

Figure 2.2.: Architecture of the adapter module and its integration with the Transformer. During adapter tuning, the green layers are trained on the downstream data, this includes the adapter, the layer normalization parameters, and the final classification layer (not shown in the figure). Image and caption adapted from (Houlsby et al. 2019).

One of the earliest and most influential PEFT methods is Adapter tuning, introduced

by Houlsby et al. (Houlsby et al. 2019). Adapter modules are small neural networks inserted within each layer of the pre-trained model as shown in figure 2.2, enabling task-specific fine-tuning without updating the entire model's parameters. This technique significantly reduces the number of trainable parameters, allowing efficient transfer learning with minimal computational resources. The primary benefit of this method lies in its ability to achieve comparable performance to full fine-tuning while only adjusting a fraction of the model's parameters. Specifically, adapters can reduce the number of trainable parameters by over 95%, resulting in faster training times and lower memory requirements. This makes it feasible to deploy and fine-tune LLMs in resource-constrained environments, facilitating broader accessibility and practical application of advanced NLP technologies.

Method	**Key Characteristic**
Adapter Tuning	Task Specific Layers inserted into Transformers for tuning
Compacter	Uses Low-Rank Factorization with hypercomplex multiplications
HyperFormer	Generates adapter parameters using a shared network that adjusts for each task
IA3	Uses learned scaling vectors, focusing on modifying only attention matrices
HyperX	Uses a single Hypernetwork that generates adapter parameters
LoRA	Introduces trainable Low-rank matrices while the rest of the model is frozen
DyLoRA	Dynamically adjusts LoRA across a range of ranks
O-LoRA	Uses Orthogonal Subspaces to prevent catastrophic forgetting
LoRAPrune	Combines LoRA with parameter pruning
RsLoRA	Improves LoRA by stabilizing gradient scaling, enabling higher-rank fine-tuning
NOLA	Represents model updates as Linear combination of low-rank random bases
LongLoRA	Extends LoRA for long context models using shifted sparse attention
VeRA	Leverages vector-based decomposition of random shared matrices
LoRAShear	Integrates structured pruning using dependency graphs and dynamic fine-tuning
DoRA	Enhances LoRA by decomposing pre-trained weights into magnitude and direction

Table 2.1.: Overview of Adapter Based PEFT methods

Mahabadi et al. presented a method called Compacter (Karimi Mahabadi et al. 2021), which significantly enhances the efficiency of fine-tuning large language models. Compacter combines ideas from adapters, low-rank optimization, and hypercomplex multiplication layers. It introduces task-specific weight matrices computed as the sum of Kronecker products between shared "slow" weights and "fast" rank-one matrices. This method trains only a fraction of the model's parameters, approximately 0.1%, while achieving performance on par with full fine-tuning on benchmarks like GLUE (General Language Understanding Evaluation) and SuperGLUE. Compacter's innovative approach enables substantial reductions in the number of trainable parameters and computational resources, making it highly efficient for adapting large-scale language models to new tasks.

Later the same year, Mahabadi et al. introduced another innovative approach to fine-tuning large language models for multiple tasks simultaneously. This method, known as HyperFormer (Karimi Mahabadi et al. 2021), employs shared hypernetworks to generate task-specific adapter parameters conditioned on task, adapter position, and layer ID. By leveraging these hypernetworks, HyperFormer efficiently shares information across tasks, significantly reducing the number of parameters needed for fine-tuning. This approach allows for positive transfer effects between related tasks while minimizing negative interference. Experiments on the GLUE benchmark demonstrate that HyperFormer not only outperforms traditional adapter-based methods but also achieves substantial parameter efficiency, making it a highly effective solution for multi-task learning in resource-constrained environments.

In their 2022 paper, Liu et al. critically evaluated the dominant methods in few-shot learning, contrasting PEFT techniques with in-context learning (ICL) (Liu et al. 2022). The authors present $(IA)^3$, a novel parameter-efficient fine-tuning method that scales intermediate activations with learned vectors. This approach significantly reduces the number of trainable parameters—by up to 10,000 times—while achieving superior performance. The authors also introduce T-Few, a recipe based on the T0 model, which integrates $(IA)^3$ and demonstrates exceptional performance on unseen tasks, surpassing ICL and even achieving super-human performance on the RAFT (Real-world Annotated Few-shot Tasks) benchmark. This method addresses the high computational and memory costs associated with ICL, offering a scalable and cost-effective alternative for adapting pre-trained models to new tasks with minimal data.

Hyper-X is a unified hypernetwork designed for efficient multi-task and multilingual transfer learning (Üstün et al. 2022). Hyper-X generates task and language-specific adapter parameters through a shared hypernetwork, which enables it to handle multiple tasks and languages simultaneously. This innovative approach conditions the hypernetwork on task, language, and layer embeddings, allowing it to adapt to new, unseen task-language pairs during inference. Hyper-X leverages masked language modeling (MLM) as an auxiliary task, enhancing its zero-shot transfer capabilities to languages not included during pre-training. The model demonstrates significant improvements over

traditional adapter-based methods, particularly in mixed-language multi-task settings, achieving competitive performance with fewer parameters and reduced training time. This makes Hyper-X a robust and scalable solution for deploying pre-trained models across diverse linguistic and task-specific contexts.

Low-rank Adaptation (LoRA) (Hu et al. 2021) is a PEFT method that modifies LLMs by adding a small subset of parameters that use low-rank matrices to capture the essential information. LoRA addresses the challenge of the prohibitive computational cost associated with fine-tuning all model parameters by freezing the original model weights and injecting trainable low-rank decomposition matrices into each layer of the Transformer architecture. A high-level overview of how LoRA looks is displayed in Figure 2.3. It operates on the principle of low-rank matrix approximation, a mathematical approach that represents a high-dimensional matrix using the product of two lower-dimensional matrices. LoRA focuses on adjusting the added low-rank matrices (A & B). The updated low-rank matrices are combined to reconstruct the full-weight matrices, incorporating task-specific knowledge into the model with minimal parameter updates. This process effectively captures the essential information required for the new task without the need to retrain the entire model, leading to significant reductions in computational resources and memory requirements. LoRA significantly reduces the number of trainable parameters required for fine-tuning. Specifically, it can lower the parameter count by a factor of up to 10,000 compared to traditional fine-tuning methods, while also reducing GPU memory requirements by a factor of three. This is achieved without compromising model performance. Empirical results demonstrate that LoRA can perform on par or even better than full fine-tuning across various models such as RoBERTa, DeBERTa, GPT-2, and GPT-3.

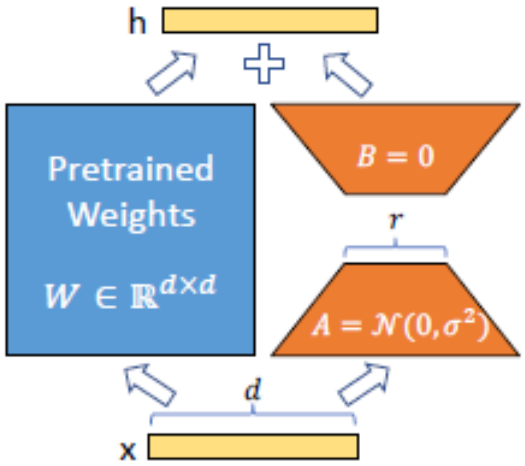

Figure 2.3.: Architecture of LoRA. The pretrained weights W are kept frozen and only the A and B matrices are updated during fine-tuning. Image adapted from (Hu et al. 2021).

One of the key advantages of LoRA is that it does not introduce additional inference latency, unlike adapter-based methods. This is because the low-rank updates can be precomputed and added to the frozen weights, allowing the model to operate at the same speed as a fully fine-tuned model during inference. Additionally, LoRA facilitates a more flexible and modular approach to model adaptation, enabling efficient task switching by simply applying different low-rank matrices for different tasks. The effectiveness of LoRA is rooted in its ability to leverage the low "intrinsic rank" of updates necessary for task adaptation. This insight allows LoRA to maintain the expressiveness of the original model while dramatically reducing the computational overhead. The authors provide a comprehensive analysis of the rank-deficiency in language model adaptations, which supports the efficacy of their approach. LoRA's practical benefits and robust performance make it a powerful tool for scaling large language models to diverse applications without incurring substantial computational costs.

DyLoRA, introduced in 2022, is a method designed to address the limitations of fixed-rank low-rank adapters in pre-trained language models (Valipour et al. 2022). Traditional LoRA techniques require selecting a specific rank for adaptation, a process that often necessitates exhaustive search and retraining for different rank configurations. DyLoRA overcomes this by dynamically adjusting the rank during training, eliminating the need for predefined ranks and enabling a single model to adapt across a range of ranks without retraining. This flexibility is achieved by sorting the representation learned at different ranks and using a stochastic sampling approach to update the model during training. Evaluations on tasks such as the GLUE benchmark and various natural

language generation tasks show that DyLoRA can achieve comparable performance to traditional LoRA methods while significantly reducing training time and computational overhead. By enabling efficient and flexible rank adaptation, DyLoRA presents a robust solution for parameter-efficient tuning in resource-constrained environments.

Xiao Wang and colleagues proposed Orthogonal Low-Rank Adaptation (O-LoRA) (Wang et al. 2023), a method aimed at addressing catastrophic forgetting in continual learning for LLMs. O-LoRA leverages the LoRA framework, adapting new tasks within orthogonal subspaces to the subspaces of previous tasks. This orthogonal constraint ensures minimal interference between tasks, preserving learned knowledge while enabling efficient new task learning. Unlike other methods requiring data storage or extensive parameter updates, O-LoRA achieves high efficiency with marginal additional parameter costs. Experiments on standard benchmarks demonstrate that O-LoRA outperforms existing continual learning methods, maintaining robust performance and generalization on unseen tasks.

Zhang et al. introduced LoRAPrune (Zhang et al. 2024), a method combining structured pruning and LoRA for efficient fine-tuning of large pre-trained models. LoRAPrune uses LoRA-guided pruning to reduce memory and computational costs while maintaining high performance. This approach prunes redundant channels and heads, optimizing model size and complexity. Experiments on models like LLaMA showed that LoRAPrune significantly reduces memory overhead and outperforms traditional methods in perplexity reduction on various datasets.

In 2023, Kalajdzievski introduced a modification to the LoRA method called rank-stabilized LoRA (rsLoRA) (Kalajdzievski 2023). This technique addresses the issue of gradient collapse in standard LoRA when higher-rank adapters are used. By dividing LoRA adapters by the square root of their rank rather than the rank itself, rsLoRA ensures stable learning and improved performance with larger ranks. Experimental results indicate that rsLoRA outperforms traditional LoRA, particularly in scenarios where increased computational resources are available for fine-tuning. This advancement allows for more efficient and effective adaptation of large language models without compromising on inference speed or performance.

Koohpayegani and colleagues introduced NOLA (Networks as Linear Combination of Low-Rank Random Basis), a method designed to enhance the parameter efficiency of LoRA by overcoming its inherent limitations (Koohpayegani, Navaneet, Nooralinejad, Kolouri, and Pirsiavash Koohpayegani et al.). NOLA reparameterizes the low-rank matrices in LoRA using linear combinations of randomly generated matrices (basis). This approach decouples the number of trainable parameters from both the rank and network architecture, enabling higher compression ratios while maintaining performance. By optimizing only the linear mixture coefficients of these random matrices, NOLA achieves significant parameter reductions and demonstrates superior results on both natural language and computer vision tasks. The method's efficiency is highlighted

in its ability to halve the parameters of larger models compared to LoRA, without sacrificing performance, thereby offering a scalable and efficient solution for fine-tuning large pre-trained models.

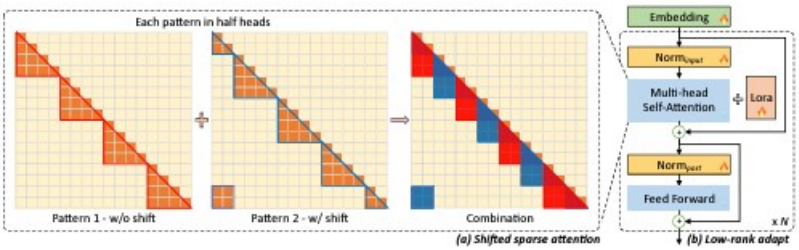

Figure 2.4.: Overview of LongLoRA. Introduces Shifted Sparse Attention during fine-tuning. In addition to training LoRA weights in linear layers, LongLoRA further makes embedding and normalization layers trainable. This extension is pivotal for context extension and only introduces a minimal number of additional trainable parameters. Image and Caption adapted from (Chen et al. 2023).

Chen et al. tackled the challenge of efficiently fine-tuning language models for applications that require processing long contexts by developing LongLoRA (Chen et al. 2023). LongLoRA employs a two-pronged approach: the Shifted Sparse Attention (S2-Attn) mechanism during fine-tuning, which reduces computation by splitting attention into local groups with shifted tokens, and the integration of trainable embeddings and normalization layers to enhance long-context adaptation. This combination allows LongLoRA to significantly reduce memory costs and training time while maintaining high performance. Experiments demonstrated LongLoRA's ability to extend context lengths to 100k tokens on models like LLaMA2 with up to 70 billion parameters, providing a scalable solution for handling long sequences in LLMs without compromising efficiency.

In their recent work, Dawid J. Kopiczko, Tijmen Blankevoort, and Yuki M. Asano introduced VeRA, an advanced method that enhances parameter efficiency in fine-tuning LLMs (Kopiczko et al. 2024). Unlike traditional LoRA, VeRA employs a pair of frozen low-rank matrices shared across all layers and adapts these matrices using trainable scaling vectors. The comparison between LoRA and VeRA is visually represented in Figure 2.5. This approach drastically reduces the number of trainable parameters while maintaining model performance. VeRA's efficacy was demonstrated across various benchmarks, including GLUE and E2E, and tasks such as instruction tuning for LLaMA models and image classification. By addressing the storage and computational challenges associated with large-scale model adaptation, VeRA presents a scalable and efficient solution for fine-tuning pre-trained models in diverse applications.

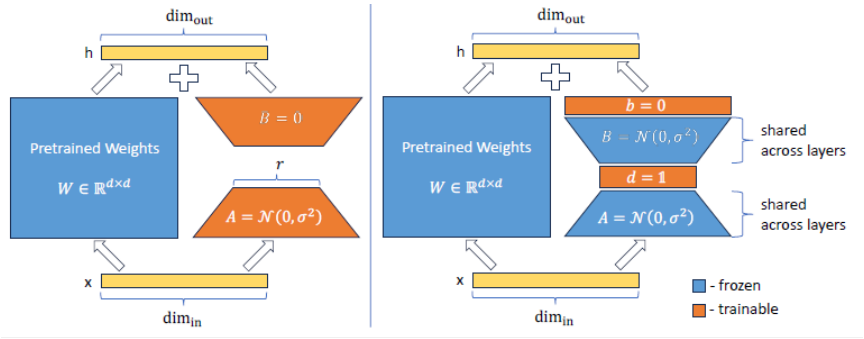

Figure 2.5.: Comparision of LoRA (left) and VeRA (right). LoRA updates the weights matrix W by training the low-rank matrices A and B, with intermediate rank r. In VeRA these matrices are frozen, shared across all layers, and adapted with trainable vectors d and b, substantially reducing the number of trainable parameters. Image and Caption adapted from (Kopiczko et al. 2024).

In a recent study, Tianyi Chen and colleagues proposed LoRAShear (Chen et al. 2023), a method designed for the structured pruning of LLMs with a focus on knowledge retention. LoRAShear employs a novel LoRA Half-Space Projected Gradient (LHSPG) technique to progressively prune less essential structures while transferring their knowledge to more critical components. This approach not only reduces the model's size by up to 50% but also preserves up to 82% of its original performance. By analyzing dependency graphs and utilizing dynamic knowledge recovery, LoRAShear effectively mitigates the trade-off between model compression and performance degradation, significantly outperforming existing pruning methods in computational efficiency and accuracy recovery.

In 2024, Liu et al. introduced DoRA (Weight-Decomposed Low-Rank Adaptation) (Liu et al. 2024), a method enhancing LoRA by decomposing pre-trained weights into magnitude and directional components. This decomposition allows for efficient fine-tuning by leveraging LoRA for directional updates and maintaining the magnitude component. DoRA's novel approach ensures high learning capacity, closely resembling full fine-tuning without additional inference overhead. Empirical results demonstrate DoRA's superior performance over traditional LoRA across various tasks, including commonsense reasoning and visual instruction tuning, highlighting its efficiency and robustness in parameter-efficient fine-tuning.

2.5.2. Token-Based Methods

Token-based parameter-efficient fine-tuning methods focus on optimizing specific token embeddings or representations within a pre-trained model rather than modifying the entire model's parameters. This subsection focuses on a selection of these techniques.

Method	Key Characteristic
Prefix Tuning	Optimizes continuous task-specific prefixes
Prompt Tuning	Tunes continuous, trainable vectors added to the input embeddings (soft prompts)
Input Tuning	Optimizes task-specific continuous embeddings in the model's input layer
PaSTA	Focuses on updating special tokens like [CLS] and [SEP]
KnowPrompt	Improves prompt tuning by adding learnable key elements in prompt
DePT	Splits soft prompts into shorter prompt and low-rank matrices

Table 2.2.: Overview of Token Based PEFT methods

Li and Liang introduced an approach termed prefix-tuning, which provides a parameter-efficient method for fine-tuning large pre-trained models on generative tasks (Li and Liang 2021). This method freezes the parameters of the pre-trained language model and instead optimizes a small, continuous task-specific vector called the prefix. These prefixes are prepended to the input tokens, allowing subsequent tokens to attend to them as if they were virtual tokens. The primary benefit of prefix-tuning is its efficiency: it requires tuning only 0.1% of the model's parameters, significantly reducing the computational and storage overhead compared to full model fine-tuning. Empirical results demonstrate that prefix-tuning achieves performance comparable to full fine-tuning in full-data settings and even outperforms it in low-data scenarios, making it a highly effective and resource-efficient method for adapting large language models to specific tasks.

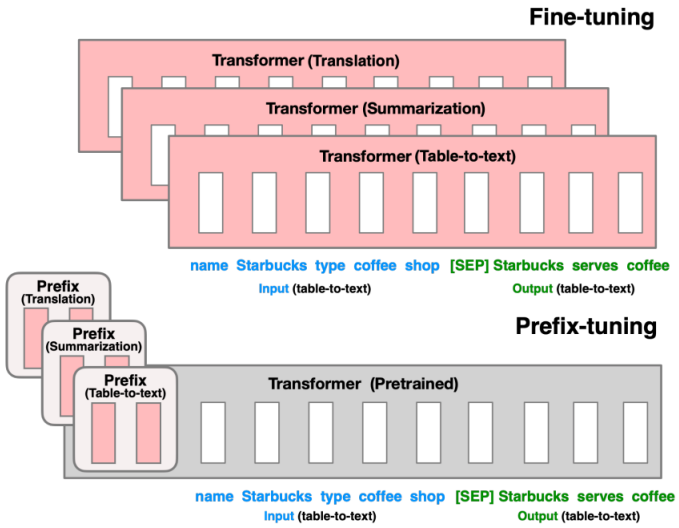

Figure 2.6.: Fine-tuning (top) updates all parameters (the red Transformer box) and requires storing a full model copy for each task. Prefix-tuning (bottom), freezes the Transformer parameters and only optimizes the prefix (the red prefix blocks). Consequently, only needed to store the prefix for each task, making prefix-tuning modular and space-efficient. Image and Caption adapted from (Li and Liang 2021).

In their 2021 study, Lester et al. explored the scalability of prompt tuning (Lester et al. 2021), a parameter-efficient method that adapts large language models like T5 by training only a small set of soft prompts instead of the entire model. This method maintains the efficiency of using a single pre-trained model across multiple tasks while significantly reducing the number of trainable parameters. The authors demonstrate that prompt tuning can achieve performance comparable to full model tuning, particularly as model size increases. For instance, prompt tuning with large models, such as T5-XXL, matches the quality of full model tuning while using orders of magnitude fewer task-specific parameters. This approach is especially advantageous in scenarios requiring the deployment of multiple task-specific models, as it simplifies storage and reduces computational overhead, making it a highly efficient solution for practical applications.

In 2022, An et al. introduced Input-Tuning (An et al. 2022), a method designed to enhance prompt-tuning by effectively handling unfamiliar inputs for natural language generation (NLG) tasks. Unlike traditional prompt-tuning, which struggles with inputs that differ significantly from the pretraining corpus, Input-Tuning employs a lightweight trainable module, called an input-adapter, placed between word embeddings and the model's input layer. This module transforms the input embeddings to better align with the pre-trained model's representations. The approach bridges the performance gap

between prompt-tuning and fine-tuning, achieving superior results on seven NLG tasks, and demonstrating that Input-Tuning can significantly outperform prompt-tuning and even rival fine-tuning in certain cases. This innovative method ensures that pre-trained language models can be more effectively adapted to a wide range of tasks with diverse input types, making it a robust solution for improving model performance in low-resource and varied input scenarios.

Xiaocong Yang et. al introduced Parameter-Efficient Tuning with Special Token Adaptation (PaSTA) (Yang et al. 2023). This method focuses on adapting special token representations, such as [CLS] and [SEP] in BERT, rather than modifying the entire model's parameters. By only adding trainable vectors to the hidden representations of these tokens before the self-attention module at each layer, PaSTA achieves parameter efficiency without sacrificing performance. With only up to 0.029% of the total parameters trained, PaSTA matches the effectiveness of full fine-tuning in tasks like text classification and named entity recognition. This approach underscores the critical role of special tokens in pre-trained language models and offers a scalable solution for multi-task, memory-limited scenarios, facilitating efficient adaptation across diverse tasks and datasets.

KnowPrompt (Chen et al. 2022), by Chen et al., is a method that enhances prompt-tuning for relation extraction by incorporating knowledge-aware elements. The authors propose using learnable virtual type words and answer words that integrate semantic knowledge from relation labels into the prompt construction process. KnowPrompt synergistically optimizes these elements with structured constraints, enabling the model to leverage implicit structural knowledge among relational triples. This approach significantly improves the performance of prompt-tuning, particularly in low-resource settings, by dynamically adjusting virtual words based on context, thus ensuring robust and accurate relation extraction. Experimental results on multiple datasets demonstrate that KnowPrompt outperforms existing methods, achieving state-of-the-art results while maintaining efficiency in parameter tuning and computational requirements.

In their 2023 paper, Zhengxiang Shi and Aldo Lipani introduced DePT, a method designed to optimize the prompt tuning approach by decomposing the soft prompt into a shorter prompt and a pair of low-rank matrices (Shi and Lipani 2024). This decomposition significantly reduces the input sequence length, thereby cutting down computational costs associated with training and inference in transformer models. DePT employs two distinct learning rates for the prompt and the low-rank matrices, enhancing convergence and efficiency. Experimental results across 23 NLP and vision-language tasks demonstrate that DePT outperforms existing PEFT methods, including vanilla prompt tuning and even full fine-tuning in some scenarios. The method shows particular promise in scaling efficiency as the model size increases, making it highly suitable for LLMs.

2.5.3. Other PEFT Methods

This section highlights PEFT methods that target different parts of a neural network or employ unique techniques compared to the previously discussed approaches.

Method	Key Characteristic
Trained Rank Pruning	Pushes network weights into a low-rank form
Scatterbrain	Leverages Sparse Attention with learned token interactions
DSEE	Combines Low-rank updates and sparse weight structures
Mixture of Experts	Uses Matrix Product Operator decomposition to share parameters
Scaling & Shifting Features	Adjusts deep features with linear transformations
BitFit	Fine-tunes only the bias terms of the model
BOFT	Applies Orthogonal Transformations to bitwise features

Table 2.3.: Overview of other PEFT methods

Yuhui et al. introduced a method known as Trained Rank Pruning (TRP) to enhance the efficiency of deep neural networks (Xu et al. 2019). The authors propose a technique that integrates low-rank approximation and pruning directly into the training process. Trained Rank Pruning aims to simplify the weight matrices by reducing their rank, essentially trimming down the complexity of the model. The key idea is to do this pruning during the training process, so the model learns which parts of the matrix are most important and which can be removed without significantly affecting its performance. This method leverages the singular value decomposition (SVD) of weight matrices, iteratively pruning less significant singular values while retaining the crucial components that preserve the network's performance. TRP effectively reduces the number of parameters and computational overhead, achieving up to a 90% reduction in parameters without significant loss of accuracy. The primary advantage of TRP lies in its ability to maintain the capacity and discriminative power of the original network by dynamically adapting the rank during training, ensuring efficient compression and minimal performance degradation.

Scatterbrain (Chen et al. 2021) is an approach to combining sparse attention with learned token interactions. Traditional methods often utilize either sparsity or low-rank properties of attention matrices to mitigate the computational and memory bottlenecks associated with modeling long sequences. However, these methods individually excel in

different regimes based on the softmax temperature, which can limit their effectiveness. Chen et al. address this by integrating both sparse and low-rank approximations, leveraging the strengths of each. ScatterBrain uses sparse attention by selectively focusing on a subset of tokens, using low-rank approximation, rather than processing all tokens at each attention layer. This approach reduces the computational load by concentrating resources on the most relevant parts of the input. By combining sparse attention with this learned token prioritization, ScatterBrain efficiently manages long sequences while maintaining strong performance and reducing memory usage. This combination allows for more accurate and efficient attention approximations. Empirical results show that Scatterbrain can achieve up to 2.1 times lower error than existing baselines when used as a drop-in replacement in tasks like BigGAN image generation and pre-trained T2T Vision Transformer (T2T-ViT). Notably, on the T2T-ViT model, Scatterbrain reduces 98% of attention memory with only a 1% drop in accuracy. Furthermore, it improves perplexity by up to 4 points and average accuracy by 5 points over sparse or low-rank efficient Transformers in language modeling and long-range arena tasks, demonstrating its efficacy in both training and inference scenarios.

Xuxi Chen et al. introduced Dually Sparsity-embedded Efficient Tuning (DSEE), a framework aimed at enhancing both parameter- and resource-efficiency during the fine-tuning of pre-trained language models (Chen et al. 2023). DSEE leverages the sparsity prior in weight updates and final model weights to achieve these goals. Specifically, the method integrates sparsity-aware low-rank updates during the fine-tuning process and enforces a sparse weight structure in the final model. This dual approach ensures that both the training and inference phases benefit from reduced computational requirements and memory usage. By incorporating both unstructured and structured sparsity patterns, DSEE significantly cuts down on the number of trainable parameters and improves inference efficiency. Extensive experiments on models like BERT, GPT-2, and RoBERTa across various datasets demonstrate that DSEE maintains competitive performance while achieving substantial reductions in parameter count and computational overhead, thereby making it a robust solution for efficient model deployment in resource-constrained environments.

In 2022, a parameter-efficient Mixture-of-Experts (MoE) architecture was designed to enhance the capacity of pre-trained language models without a proportional increase in parameters (Gao et al. 2022). This method, termed MPOE (Matrix Product Operator-based Expert), employs a tensor decomposition technique from quantum many-body physics to reduce parameter redundancy across experts. By sharing a global central tensor among different experts and allowing task-specific adaptations through auxiliary tensors, MPOE significantly lowers the parameter count while maintaining or even improving performance. The authors also introduce a gradient mask strategy to address unbalanced optimization, ensuring efficient training. Extensive experiments with models like T5 and GPT-2 demonstrate that MPOE achieves a 27.2x reduction in parameters compared to traditional MoE setups, making it a powerful solution for scaling large models efficiently.

A novel PEFT method called Scaling and Shifting Features (SSF) was introduced by Lian et al (Lian et al. 2022). Unlike traditional fine-tuning that updates all model parameters, SSF adjusts the deep features of a pre-trained model using linear transformations (scaling and shifting) to adapt the model to new tasks without adding extra parameters or increasing computational complexity during inference. This approach simplifies fine-tuning by modifying only the feature space, making it both resource-efficient and effective. SSF is particularly advantageous for scenarios requiring rapid adaptation with minimal overhead, providing a strong alternative to more complex methods. This method significantly reduces the number of trainable parameters—only about 0.3M compared to the full model's parameters. SSF demonstrates superior performance across various tasks, achieving higher accuracy than full fine-tuning in benchmarks like FGVC and VTAB-1k. The key advantage of SSF lies in its ability to maintain performance while eliminating additional inference costs, making it highly efficient for deployment in resource-constrained environments.

Ben-Zaken et al. proposed BitFit (Zaken et al. 2021), a PEFT method focusing exclusively on the bias terms of Transformer-based models. Unlike conventional fine-tuning, which adjusts all model parameters, BitFit modifies only the bias terms while freezing the rest. This approach dramatically reduces the number of trainable parameters—down to 0.08%—while maintaining comparable performance to full fine-tuning. BitFit is particularly effective in low-data regimes and memory-constrained environments, making it a practical solution for multi-task fine-tuning. The method's simplicity and efficiency provide new insights into the role of bias terms in pre-trained models and open up possibilities for hardware-friendly model implementations.

A novel parameter-efficient finetuning method known as Orthogonal Butterfly Finetuning (BOFT) was introduced by a group of researchers (Liu et al. 2024). This method builds upon traditional orthogonal finetuning (OFT) by employing a butterfly factorization to achieve a more efficient representation of orthogonal matrices as shown in Figure 2.7. The butterfly structure allows for the composition of dense orthogonal matrices from multiple sparse matrices, significantly reducing the number of parameters required. OFT itself operates by reparameterizing the weight matrices of neural networks as the product of orthogonal matrices and the original weights. This approach helps preserve the angular relationships between neurons, which is crucial for maintaining the semantic integrity of the pre-trained model during finetuning. However, the original OFT methods often rely on block-diagonal structures, which can be less efficient. BOFT addresses this inefficiency by leveraging butterfly matrices, known for their effectiveness in fast Fourier transforms and efficient matrix-vector multiplications. This factorization not only maintains the orthogonal properties essential for stable and effective finetuning but also reduces the computational and memory overhead associated with large-scale models. Experiments demonstrated that BOFT could achieve competitive performance on various downstream tasks while significantly reducing the parameter count compared to traditional finetuning methods. By integrating the butterfly factorization, BOFT presents a scalable and efficient solution for adapting large pre-trained models to new

tasks, making it an essential advancement in the field of parameter-efficient fine-tuning.

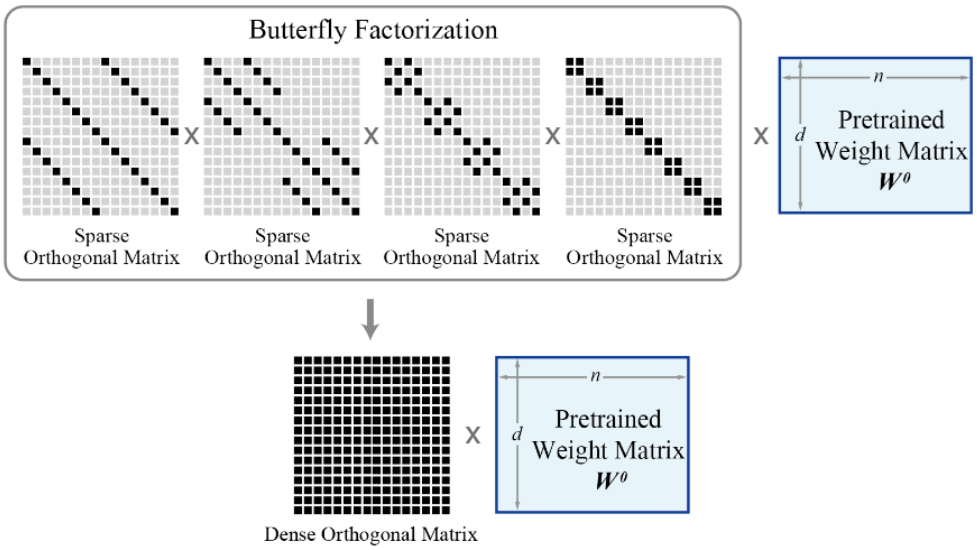

Figure 2.7.: BOFT uses the butterfly factorization to parameterize the orthogonal matrix, which yields a dense orthogonal matrix (in contrast, OFT has to use a sparse block-diagonal orthogonal matrix to reduce the number of trainable parameters). The butterfly parameterization of orthogonal matrices naturally generalizes the original OFT framework where the block-diagonal structure now becomes a special case of BOFT. Image and Caption adapted from (Liu et al. 2024).

In summary, the literature on PEFT methods is vast and continually evolving, with numerous techniques tailored to enhance the efficiency of fine-tuning large pre-trained models across various tasks. While this section has highlighted several notable methods, there are many more innovative approaches in the field. In the subsequent experiments, we will explore a select few PEFT methods to demonstrate their practical applications and effectiveness. Based on its popularity and usage in practical applications, LoRA has been a standout PEFT method. It is simple, efficient, and effective. It maintains high performance with minimal computational and memory overhead. It integrates seamlessly with existing models, adding no extra inference latency, which makes it versatile and widely applicable across various tasks and model architectures.

Moving forward, the next section will delve into Quantization, another crucial aspect of optimizing large language models for resource-efficient deployment.

2.6. Quantization

Quantization is a technique used in neural networks to reduce the precision of the numbers that represent the model's parameters, activations, or both, thereby decreasing the model's memory footprint and computational requirements. his process involves converting high-precision (typically 32-bit floating-point) numbers to lower-precision formats such as 16-bit, 8-bit, or even 4-bit integers. The primary goal of quantization is to reduce the computational and memory requirements of the model, making it more efficient to deploy and execute on hardware with limited resources, such as mobile devices and embedded systems.

As neural networks, particularly LLMs like GPT-3 and LLaMa, have grown in size and complexity, the need for efficient deployment has become increasingly critical. These models often consist of billions of parameters, leading to substantial demands on memory and computational power. This has led to a surge in research focused on making these models more efficient and accessible. Quantization addresses these challenges by significantly reducing the model size and the amount of computation required during inference. This reduction not only enables the deployment of large models on resource-constrained devices but also speeds up inference and reduces power consumption (Jacob et al. 2018). Quantization has become a key strategy to address memory issues, especially in parameter-efficient fine-tuning techniques (Dettmers et al. 2022b). By reducing the memory requirements and computational costs, quantization makes it feasible for researchers with limited resources to fine-tune large models.

There are two main categories of quantization techniques in deep learning:

- **Post-Training Quantization:** Applied only during inference, this method converts model weights and activations to lower precision (e.g., 8-bit) after full-precision training, reducing memory and computational costs without retraining.

- **Quantization-Aware Training (QAT):** Used during training, this method simulates quantization effects by incorporating low-precision operations in the training loop, leading to models that perform well even after quantization at inference, thereby minimizing accuracy loss.

Quantization is especially crucial for large models due to several reasons:

- **Resource Constraints:** Deploying large models on devices with limited memory and processing power requires reducing the model's footprint.

- **Scalability:** Quantization allows for the deployment of multiple instances of the model on the same hardware, enhancing scalability.

- **Inference Speed:** Lower precision arithmetic can be performed faster, thus speeding up the inference process.

- **Energy Efficiency:** Reduced precision operations consume less power, which is vital for battery-operated devices.

The next parts will review various quantization techniques employed to optimize large models. This review will examine each approach, post-training quantization, and quantization-aware training in detail. Each technique offers unique advantages and trade-offs, providing a spectrum of options to balance model performance with efficiency.

2.6.1. Quantization Aware Training

An important development by Micikevicius et al. is the application of mixed-precision training, which uses higher precision (FP32) for certain areas of the model, like gradient accumulation, and lower precision (FP16) for the forward and backward passes (Micikevicius et al. 2018). This technique employs three key strategies: maintaining a master copy of the weights in FP32, applying loss scaling to preserve small gradient values, and accumulating intermediate results in FP32 before converting them back to FP16. These methods collectively reduce memory requirements and speed up training on modern GPUs without modifying hyperparameters. Mixed Precision Training demonstrates its effectiveness across various deep learning tasks, including image classification, object detection, and language modeling, making it a robust solution for scaling neural networks efficiently.

Benoit Jacob and colleagues at Google introduced a quantization scheme designed for efficient integer-arithmetic-only inference (Jacob et al. 2018). This method converts both weights and activations to 8-bit integers, while a few parameters, like biases, are kept at 32-bit integers. The key advantage of this approach is that it allows inference to be performed using integer-only arithmetic, which is significantly more efficient on common mobile hardware compared to floating-point operations. The training procedure co-designed with this quantization scheme ensures that the accuracy of the model is preserved post-quantization. The method has demonstrated significant improvements in the trade-off between accuracy and on-device latency, particularly with models like MobileNets. It is highly effective for tasks such as ImageNet classification and COCO detection on mobile CPUs, showcasing the potential of integer-arithmetic-only inference for deploying deep learning models in resource-constrained environments.

Method	Key Characteristic
Mixed Precision Training	Switches between 16-bit and 32-bit precision while training
Integer-only Arithmetic	Introduces integer level QAT while maintaining accuracy
Q8BERT	Quantizes both Fully connected & embedding layers into 8-bit during finetuning
Block-Wise Quantization	Partitions tensors into blocks to reduce outlier impact
KD on QAT	Uses Knowledge Distillation techniques on QAT
AdaPT	Decides about precision switches between training epochs based on information theoretic conditions
GPT3 Int8	Efficient Matrix multiplication by quantizing weights and activations to 8-bit
Quantized Distributed Training	Quantizes the gradients and uses an innovative error feedback mechanism
Integer Forward/Backward Propagation	Leverages dynamic fixed-point representation to map floating-point numbers to integers
Activation Aware Quantization	Identifies and preserves a small fraction of salient weights crucial for maintaining model performance
Quantized Full Tuning	Uses Lion optimizer to track momentum while training with Quantization

Table 2.4.: Overview of Quantization methods used during training

Q8BERT addressed the challenge of deploying BERT models in production by introducing an 8-bit quantization technique during the fine-tuning phase (Zafrir et al. 2019). This method involves quantizing both the Fully Connected and Embedding layers of BERT, resulting in significant reductions in model size and inference time without compromising accuracy. Specifically, Q8BERT achieves a 4x reduction in memory footprint and maintains comparable performance to its 32-bit counterpart across various NLP tasks. The quantization-aware training approach ensures the model learns to bridge the quantization error gap, thereby optimizing performance for 8-bit hardware environments. This advancement facilitates the deployment of large-scale language models in resource-constrained production settings, offering both computational efficiency and reduced latency.

8-bit Optimizers via Block-wise Quantization by Dettmers et al. presents a groundbreaking technique for optimizing deep learning models using 8-bit precision without sacrificing performance (Dettmers et al. 2022). The approach involves block-wise dynamic quantization, which partitions tensors into smaller blocks, reducing the impact of outliers and enhancing stability. This method allows for substantial memory savings, reducing the optimizer state memory usage by up to 75% while maintaining 32-bit performance levels across various tasks, including language modeling, image classification, and machine translation. By requiring only minor code adjustments, this method offers a practical solution for efficient model training in resource-constrained environments.

Minsoo Kim et al. introduced a detailed study on improving Knowledge Distillation (KD) techniques for Quantization-Aware Training (QAT) of large Transformer encoders (Kim et al. 2022). The paper examines the limitations of current KD approaches, such as using MSE loss on attention scores, and proposes two new methods: attention-map and attention-output losses. These approaches aim to better recover self-attention information in quantized models. Experiments on BERT-Base and BERT-Large show that the proposed methods significantly enhance model accuracy in sub-2-bit quantization scenarios, setting new benchmarks for QAT performance

Adaptive Precision Training (AdaPT) is a dynamic approach to quantized training of deep neural networks (Kummer et al. 2023). AdaPT dynamically adjusts the precision of weights and activations during training on a per-layer basis, using a precision-switching mechanism guided by information-theoretic criteria. This method balances low precision for efficiency with sufficient precision for learning, thereby improving both training and inference efficiency. AdaPT demonstrates significant reductions in computational cost and memory usage while maintaining high accuracy across models like AlexNet and ResNet on CIFAR-10/100 datasets.

Tim Dettmers et al. introduced an innovative approach to optimizing transformer models through 8-bit quantization (Dettmers et al. 2022a). The proposed method enables efficient matrix multiplication in GPT-3 models by reducing the precision of weights and activations to 8-bit, significantly enhancing computational efficiency without compromising model accuracy. This approach achieves notable memory and speed improvements, with up to 4.5x faster inference and 5.6x reduction in memory footprint compared to standard 32-bit precision. By implementing mixed-precision training and a novel quantization algorithm that preserves critical information in lower precision, the authors successfully demonstrate that transformer models can be effectively scaled while maintaining performance. This method proves particularly beneficial for deploying large-scale language models in production environments where computational resources are a constraint. The study's empirical evaluations validate the robustness of the 8-bit quantization technique, highlighting its potential to revolutionize the deployment of transformer-based models in real-world applications.

Quantized distributed training methods have gained significant attention for their

ability to reduce the computational complexity and memory usage of large-scale models, particularly in the context of deep learning. Markov et al. in the paper "Quantized Distributed Training of Large Models with Convergence Guarantees" (Markov et al. 2023) explores a novel quantized training method that ensures convergence while significantly reducing computational overhead. This method leverages low-precision arithmetic to compress model weights, thereby decreasing memory usage and accelerating training times. The approach is particularly beneficial in distributed training scenarios, where communication costs are a major bottleneck. By quantizing the gradients and using an innovative error feedback mechanism, the method maintains model accuracy comparable to full-precision training. The empirical results demonstrate that this quantized training approach not only preserves convergence properties but also achieves substantial improvements in training speed and resource utilization, making it a promising solution for scaling large models efficiently.

Tayaranian et al. introduced an innovative method for fine-tuning pre-trained language models using integer arithmetic for both forward and backward propagation (Tayaranian Hosseini et al. 2023). This approach, a departure from traditional methods that rely on floating-point operations, leverages dynamic fixed-point representation to map floating-point numbers to integers. Their experiments demonstrated that fine-tuning BERT with 16-bit integers matches the performance of FP16 and FP32 baselines, while 8-bit integers incurred minimal performance loss, highlighting the potential for significant computational and memory efficiency gains.

Activation-aware Weight Quantization presents a novel method for compressing large language models by focusing on the significance of activation-aware weights (Lin et al. 2024). Unlike traditional quantization techniques that uniformly reduce precision across all weights, AWQ identifies and preserves a small fraction (0.1%-1%) of salient weights crucial for maintaining model performance. This selective preservation minimizes quantization errors and enhances computational efficiency. Experiments show that AWQ achieves a 3.2-3.3x speedup over FP16 implementations while maintaining comparable performance, making it a viable solution for deploying large models on resource-constrained devices.

The method Quantized Full-Parameter Tuning (QFT) by Li et. al. is a framework that employs quantization for efficient memory use in fine-tuning LLMs (Li et al. 2024). Utilizing the Lion optimizer, which tracks only momentum, QFT quantizes all model states to significantly reduce memory usage. This approach allows fine-tuning a LLaMA-7B model within 30GB of RAM, achieving comparable performance to standard methods. QFT's integration of comprehensive quantization and memory-efficient optimizers demonstrates its potential for resource-constrained environments.

The next section will explore Post Training quantization techniques which are mainly used for deploying large models in resource-constrained environments.

2.6.2. Post Training Quantization

Early work in quantization for neural networks focused on simple linear quantization schemes. For instance, Han et al. demonstrated that reducing the precision of weights can significantly compress models without a considerable drop in accuracy (Han et al. 2016). This method involves a three-stage pipeline: pruning, quantization, and Huffman coding. Initially, network pruning removes redundant connections, retaining only the most critical ones. This is followed by trained quantization, which reduces the precision of weights and enforces weight sharing through k-means clustering. Finally, Huffman coding further compresses the quantized weights. The combined effect of these techniques significantly reduces the storage size of models without sacrificing accuracy. For instance, the AlexNet model, originally requiring 240MB, was compressed to 6.9MB, and VGG-16 from 552MB to 11.3MB. This reduction enables large models to fit into the on-chip SRAM cache rather than relying on off-chip DRAM memory, thereby improving energy efficiency and inference speed. The Deep Compression method facilitates the deployment of complex neural networks in mobile and resource-constrained environments, making it a foundational contribution to the field of model compression.

In 2018, Polino et al. introduced a method combining distillation and quantization to compress deep neural networks (Polino et al. 2018). This approach involves two key techniques: quantized distillation and differentiable quantization. Quantized distillation trains a smaller student network using the distillation loss from a larger teacher model while simultaneously quantizing the student's weights. Differentiable quantization optimizes quantization points through gradient descent, enhancing the student model's accuracy by better aligning with the teacher model's behavior. Experiments demonstrated that these methods achieve significant model compression and speedup, making them highly effective for deploying deep learning models in resource-constrained environments.

Ye Lin and colleagues explored an innovative approach to enable almost fully 8-bit integer inference in Transformer models, addressing significant memory and computational efficiency concerns (Lin et al. 2021). This is achieved through a method called Scale Propagation, which maintains integer operations throughout the model by managing scales associated with INT8 tensors. The authors also propose the Integer Transformer, which modifies traditional functions like the exponential and square root to more integer-friendly alternatives, such as polynomial functions and L1 normalization. Experiments conducted on machine translation and language modeling tasks demonstrate that this approach achieves competitive performance while reducing memory usage by nearly fourfold and speeding up inference by approximately 3.5 times. This advancement opens new possibilities for deploying high-performance NLP models in resource-constrained environments without sacrificing accuracy.

Method	Key Characteristic
Deep Compression	Prunes, quantizes and uses Huffman coding to reduce model size
Distillation & Quantization	Trains smaller network using distillation loss and optimizes quantization points
8-bit int inference	Modifies traditional functions like the exponential to more integer-friendly alternatives
Q-BERT	Uses a Hessian-based approach to guide the quantization process, exploiting the sensitivity of different layers
Compressing GPTs	Implements 8-bit quantization for weights and activations
ZeroQuant	Fine-grained Quantization and layer-by-layer Knowledge distillation
GPTQ	Utilizes approximate second-order information
LRRVQ	Combines Low-Rank representations with vector quantization
4-bit FP	Quantizes weights and activations by Leveraging per-channel activation quantization
Atom	Combines mixed-precision and fine-grained group quantization, dynamically adjusting quantization parameters during inference
AffineQuant	Employs Affine transformations, dynamically adjusts scaling factors for different layers
Partially-Binarized LLMs	Maintains full precision for crucial parts of the network while binarizing less critical components
LUT-GEMM	Utilizing Look-Up Tables for efficient matrix multiplication

Table 2.5.: Overview of Post Training Quantization methods

Shen et al. introduced a layer-wise quantization method that adjusts precision based on layer-specific characteristics, leading to more efficient and robust models (Shen et al. 2020). Q-BERT, is a method aimed at achieving ultra-low precision quantization for BERT models without significant performance degradation. The authors use a Hessian-based approach to guide the quantization process, exploiting the sensitivity of different layers to precision reduction. By applying mixed-precision quantization and a novel group-wise quantization scheme, Q-BERT effectively reduces the bit-width of weights

and activations while preserving model accuracy. Experiments on tasks like SST-2, MNLI, CoNLL-03, and SQuAD demonstrate that Q-BERT achieves substantial compression ratios with minimal accuracy loss, enabling efficient deployment of BERT models in resource-constrained environments.

Efficient quantization techniques were discussed by Tao et al. (Tao et al. 2022), for compressing Generative Pre-trained Language Models (GPTs) to make them more suitable for deployment in resource-constrained environments. By implementing 8-bit quantization for both weights and activations, the authors achieve significant reductions in memory usage and computational demands without substantially impacting model performance. This method ensures that the compressed models retain high accuracy levels across various natural language processing tasks. Experimental results demonstrate that quantized models can achieve nearly the same performance as their full-precision counterparts, making them a viable option for real-world applications where efficiency is paramount.

Yao et al. introduced ZeroQuant, an effective post-training quantization (PTQ) approach to compress Transformer models (Yao et al. 2022). ZeroQuant employs fine-grained quantization schemes and a novel layer-by-layer knowledge distillation (LKD) algorithm, enabling the compression of BERT and GPT-3 models to INT8 with minimal accuracy loss. The method achieves up to 5.19x speedup on BERT and 4.16x on GPT-3, significantly reducing memory footprint. By leveraging hardware-friendly optimizations, ZeroQuant demonstrates substantial efficiency improvements, making it feasible to deploy large models in resource-constrained environments without the need for retraining.

Dettmers and Zettlemoyer investigated optimal bit-precision for LLMs through extensive experiments, establishing that 4-bit quantization strikes the best balance between model size and accuracy. This study spans models from 19M to 176B parameters, showing that 4-bit precision consistently delivers superior performance compared to higher precisions. The findings emphasize the efficiency of small block sizes and specific data types, such as float and quantile quantization, in enhancing model accuracy and stability. This optimal scaling law is pivotal for deploying large models in resource-constrained environments while maintaining high inference performance (Dettmers and Zettlemoyer 2023).

A study by Wu et al. (Wu et al. 2023) examines the impact of INT4 quantization on language models, focusing on latency improvements, composability, and potential failure modes. By reducing the precision to 4-bit integers, the models achieve significant speedups in inference latency, making them highly efficient for deployment in resource-constrained environments. The research highlights the balance between maintaining model accuracy and achieving computational efficiency. However, it also addresses the challenges, such as potential accuracy degradation in specific scenarios and issues with model composability. These findings are crucial for optimizing large language models

for practical applications.

The authors, Elias Frantar and colleagues, introduced GPTQ (Frantar et al. 2023), an efficient post-training quantization method for compressing large-scale GPT models to 3 or 4-bit precision. GPTQ utilizes approximate second-order information to achieve high accuracy and efficiency, compressing models like GPT-3 (175 billion parameters) in about four GPU hours with minimal accuracy loss. The method more than doubles the compression gains compared to previous techniques while preserving performance, allowing execution on a single GPU. This makes GPTQ a significant advancement for deploying massive language models in resource-constrained environments.

Low-Rank Representation Vector Quantization (LRRVQ) (Zhu et al. 2023) is a method for compressing deep neural networks by combining low-rank representations (LRR) with vector quantization (VQ). This approach decouples subvector size from clustering dimensionality, enabling more efficient compression. By optimizing both parameters, LRR-VQ achieves significant improvements in model performance and compression ratios. For instance, it enhances the top-1 accuracy of ResNet-18 and ResNet-50 by 2.8% and 1.0%, respectively, over existing VQ methods, while achieving compression factors of 43x and 31x. This method leverages low-rank approximations to reduce clustering errors in quantization, thus maintaining high model accuracy with lower storage requirements. The study's comprehensive experiments on ImageNet and COCO datasets validate the method's effectiveness, demonstrating its potential for real-world applications in resource-constrained environments.

Liu et al. propose a novel method to quantize both weights and activations of large language models to 4-bit floating-point values (Liu et al. 2023). The technique addresses high inter-channel variance in activation distributions, leveraging per-channel activation quantization to maintain model performance. This approach enables efficient compression, achieving significant memory and computational savings with minimal loss in accuracy. For instance, the method quantizes LLaMA-13B to 4-bit with only a 5.8-point drop in zero-shot reasoning tasks, outperforming previous state-of-the-art techniques by a substantial margin.

Atom (Zhao et al. 2024) is a sophisticated low-bit quantization method specifically designed for optimizing LLM serving. Atom employs a combination of mixed-precision and fine-grained group quantization, dynamically adjusting quantization parameters during inference to enhance efficiency. By quantizing both weights and activations to 4-bit precision, Atom achieves up to 7.7x improvement in throughput compared to FP16, with negligible accuracy loss. This method also incorporates KV-cache quantization, further boosting memory and computational efficiency, making it a highly effective solution for deploying LLMs in resource-constrained environments.

AffineQuant, by Ma et al. (Ma et al. 2024), is a quantization method that employs affine transformations to enhance the efficiency of large language models. Unlike tra-

ditional techniques, AffineQuant dynamically adjusts scaling factors for different model layers, thus preserving the precision of critical computations. This technique results in substantial memory savings and computational efficiency, achieving near full-precision performance with reduced bit-width representations. The study's empirical results show that AffineQuant maintains high accuracy across various NLP benchmarks, making it a practical approach for deploying large models in constrained environments.

Jing Liu and colleagues present an innovative quantization technique aimed at optimizing large language models for deployment in resource-constrained environments (Liu et al. 2024). By employing advanced low-bitwidth quantization methods, the paper demonstrates how to maintain high accuracy while significantly reducing memory and computational requirements. The proposed method achieves substantial efficiency improvements, making it feasible to serve large models on hardware with limited capabilities. Empirical results indicate that QLLM can perform on par with or even exceed the performance of higher-bitwidth counterparts in various tasks.

Partially Binarized LLMs, introduced by Yuan et al. (Yuan et al. 2024), is an approach to reduce the computational complexity and memory footprint of large language models by selectively binarizing portions of the model. This method maintains full precision for crucial parts of the network while binarizing less critical components, striking a balance between efficiency and performance. The partially binarized models achieve substantial improvements in inference speed and resource utilization, with minimal impact on accuracy. This technique demonstrates significant potential for deploying LLMs in environments with limited computational resources.

The research by Park et. al., introduces LUT-GEMM (Park et al. 2024), a novel quantization approach utilizing Look-Up Tables (LUTs) for efficient matrix multiplication in large-scale generative language models. By mapping frequent multiplication results to precomputed LUTs, this method significantly reduces computational overhead. The paper demonstrates that LUT-GEMM achieves comparable accuracy to full-precision models while offering substantial improvements in inference speed and energy efficiency. Specifically, experiments reveal up to a 5x speedup and a 4x reduction in energy consumption, making it an effective solution for real-time applications and deployment in resource-constrained environments.

Recent advancements in quantization techniques for large models underscore a rapidly evolving field focused on enhancing model efficiency and deployment feasibility. From INT4 quantization to dynamic methods utilizing Look-Up Tables, the innovations aim to balance computational speed and model accuracy. As these methods continue to mature, they promise to unlock new possibilities for deploying sophisticated models in resource-limited settings. In the next chapter, we will explore fine-tuning strategies with quantized models, examining how these approaches can further optimize performance and efficiency.

2.7. Summary

This chapter provided a comprehensive overview of key concepts relevant to this thesis. A thorough literature review was conducted on PEFT and Quantization, highlighting significant advancements in these areas. It was observed that most research in Quantization focuses on Post-Training Quantization (PTQ). There is a need for more development of Quantization Aware Training. Since Quantization is mainly used post-training for deployment, this also explains the gap in studies that combine Quantization with PEFT. The next chapter focuses on the intersection of these techniques.

3. Efficiently Fine-Tuning Quantized Models

As the previous chapter illustrated, Quantization and PEFT, have emerged as essential techniques for large-scale neural network fine-tuning and deployment, particularly in the natural language processing domain. These techniques have made LLMs far more accessible to a wider range of users in different industries. Each technique independently addresses critical challenges associated with the scaling and deployment of deep learning models. Together, they offer complimentary advantages that significantly improve the efficiency of the model. With the use of their own datasets and combined use of Quantization & PEFT, academics, developers, and even small businesses may now fine-tune huge models for certain applications without requiring a lot of processing capacity.

When combined, these two techniques significantly lower the barrier to entry for customizing LLMs. The integration of quantization and PEFT leverages the strengths of both methodologies to achieve an optimal balance between model performance and efficiency. Quantization significantly lowers the computational requirements for both training and inference by reducing the precision of model parameters. When applied alongside PEFT methods, like Adapters, LoRA, or IA3, which minimizes the number of trainable parameters, the overall resource consumption is drastically reduced. This dual approach ensures that quantized models occupy less memory, and the additional storage required for fine-tuned parameters is minimized. This is particularly crucial in scenarios where storage capacity is a bottleneck. The combined application enables the scaling of NLP models to handle larger datasets and more complex tasks without proportional increases in resource requirements. This approach makes it easier to install cutting-edge models on smaller, resource-constrained devices while simultaneously reducing the cost of model deployment and maintenance. Together, these two synergies reduce the memory footprint and computational overhead, making advanced NLP technologies more widely available and useful for a greater number of applications.

The integration between quantization and PEFT can be effectively utilized through a well-defined process:

- Initially, a Large Language Model is pre-trained on a huge corpus using FP32 precision to ensure high accuracy. Post-training, the model undergoes initial quantization to a lower precision such as BF16, INT8 or INT4.

- The quantized model is then optimized using PEFT techniques for a specific task. For instance, in QLoRA, only the low-rank adaptation matrices are fine-tuned while the original LLM is frozen in a quantized state.

- The resulting model, now optimized through both quantization and PEFT, is deployed for inference. This final model is highly efficient in terms of memory usage and computational requirements while still delivering high accuracy.

The practical implications of combining quantization and PEFT are significant, facilitating the customization of LLMs for specific applications with minimal resource investment. This approach democratizes access to advanced language models, enabling diverse use cases across various domains. For instance, Researchers can fine-tune a pre-trained model on clinical trial data, creating a specialized model adept at understanding and generating medical text. This quantized and fine-tuned model can be deployed in healthcare applications, offering insights or generating reports efficiently, even on devices with limited computational power. Developers working with regional languages or dialects can fine-tune a pre-trained model using a small corpus of text specific to that dialect. This results in an efficient and accurate language model capable of running on standard laptops, making advanced NLP accessible in linguistically diverse regions. Businesses in the financial sector can fine-tune large models on domain-specific data, such as financial reports and news articles. This creates tools that assist in financial analysis and decision-making, providing enhanced insights and efficiencies in financial operations.

The way large language models are deployed and fine-tuned has been revolutionized by the combination of quantization and PEFT. These approaches have enabled a wide range of users, from individual developers to small organizations, to customize and deploy strong language models that are specifically suited to their needs. This democratization of AI technology paves the way for more innovative and accessible applications, driving advancements across various fields and industries. The following literature will explore several methods that leverage the combination of quantization and PEFT to achieve efficient fine-tuning.

Dettmers et. al. revolutionalized this domain of Quantization+PEFT methods by introducing QLoRA (Dettmers et al. 2024). QLoRA leverages quantization techniques, specifically using a 4-bit NormalFloat (NF4) data type that optimally represents normally distributed weights, to finetune LLMs efficiently. By backpropagating gradients through a frozen, 4-bit quantized pre-trained language model into Low-Rank Adapters (LoRA), QLoRA significantly reduces memory usage, allowing the fine-tuning of a 65 billion parameter model on a single 48GB GPU while maintaining performance levels comparable to 16-bit fine-tuning. Empirical results demonstrate that QLoRA achieves near state-of-the-art performance on their Vicuna model, attaining 99.3% of the performance level of ChatGPT with only 24 hours of fine-tuning on a single GPU. Additionally, QLoRA's methodology ensures that smaller, high-quality datasets can still yield com-

petitive results, offering a cost-effective alternative to traditional fine-tuning methods. We will learn more about QLoRA in the next section.

Method	Key Characteristic
QLoRA	Innovative 4-bit quantization and double quantization combined with LoRA
PEQA	Updates only quantization scales while keeping rest of the model frozen in 4-bit
QA-LoRA	Upgrades QLoRA by using group-wise operators
LoftQ	Focuses on reducing quantization error by improving LoRA initialization
LQ-LoRA	Decomposes pre-trained matrices into high-precision low-rank and quantized components
L4Q	Optimizes LoRA Weights and quantization parameters

Table 3.1.: Overview of PEFT Methods used on Quantized Models

Jeonghoon Kim and colleagues introduced an advanced technique designed to mitigate the high memory demands and computational costs associated with fine-tuning LLMs (Kim et al. 2024). This method called Parameter-Efficient and Quantization-aware Adaptation (PEQA), combines the benefits of PEFT with the advantages of sub-4-bit quantization. By updating only the quantization scales while keeping the integer matrix frozen, PEQA significantly reduces the memory overhead typically associated with the optimizer state during fine-tuning. This approach ensures that the quantization structure remains intact even after fine-tuning, thereby allowing for accelerated inference during deployment. Empirical results demonstrate that PEQA can restore or even improve the performance of LLMs in language modeling and comprehension tasks, despite the models being quantized to below 4-bit precision. This makes PEQA a highly effective solution for enhancing the efficiency and scalability of LLMs while maintaining their performance integrity.

Inspired by QLoRA, Xu et al. presented QA-LoRA (Xu et al. 2024). By employing group-wise operators, QA-LoRA increases the degrees of freedom for quantization while reducing those for adaptation, resulting in lower memory usage and faster inference. Applied to LLaMA and LLaMA2 models, QA-LoRA outperforms baseline QLoRA in accuracy, especially under low-bit quantization conditions, making it a significant advancement in LLM fine-tuning techniques.

Yixiao Li et al. introduced LoftQ (Li et al. 2024), a novel quantization framework designed to enhance the performance of quantized LLMs when fine-tuned using Low-Rank Adaptation (LoRA). LoftQ integrates quantization with low-rank approximation, optimizing the initialization for LoRA fine-tuning. This method addresses the performance degradation seen in quantized models, particularly in low-bit regimes. Experimental results show that LoftQ outperforms existing methods like QLoRA across various NLP tasks, demonstrating significant improvements in efficiency and generalization.

An approach called LQ-LoRA, combining low-rank adaptation with quantized matrix decomposition to enhance memory efficiency in LLM finetuning was explored by Guo et. al. (Guo et al. 2024). Using an iterative algorithm, it decomposes pretrained matrices into high-precision low-rank and quantized components. This method enables sub-3-bit quantization with minor performance loss, outperforming QLoRA and GPTQ-LoRA baselines. LQ-LoRA demonstrates significant memory savings while maintaining high performance across multiple tasks.

One of the most recent methods called L4Q, combined Quantization Aware Training (QAT) with LoRA (Jeon et al. 2024). Leveraging LoRA-wise learned quantization step size, L4Q optimizes both LoRA weights and quantization parameters, enabling efficient fine-tuning of large models like LLaMA and LLaMA2. This approach achieves sub-4-bit precision, maintaining high accuracy with reduced memory usage and training times, outperforming traditional PEFT and QAT methods.

As can be observed, the volume of research in this field has mostly been in the last year. There has been an increasing trend to develop more methods in the intersection of Quantization and PEFT. These methods demonstrate significant advancements in optimizing memory efficiency and computational resource utilization while maintaining or enhancing model performance.

The approaches that will be used in the experiments with Quantization and PEFT will be thoroughly discussed in the sections that follow. Their usefulness and practical uses in LLM fine-tuning will be assessed based on the results of this concentrated analysis.

3.1. QLoRA

Quantization and Low-Rank Adaptation (QLoRA) is a novel technique that leverages both quantization and low-rank adaptation methods to fine-tune large-scale language models efficiently. This method is particularly valuable for adapting large pre-trained models to specific tasks while keeping computational and storage costs manageable.

Core Components of QLoRA:

- **Quantization:** Reduces the precision of model parameters to decrease the computational and storage burden, typically converting floating-point numbers to lower bit-width integers.

- **LoRA:** Fine-tunes pre-trained models by adapting a small subset of parameters, projected into a lower-dimensional space, which significantly reduces the number of parameters updated during fine-tuning.

QLoRA integrates quantization with LoRA to achieve efficient fine-tuning of large language models. QLoRA employs advanced quantization techniques, specifically 4-bit NormalFloat (NF4) quantization and double quantization, to enhance memory efficiency and computational speed while maintaining model accuracy. This technique reduces the precision of the model's parameters from the standard 32-bit floating-point format (FP32) to more compact formats (4 bit), thereby facilitating the deployment of large models on resource-constrained hardware. Figure 3.1 shows the difference between Full-finetuning, LoRA-finetuning, and QLoRA fine-tuning. QLoRA improves over LoRA by quantizing the frozen transformer model to 4 bits.

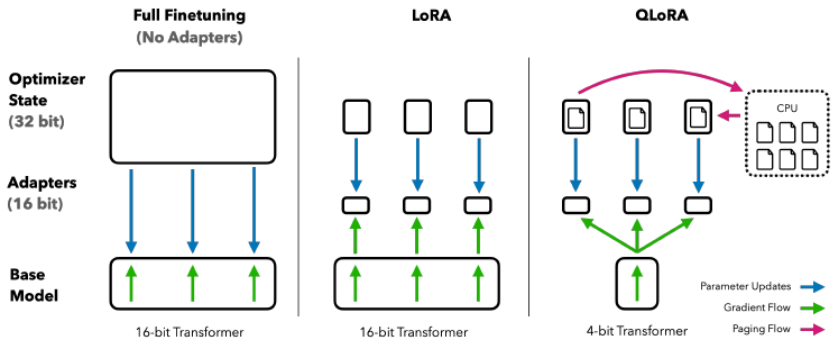

Figure 3.1.: Different finetuning methods and their memory requirements. QLoRA improves over LoRA by quantizing the transformer model to 4-bit precision and using paged optimizers to handle memory spikes. Image and Caption adapted from (Dettmers et al. 2024).

3.1.1. Quantization in QLoRA

4-bit NormalFloat (NF4) Quantization

The goal of NF4 quantization is to drastically reduce the model size while maintaining the original floating-point values' dynamic range. The key advantage of NF4 quantization

lies in its ability to strike a balance between precision and memory efficiency.

The NormalFloat (NF4) data type builds on Quantile Quantization, which is information-theoretically optimal for normally distributed weights. It ensures each quantization bin has an equal number of values assigned from the input tensor. The quantization process for NF4 involves the following steps:

- The quantiles for a zero-centered normal distribution are calculated to obtain a k-bit quantile quantization data type. This involves determining the $2^k + 1$ quantiles of a standard normal distribution $N(0, 1)$

$$q_i = \frac{1}{2}\left(Q_X\left(\frac{i}{2^k+1}\right) + Q_X\left(\frac{i+1}{2^k+1}\right)\right)$$

  where Q_X is the quantile function of the standard normal distribution.

- The calculated quantiles are normalized to fit into the range $[-1, 1]$, ensuring that the data type matches the distribution of the neural network weights.

- An input weight tensor is normalizing into the $[-1, 1]$ range through absolute maximum rescaling.

- Once the weight range and data type range match, we can quantize using the Block-wise k-bit Quantization as described in the related work section.

- To ensure a discrete zero point and use all 2^k bits, NF4 employs an asymmetric data type by estimating quantiles for negative and positive parts of the distribution, unifying them and then removing one of the two zeros.

Double Quantization

Double quantization quantizes the quantization constants themselves, significantly compressing the model. This two-step process reduces the memory footprint significantly.

- NF4 quantization is used in the first quantization to transform the FP32 weights to a 4-bit representation. This involves calculating the quantization constants c for each block of weights.

- Further quantization converts the first level's quantization constants, which are normally stored as 32-bit floats, to 8-bit floats. By doing this, the overhead of keeping these constants is decreased. Formally, if c_2 are the quantization constants from the first quantization, they are treated as inputs to the second quantization.

By applying double quantization, QLoRA achieves further memory efficiency without significant performance degradation.

3.1.2. Low Rank Adapters

Low-rank adaptation (LoRA) is among the most widely used and effective techniques for efficiently training custom LLMs. LoRA works by augmenting the existing weights of a neural network with low-rank matrices that are specifically trained for the new task. The procedure is explained as follows:

- Given a weight matrix $W \in R^{d*k}$ of a neural network layer. LoRA decomposes it into two lower-rank matrices $A \in R^{d*r}$ and $B \in R^{r*k}$ where r is a hyperparameter and is usually chosen much smaller than both d and k. The original weight matrix can be approximated by $W + \Delta W$, where $\Delta W = A$ x B.

- These low-rank matrices A and B are initialized such that ΔW is a zero matrix. Generally, A is initialized with small random values and B is initialized with zeros.

- During fine-tuning, the parameters of the original pre-trained model W are kept frozen, and only the parameters in A and B are updated.

- After training, the effective weight matrix used in the model is $W + \Delta W$. The updates from the low-rank matrices A and B provide the necessary task-specific adjustments to the model.

This decomposition and selective training approach ensure that the model can be efficiently adapted to new tasks with minimal additional computational resources.

In our experiments, we focus on three crucial parameters: the rank (r), the scaling factor (α), and LoRA dropout. These parameters play a significant role in determining the performance and efficiency of the fine-tuning process.

The rank parameter determines the dimensionality of the low-rank matrices A and B. Essentially, r defines the shape and complexity of these matrices. According to the original LoRA paper, even a small rank can yield excellent results, as the low-rank approximation effectively captures the essential updates needed for fine-tuning (Hu et al. 2021). Setting an appropriate rank is crucial, as it balances the trade-off between model complexity and computational efficiency. A lower rank reduces the number of parameters, leading to faster computations and lower memory usage, while a higher rank may capture more intricate patterns in the data but at the cost of increased computational demands.

The scaling factor (α) acts as a multiplicative factor that scales the low-rank updates applied to the weight matrix. This parameter controls the impact of the low-rank updates on the original weights, effectively functioning as a learning rate. By adjusting α, we can regulate the magnitude of the updates BA during the fine-tuning process. A higher scaling factor increases the influence of the low-rank updates, potentially accelerating convergence but also risking instability if set too high. Conversely, a lower scaling factor moderates the updates, promoting stable training at the expense of slower convergence.

The dropout parameter introduces a dropout mechanism to the low-rank updates, which helps in regularizing the fine-tuning process and preventing overfitting. Dropout randomly zeroes out a fraction of the low-rank update units during training, encouraging the model to learn more robust representations that generalize better to unseen data.

The combined effect of these parameters significantly influences the fine-tuning performance and efficiency of LoRA. By carefully tuning these parameters, we can achieve an optimal balance between computational efficiency, memory usage, and model accuracy. Lower ranks reduce computational load but may miss out on capturing finer details in the data, while higher ranks provide richer updates at the cost of increased resource consumption. Similarly, α needs to be set appropriately to ensure that the updates are neither too aggressive nor too timid, ensuring stable and effective learning.

3.1.3. Combination of Quantization and LoRA

The concepts of quantization and LoRA are combined in QLoRA to provide a potent and effective model fine-tuning method. Combining the parameter-efficient adaptation capabilities of LoRA with the memory and computational efficiency of quantization results in a powerful combination.

QLoRA for a single layer is defined as follows:

$$Y^{BF16} = X^{BF16} doubleDequant(c_1^{FP32}, c_2^{k-bit}, W^{NF4}) + X^{BF16} A^{BF16} B^{BF16} \tag{3.1}$$

where doubleDequant(.) is defined as:

$$\begin{aligned} doubleDequant(c_1^{FP32}, c_2^{k-bit}, W^{k-bit}) &= dequant(dequant(c_1^{FP32}, c_2^{k-bit}), W^{k-bit}) \\ &= W^{BF16} \end{aligned} \tag{3.2}$$

In summary, QLORA supports one calculation data type (BF16) and one storage data type (NF4). To conduct the forward and backward passes, it dequantizes the storage data type to the calculation data type; however, it only computes weight gradients for the LoRA parameters that employ BF16.

3.1.4. Advantages of QLoRA

- By combining low-precision quantization with low-rank adaptation, QLoRA dramatically reduces memory and storage needs, enabling the effective deployment of large-scale models on hardware with limited resources.

- The integration of 4-bit quantization and LoRA reduces the number of arithmetic operations during both training and inference, enabling faster processing times and reduced computational costs.

- QLoRA's fine-tuning approach, focusing on low-rank adaptations within a quantized model, ensures that large pre-trained models can be efficiently adapted to specific tasks with minimal resource overhead.

3.1.5. Implementation Considerations

- Optimising QLoRA's performance and efficiency requires fine-tuning its hyperparameters, which include learning rates and the rank of the low-rank decomposition.

- The effectiveness of QLoRA can vary based on the neural network architecture. Models with larger weight matrices and significant redundancy in their parameters tend to benefit more from this approach.

3.2. Rank Stabilized LoRA

Rank-stabilized LoRA (rsLoRA) introduces a novel scaling factor for fine-tuning LLMs using LoRA. Traditional LoRA applies a low-rank matrix product scaled by a rank-dependent factor, which leads to slowed learning and diminished performance with higher-rank adapters. RsLoRA modifies this by scaling the adapters with the square root of the rank, enhancing learning stability and performance, especially for larger ranks.

Mathematical Formulation
The conventional LoRA-augmented sub-module is given by:

$$x_{out} = (W + \gamma_r BA)x_{in} + b \tag{3.3}$$

where W and b are pre-trained parameters, and γ_r is the scaling factor. In rsLoRA,

the scaling factor is adjusted to:

$$\gamma_r = \frac{\alpha}{\sqrt{r}} \tag{3.4}$$

This adjustment ensures that the learning trajectory remains stable, preventing gradient collapse as rank r increases.

Advantages:
RsLoRA's critical improvement lies in its ability to utilize higher ranks effectively without the drawbacks of gradient collapse, which is prevalent in conventional LoRA as the rank increases. This allows for a compute/performance trade-off where larger ranks can be used to improve fine-tuning performance with no additional inference cost. Experimental results show that rsLoRA unlocks better performance with higher ranks, whereas traditional LoRA fails to benefit from increased rank due to its overly aggressive scaling factor.

RsLoRA is significant for its potential to enhance fine-tuning efficiency and performance in LLMs. By stabilizing the learning process for higher ranks, it facilitates the use of larger computational resources effectively, leading to better model adaptation and performance. This method provides a pathway to more efficient and scalable fine-tuning of LLMs, which is crucial as model sizes continue to grow.

3.3. LoftQ

LoftQ, is an approach designed to enhance the efficiency of training LLMs by combining principles of LoRA and quantization techniques. LoftQ aims to address the computational and memory challenges associated with fine-tuning and deploying LLMs, especially when dealing with resource-constrained environments. The method addresses the performance gap between full-precision and quantized models by incorporating low-rank approximations during the quantization process. This ensures a more effective initialization for LoRA fine-tuning, reducing the discrepancy introduced by quantization.

The core idea of LoftQ, similar to QLoRA, is to apply quantization with LoRA, effectively compressing the model while preserving its performance. The LoftQ method operates in three primary stages: low-rank adaptation, quantization, and weight initialization. Proper initialization of weights is crucial in LoftQ to ensure stable and efficient training. In this method, the weights of the low-rank matrices A and B are initialized using a scheme that preserves the properties of the original weight matrix.

Mathematical Formulation
LoftQ minimizes the difference between the original pre-trained weights W and the

quantized weights plus low-rank matrices $Q + BA$. The objective function is:

$$min_{Q,A,B}||W - Q - BA||_F \tag{3.5}$$

where Q is the quantized weight matrix, A and B are low-rank matrices, and $||.||_F$ denotes the Frobenius norm. The method uses alternating optimization, alternating between quantizing the residual and applying singular value decomposition (SVD) to refine A and B.

Advantages:
LoftQ offers significant advantages over traditional LoRA by integrating quantization and low-rank approximation, leading to enhanced initialization that keeps the initial weights closer to the original pre-trained weights, thereby improving fine-tuning stability and performance. It excels in low-bit quantization scenarios, maintaining superior performance compared to QLoRA, especially in challenging lower-bit mixed precision regimes. Additionally, LoftQ consistently outperforms QLoRA across various NLP tasks, including natural language understanding, question answering, summarization, and language generation.

LoftQ is significant as it bridges the gap between full-precision and quantized models, enabling efficient and effective fine-tuning of LLMs with reduced computational resources. This is particularly crucial for deploying large models in resource-constrained environments, ensuring they can still achieve high performance across diverse tasks.

3.4. IA3

IA3, which stands for Infused Adapter by Inhibiting and Amplifying Inner Activations, is a PEFT method designed for optimizing LLMs with minimal additional parameters. IA3 scales intermediate activations in transformers by learned vectors, enabling effective fine-tuning with a tiny fraction of the model's parameters. This method leverages the existing architecture without extensive modifications, making it highly efficient and scalable.

Mathematical Formulation
IA3 introduces learned vectors $l_k \in R^{d_k}$, $l_v \in R^{d_v}$, $l_{ff} \in R^{d_f f}$ that rescale the keys and values in attention layers and the activations in feed-forward networks.

The key transformation is defined as:

$$Attention(Q, K, V) = softmax(\frac{Q(l_k \odot K)^T}{\sqrt{d_k}})(l_v \odot V) \tag{3.6}$$

For the feed-forward network, the transformation is:

$$FNN(x) = l_{ff} \odot ReLU(W_1 x)W_2 \tag{3.7}$$

where $\odot$ represents element-wise multiplication.

Advantages:
IA3 surpasses traditional LoRA by requiring fewer parameters while achieving superior performance. The method scales the activations directly, providing a more stable learning process and better optimization. IA3 can handle mixed-task batches efficiently, unlike many other PEFT methods that re-parameterize the model. This allows for flexible and efficient multitask learning.

By updating only a small set of parameters, IA3 ensures that models can be adapted to new tasks quickly and efficiently without extensive retraining. Additionally, quantizing the frozen model further enhances efficiency by reducing memory usage and computational load, making it even more beneficial.

3.5. Weight-Decomposed Low-Rank Adaptation

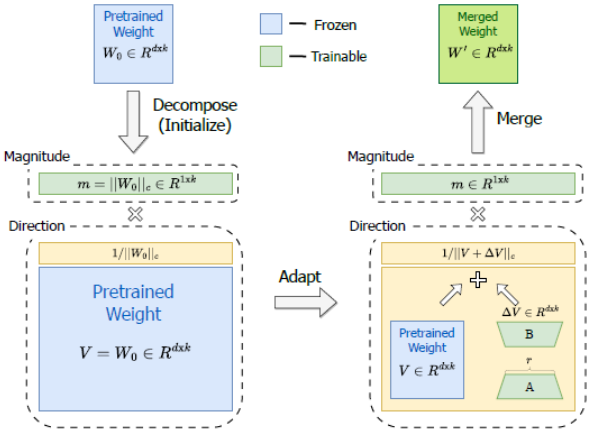

Figure 3.2.: DoRA decomposes the pre-trained weight into magnitude and direction components for fine-tuning, especially with LoRA to efficiently update the direction component. Image and Caption adapted from (Liu et al. 2024).

Weight-Decomposed Low-Rank Adaptation or DoRA is a method designed to enhance the learning capacity and stability of LoRA without adding any inference overhead. DoRA decomposes pre-trained weights into two distinct components: magnitude and direction. This decomposition is inspired by weight normalization techniques and aims to closely resemble the learning capacity of full fine-tuning while maintaining the efficiency of LoRA.

Mathematical Formulation
The core idea behind DoRA is to reparameterize the weight matrix W into magnitude m and directional V components:

$$W = m\frac{V}{||V||_c} = ||W||_c\frac{W}{||W||_c} \tag{3.8}$$

where $||.||_c$ denotes the vector-wise norm of a matrix across each column. In DoRA, m is the magnitude vector and V is the directional matrix, ensuring each column of V remains a unit vector.

DoRA applies this decomposition to the pre-trained weights and uses LoRA for the directional updates. The fine-tuned weight W' can be represented as:

$$W' = m\frac{W_0 + BA}{||W_0 + BA||_c} \tag{3.9}$$

where W_0 is the pre-trained weight matrix and BA are the low-rank updates learned by LoRA.

Advantages:
DoRA addresses the limitations of traditional LoRA by effectively managing the trade-off between learning capacity and computational efficiency. By decomposing weights into magnitude and direction, DoRA allows for more nuanced updates that resemble the learning patterns of full fine-tuning, thus improving the learning capacity over traditional LoRA. The decomposition stabilizes the optimization process, leading to more efficient training and better performance. DoRA maintains the inference efficiency of LoRA by merging the learned updates with the pre-trained weights before deployment, ensuring no additional latency is introduced.

It enhances fine-tuning performance across various tasks, including natural language processing and vision-language understanding, without increasing the computational burden during inference. This makes DoRA a valuable method for optimizing large-scale models in resource-constrained environments.

3.6. Half Quadratic Quantization

Half Quadratic Quantization (HQQ) (Badri and Shaji 2023) is a novel approach to quantization that aims to reduce the memory footprint and computational cost of LLMs while preserving model accuracy. Traditional quantization methods often suffer from significant quantization errors, particularly when reducing the precision of weights to improve efficiency. HQQ addresses this challenge by approximating the quantization function using a quadratic form, thereby enhancing the fidelity of the quantized weights.

Mathematical Formulation
HQQ operates by minimizing the difference between the original weights of a neural network and their quantized counterparts using a quadratic loss function. This approach ensures that the quantization error is reduced compared to linear quantization methods. Specifically, HQQ formulates the quantization process as:

$$Q(\theta) = argmin_Q ||\theta - Q||_F^2 + \lambda ||Q||_F^2 \tag{3.10}$$

where θ represents the original weights of the model, Q denotes the quantized weights, $||.||_F$ denotes the Frobenius norm, and λ is a regularization parameter that controls the trade-off between fidelity to the original weights and quantization error.

Advantages:

- By incorporating the quadratic loss term, HQQ minimizes the discrepancy between original and quantized weights, thereby preserving model accuracy better than linear quantization methods.
- HQQ offers a balanced trade-off between model efficiency and accuracy.

3.7. Summary

This chapter briefly explored the intersection of Quantization and PEFT methods and provided an in-depth analysis of the techniques employed in the experiments. Given the limited and emerging research in this combined area, a comprehensive study and evaluation of these methods across various criteria is essential.

The next chapter outlines the experiments, detailing the objectives, datasets, models, configurations, and key research questions. It also summarizes the results, offering insights into the application of PEFT methods to quantized models.

4. Experimental Study

This chapter presents a series of experiments designed to explore the efficacy of various PEFT methods combined with different quantization techniques for fine-tuning LLMs. The primary objective of these experiments is to evaluate the performance and memory efficiency of different PEFT methods when applied to quantized LLMs. The experiments encompass a variety of configurations and methodologies to provide a comprehensive analysis of the capabilities and trade-offs associated with each approach.

The motivation behind these experiments is rooted from the the growing need for efficient tuning and deployment of LLMs in resource-constrained environments. LLMs, such as LLaMa and Mistral, have demonstrated remarkable performance across a range of NLP tasks, however, their substantial memory and computational requirements pose significant challenges for practical applications. Addressing these challenges is critical to making such models more accessible and practical for a broader range of use cases.

The experiments are conducted exclusively with causal language models, focusing on generative models and tasks related to natural language generation.

4.1. Goal of the Experiments

The primary aim of these experiments is to systematically evaluate and compare the effectiveness of various PEFT methods when combined with different quantization techniques for fine-tuning LLMs. This comprehensive analysis aims to identify the optimal strategies for enhancing the performance, efficiency, and practicality of deploying LLMs in resource-constrained environments.

- **Assess Memory and Computational Efficiency:** Quantization techniques are employed to reduce the memory footprint and computational demands of LLMs. The goal is to measure the impact of these techniques on model efficiency without compromising performance.

- **Evaluate Performance of PEFT Methods:** Several PEFT methods are explored, including LoRA, RS-LoRA, IA3, and DoRA, to determine their effective-

ness in fine-tuning LLMs with reduced trainable parameters. Each method's ability to maintain model performance and robustness is critically assessed.

- **Understand the Practical Implications:** Beyond theoretical performance metrics, the objective is to understand the practical implications of deploying these fine-tuning strategies in real-world scenarios. This includes evaluating the ease of implementation, scalability, and potential impact on various NLP tasks.

4.2. Datasets used for Fine-tuning

Unnatural Instructions Core The Unnatural Instructions Core dataset (Honovich et al. 2023) is an instruction-tuning dataset collected via various approaches of model distillation from GPT-3 Instruct and ChatGPT. This dataset leverages techniques such as prompting, in-context learning, and paraphrasing to generate diverse sets of instructions and outputs. It comprises 240,670 examples, making it a substantial resource for fine-tuning models to handle a wide variety of instruction styles and contexts. The dataset's diversity is a significant advantage, providing a broader range of instruction styles compared to other instruction tuning collections.

HH-RLHF The Anthropic HH-RLHF dataset (Bai et al. 2022) is a human preference dataset focused on the helpfulness and harmlessness of assistant replies. Each data point includes two assistant replies to a user question, accompanied by a human preference judgment indicating the better reply. This dataset contains 160,800 examples. For fine-tuning purposes, we combine the helpfulness and harmlessness data and retain only the preferred assistant reply. This approach ensures that the model learns to prioritize responses that are both useful and safe, enhancing the quality of the assistant's interactions with users.

OpenAssistant The OpenAssistant dataset (Köpf et al. 2024) was collected through crowd-sourcing efforts. It includes 161,443 unique messages across 66,497 conversations, covering 35 different languages. Typically, the dataset features multiple ranked replies for each user question. In our experiments, we use only the top reply at each conversation level, reducing the dataset to 9,209 examples. This dataset is used for fine-tuning models on entire conversations, including user queries, enabling the development of models that can engage in more natural and contextually aware interactions.

4.3. LLMs used in the experiments

LLaMa 2 LLaMa2, developed by Meta AI, is an advanced large language model designed to improve natural language understanding and generation. It builds upon the architecture of its predecessor, LLaMa, with enhanced capabilities in handling diverse NLP tasks. LLaMa2 is trained on a vast corpus of text data, enabling it to generate coherent and contextually relevant responses. Its robust performance across various benchmarks makes it a suitable choice for fine-tuning experiments aimed at optimizing model efficiency and accuracy.

LLaMa3 LLaMa3 represents the next iteration in the LLaMa series, featuring significant improvements in both architecture and training methodologies. This model incorporates advanced techniques in transformer architecture and benefits from extensive pretraining on diverse datasets. LLaMa3 is designed to further push the boundaries of natural language processing, offering superior performance in text generation, comprehension, and contextual awareness. Its enhanced capabilities make it a valuable asset for experiments focused on fine-tuning and quantization.

4.4. Implementation Details

The implementation process for my experiments follows a systematic approach to fine-tuning LLMs using various quantization techniques and PEFT methods. This section outlines the general steps involved in the implementation, providing a high-level overview of the procedure without delving into specific configurations or methods.

- **Model Selection:** A suitable pre-trained language model is selected as the base model for fine-tuning.
- **Quantization:** The original weights of the selected model are quantized using a chosen quantization technique. This step involves reducing the precision of the weights to lower-bit representations to decrease the memory footprint and computational requirements. In all our experiments, we use bf16 computation datatype.
- **PEFT Method:** A PEFT method is integrated into the fine-tuning process to further optimize the model. They are employed to reduce the number of parameters that need to be updated during fine-tuning.
- **Fine-Tuning:** The quantized and adapted model is fine-tuned on a target NLP task using one of the selected datasets. The fine-tuning is performed in a supervised

learning way with Cross-Entropy Loss. We use a constant learning rate schedule and use group-by-length to group examples of similar lengths in the same batch (note this will produce an oscillating loss curve).

- **Training and Evaluation:** During the fine-tuning process, the model's performance is monitored and evaluated using standard metrics. Memory usage and computational efficiency are also recorded to assess the impact of the quantization and adaptation methods.

4.5. Evaluation Criteria

Perplexity : Perplexity is an intrinsic measure to evaluate the performance of language models. Perplexity measures the uncertainty of a language model's predictions. It measures the confidence model in its (next word) predictions. Simply put, it quantifies how well the model-predicted probability distributions align with the actual distribution of the words in the dataset. In a practical example, we could train a model on a training dataset and evaluate its perplexity on a separate validation dataset. This helps us understand how well the model generalizes on the unseen data. Ideally, perplexity should be low on both the training and validation datasets. Low perplexity only guarantees a model is confident, not accurate.

In the experiments, perplexity is measured on a validation set to evaluate the model's confidence on the data it is being fine-tuned on. This metric helps assess how well the model generalizes and adapts during the fine-tuning process, providing insight into its overall effectiveness and reliability on unseen data.

Downstream Task - MMLU: Massive Multitask Language Understanding (MMLU) is chosen to evaluate models on a downstream tasks. It is a benchmark designed to measure knowledge acquired during pretraining by evaluating models exclusively in zero-shot and few-shot settings. In the case of thesis, a few shot settings will be considered to directly compare with the QLoRA paper.

The benchmark encompasses 57 subjects across STEM, the humanities, social sciences, and more, with difficulty levels ranging from elementary to advanced professional. It assesses both world knowledge and problem-solving ability, covering traditional subjects like mathematics and history, as well as specialized areas such as law and ethics. Given that many LLMs are typically evaluated based on their average performance across all domains, the LLMs fine-tuned in our experiments will be assessed in the same manner.

4.6. Experimental Setup

4.6.1. Combination 1 - QLoRA & variations

This series of experiments explores the application of QLoRA in fine-tuning LLMs. The goal of these experiments is to investigate the impact of different LoRA parameter configurations on model performance and memory efficiency. By varying the parameters r (rank) and α, the aim is to understand how these configurations affect the efficiency and effectiveness in fine-tuning LLMs using these techniques, ultimately providing insights into the optimal settings for specific language modeling tasks.

In addition, the experiments also investigate the impact of using Rank Stabilizing Low-Rank Adaptation (RsLoRA) with Quantization under the same configurations. RsLoRA introduces an additional stabilizing mechanism to the low-rank adaptation process as described in 3.2, aiming to improve the robustness and performance consistency of the fine-tuned models. We evaluate how RS-LoRA affects the model's performance and memory efficiency compared to standard LoRA when applied with QLoRA's quantization techniques.

Lastly, we build upon the previous investigations by incorporating LoftQ initialization into the fine-tuning process. LoftQ, as explained in section 3.3, is designed to further enhance the efficiency and performance of QLoRA by optimizing the initialization of low-rank matrices. Specifically, LoftQ has been shown to reduce quantization error by ensuring that the low-rank approximations closely match the original weight matrices from the start. This reduces the need for extensive fine-tuning and adjustment, leading to faster convergence and potentially better model performance. We can measure the impact of LoftQ initialization on model performance and memory efficiency, both with and without RS-LoRA.

This experimentation is crucial for enhancing the practical deployment of LLMs in resource-constrained environments, where both computational efficiency and model performance are critical. This experimentation is crucial for enhancing the practical deployment of LLMs in resource-constrained environments, where both computational efficiency and model performance are critical.

The motivation behind this experiment comes from the need to balance model performance with computational efficiency in the context of LLM fine-tuning. Traditional fine-tuning of LLMs is resource-intensive, requiring substantial computational power and memory. QLoRA offers a promising solution by combining quantization with low-rank adaptation, thereby reducing the model's memory footprint and computational demands without significantly compromising performance. Additionally, incorporating a rank stabilization technique may offer further improvements in terms of both robustness and ef-

ficiency. However, initializing low-rank matrices appropriately remains a challenge that can impact the overall effectiveness of these methods. LoftQ addresses this challenge by providing a structured approach to weight initialization that preserves the properties of the original weight matrix, leading to more stable and efficient training.

Specifically, this experiment addresses the following questions:

- How do different configurations of the LoRA parameters r and α influence the performance of QLoRA?
- Can we identify parameter settings that provide an optimal balance between efficiency and accuracy?
- How does QLoRA affect memory load and overall memory efficiency during the fine-tuning process?
- Assess the benefits of rank stabilization and LoftQ in the context of QLoRA.
- Compare the performance and memory efficiency of RS-LoRA against standard LoRA under the same configurations and effectiveness of LoftQ with and without RS-LoRA.

By systematically varying r and α, the objective is to derive practical guidelines for fine-tuning LLMs using QLoRA, facilitating more efficient and effective model deployment in real-world applications.

The experiment involves fine-tuning a pre-trained language model using the integrated QLoRA, RS-LoRA, and LoftQ methods. The configurations for the low-rank adaptation parameters are as follows:

- Configuration 1: $r = 64$, $\alpha = 16$
  - Replicating the settings of QLoRA paper. This configuration uses a relatively high rank with a moderate scaling factor, aiming to balance the complexity and learning capacity of the low-rank matrices.
- Configuration 2: $r = 8$, $\alpha = 16$
  - With a lower rank, this setting is expected to reduce computational demands while testing the impact on model performance.
- Configuration 3: $r = 16$, $\alpha = 32$
  - This setup increases the scaling factor compared to Configuration 2, allowing

us to evaluate how increased scaling influences the effectiveness of a moderately low rank.

- Configuration 4: $r = 64$, $\alpha = 128$
  - By significantly increasing the scaling factor while maintaining a high rank, this configuration tests the upper limits of scaling's impact on model performance.
- Configuration 5: $r = 256$, $\alpha = 128$
  - This configuration uses the highest rank among all settings, combined with a high scaling factor, to explore the effects of maximum parameter settings on both performance and computational cost.

This combined experiment aims to systematically investigate the integrated effects of QLoRA, RsLoRA, and LoftQ on the performance and memory efficiency of LLMs. By evaluating these methods together, we seek to identify the optimal strategies for efficient and robust model adaptation, facilitating the deployment of LLMs in diverse and resource-constrained environments.

The quantization techniques used in these experiments follow the same approach as QLoRA, which involves Normal Float (NF4) quantization and double quantization. The experiments are conducted on different pre-trained language models which are mentioned in Section 4.3. The implementation steps mentioned in the previous section were followed for the experiments. The rest of the experimental settings are consistent across all the experiments and can be found in the Appendix **??**.

4.6.2. Combination 2 - QIA3

In this experiment, we explore the application of a PEFT method known as IA3. Unlike previous experiments that incorporated LoRA, this experiment focuses exclusively on IA3, evaluating its effectiveness with quantization. The goal is to assess the performance and memory efficiency of IA3 when combined with NF4 and the double quantization method.

The purpose of this experiment is to investigate the potential of IA3 with Quantization. IA3 enhances model adaptation by iteratively adjusting the attention mechanisms, which are crucial for capturing dependencies and context in language tasks. This iterative approach allows the model to dynamically fine-tune its attention layers, potentially leading to more precise and contextually appropriate outputs. Implementing IA3 aims to achieve high performance with fewer trainable parameters, thereby reducing compu-

tational costs and improving efficiency. By comparing the effects of IA3 with different quantization techniques, we seek to understand its advantages over static fine-tuning methods and identify configurations that maximize both performance and resource efficiency.

IA3 is quite easy to implement and doesn't have any hyperparameters to tune or create different configurations. The results can be directly compared with the results from the previous experiments.

4.6.3. Combination 3 - QDoRA

Here, the experiments explore the application of Dynamic Low-Rank Adaptation (DoRA) in the fine-tuning process of LLMs. DoRA introduces dynamic adjustments to the low-rank adaptation process, aiming to enhance the adaptability and performance of the model. This experiment seeks to evaluate the impact of DoRA on model performance and memory efficiency, providing a comparative analysis against previous configurations without DoRA.

The motivation behind this experiment is to leverage DoRA to further enhance the robustness and flexibility of QLoRA. While previous experiments with standard LoRA and RsLoRA have demonstrated significant improvements in memory efficiency and performance, there is a need for dynamic mechanisms that can adjust the low-rank adaptation parameters during training. DoRA addresses this need by introducing dynamic adjustments based on the training process, which can lead to more efficient learning and better overall performance.

DoRA is particularly beneficial because it adapts the rank and scaling factor dynamically, allowing the model to better capture complex patterns and relationships in the data. This adaptability is expected to improve convergence rates and enhance the model's ability to generalize from the training data.

Specifically, this experiment aims to:

- Assess the benefits of DoRA in enhancing the performance and efficiency of QLoRA.
- Compare the effectiveness of DoRA against standard LoRA, RS-LoRA, and LoftQ configurations.

These configurations are consistent with those used in the previous experiments, allowing for direct comparisons.

4.6.4. Combination 4 - HQQ-LoRA

Lastly, we explore the application of a different quantization technique, Half Quadratic Quantization (HQQ). HQQ is a quantization technique that aims to further reduce the memory footprint and computational requirements of LLMs without compromising performance. This experiment evaluates the impact of HQQ on model performance and memory efficiency by applying it to the configurations used in the previous experiments.

The reason for conducting this experiment is to investigate whether HQQ can offer additional benefits over the previously used quantization techniques. While these techniques have proven effective, HQQ introduces a different approach that may yield further improvements in efficiency and performance. HQQ claims to be way faster than other quantization methods. HQQ is also able to quantize models to as low as 1-bit.

Specifically, HQQ aims to:

- Enhance the quantization process by reducing quantization error through a quadratic approximation.
- Improve the overall robustness and efficiency of the fine-tuning process when combined with PEFT methods.

This experiment aims to determine whether HQQ can deliver superior performance and memory efficiency compared to the existing quantization methods, thereby contributing to more effective fine-tuning techniques for large language models. The experimental configurations of PEFT methods are consistent with those used in the previous experiments, allowing for direct comparisons.

The previously used quantization methods are replaced with HQQ 4-bit and HQQ 2-bit. HQQ operates by approximating the quantization function using a quadratic form, effectively minimizing quantization error while better preserving the original weight distributions. This technique is designed to achieve a more refined balance between precision and efficiency, offering enhanced performance alongside the benefits of low-bit quantization.

Now that the experiments are explained in detail, the next section provides a detailed analysis of the results from these experiments.

4.7. Results

The following sections will discuss the results of the above-mentioned experiments. A total of 226 fine-tuning runs were performed. The initial 28 runs focused on testing various hyperparameter configurations, while the remaining runs aimed to address the core research questions of the thesis.

The analysis of the results is structured around the following questions:

- How does quantization affect the model size and performance?
- By how much do PEFT methods reduce the trainable parameters and what is the additional overhead memory requirement?
- How does fine-tuning using different Quantization and PEFT methods compare with each other on model performance based on evaluation perplexity and MMLU?
- How do the methods compare with each other based on memory footprint and run time?

4.7.1. Quantization Effect

We start by discussing the effect of quantization on the size of the model. As mentioned in the previous section, we use two different quantization methods (NF4 + Double Quantization (DQ) and HQQ). NF4+DQ was introduced in the QLoRA paper which quantizes a model to 4bits and HQQ was introduced recently by Mobius Labs which supports quantization up to 1-bit. In our experiments, we focus on 4-bit quantization for direct comparison with QLoRA. We also look at 2-bit quantization with HQQ to inspect the results of fine-tuning on even lower precision.

Model	Model Size	NF4+DQ	HQQ-4bit	HQQ-2bit
LLaMa 2 - 7B	13.48 GB	3.51 GB	3.51 GB	1.99 GB
LLaMa 2 - 70B	137.98 GB	32.85 GB	-	-
LLaMa 3 - 8B	16.07 GB	5.21 GB	-	-

Table 4.1.: Model sizes of the LLaMa Family before and after quantization

Table 4.1 shows us the effect of quantization on model size. We can see that quantized models save a lot of space. Using NF4 and double quantization, we reduced the model size by approximately 73% for LLaMa 2 7B and around 68% for LLaMa 3 8B. For the bigger models, we see a reduction of 76% for LLaMa 2 and % for LLaMa 3. This saves

Model	Before Quantization	After Quantization
LLaMa 2 7B	45.3	40.68
LLaMa 2 70B	68.9	67.12
LLaMa 3 8B	66.4	53.86

Table 4.2.: 5-shot MMLU Test Accuracy for LLaMa Models **without Fine-tuning**. The second column shows the MMLU scores of the actual model, without fine-tuning or quantization. The last column depicts the MMLU score after quantization.

a lot of memory and can easily help to run 7B models on a local computer, while you can run 70B models on a single 48GB GPU. The HQQ-4Bit has a similar effect, which is expected as the precision is the same. Quantizing a model further down to 2-bit using HQQ reduces the memory requirement by 85% which can help to run 7B parameter models on a mobile device.

Looking at the effect of quantization on model performance, we can see in table 4.2 that all models face a degradation. There's a clear drop in the MMLU accuracy of all the models. We know that the weights are converted from a higher precision to a lower precision data type. During quantization, rounding or truncation occurs, leading to a loss of information, resulting in a drop in MMLU accuracy. This added error after quantization is known as the quantization error. This error affects the model's ability to represent fine-grained differences between weights. However, the 70B model looks less sensitive to quantization. Although speed, memory, and power usage are highly optimized, there is an accuracy trade-off.

Let's take a closer look at the score of LLaMa 3 models. In comparison, the drop after quantization in LLaMa 3 is larger than LLaMa 2. One might expect LLaMa 3 to perform better than LLaMa 2 given the amount and quality of data it is trained on. Turns out, this high-quality training is a drawback when the model is quantized. There is a higher quantization error. In recent studies, researchers have observed that LLaMa 3 suffers from higher degradation in low-bit scenarios (Huang et al. 2024). LLaMa 3 is trained on a huge corpus of a record 15 trillion tokens and it captures extremely nuanced data relations. That means in high precision (FP32 or BF16), LLaMa 3 learns information to the smallest of decimals possible making it more sensitive to quantization degradation. LLaMa 3 has lack of redundancy within the network. Modifying the network is more damaging to overall performance when a model is trained to a high level of saturation in high-precision data type. While LLaMa 3 performs better than a lot of other models, this quantization degradation highlights the challenge of deploying the model on smaller, low-compute devices.

After studying the effect of quantization on the model size and performance, the

following sections briefly discuss the impact of PEFT methods on trainable parameters & memory requirements of these parameters.

4.7.2. Trainable Parameters

One of the primary objectives of PEFT methods is to fine-tune only a small subset of parameters. In these experiments, we observe how different configurations of LoRA impact the number of trainable parameters. This subsection will delve into the trainable parameters across various PEFT methods, highlighting how each approach balances parameter efficiency. Table 4.3 illustrates how different configurations of PEFT methods impact the number of trainable parameters.

PEFT Method	Model	Config	Trainable Parameters	Trainable Parameters (%)
LoRA	LLaMa 2 7B	$r = 8$	19,988,480	0.2855
		$r = 16$	39,976,960	0.5711
		$r = 64$	159,907,840	2.2844
		$r = 256$	639,631,360	9.1376
	LLaMa 2 70B	$r = 64$	828,375,040	1.1867
IA3	LLaMa 2 7B	-	1,581,056	0.0226
	LLaMa 2 70B	-	8,355,840	0.0121
	LLaMa 3 8B	-	1,703,936	0.0212
DoRA	LLaMa 2 7B	$r = 8$	21,348,352	0.3158
		$r = 16$	41,336,832	0.6097
		$r = 64$	161,267,712	2.3373
		$r = 256$	640,991,232	9.1571

Table 4.3.: Trainable Parameters of different PEFT methods for different configuarations

In the experiments, LoRA & DoRA are applied to all linear layers (query, key, value, and feed-forward) in the transformer block of the model. As expected, the number of trainable parameters increases with the rank of the LoRA & DoRA configuration. This is because the rank directly influences the size of the Low-Rank Adaptation matrices, which in turn dictates the number of parameters that are fine-tuned. The alpha value does not affect the number of trainable parameters, as it influences the scaling rather than the dimensions of the matrices.

For example, at a rank of 8, the model has approximately 19.99 million trainable parameters, which represents only 0.2855% of the total model parameters. In contrast, a higher rank of 256 leads to 639.63 million trainable parameters, which is about 9.1376% of the model's parameters. The shows that even at the highest configuration tested, only 9% of the parameters are being fine-tuned, which is significantly lower compared to full model fine-tuning. This highlights the efficiency of LoRA in adapting large models without the need for extensive computational resources. DoRA adds on to LoRA in terms of the number of parameters by around 1.5M for every configuration for a 7B model. This increase is because apart from optimizing the LoRA components, DoRA also finetunes the magnitude vector of the pre-trained weights as explained in 3.5. With an even lower rank and applying LoRA or DoRA to only specific components of the transformer, the trainable parameters can be reduced. These findings also suggest that while increasing the rank improves the model's ability to learn from fine-tuning, it also comes with a higher memory cost. Therefore, a balance must be struck between achieving sufficient model adaptability and maintaining computational efficiency, especially in environments with limited resources.

The number of trainable parameters in IA3 are calculated based on the dimensions of the learned vectors it adds to the key, value, and Feed Forward layers of the transformer block. As shown in the table, the trainable parameters for IA3 across different models are significantly lower compared to LoRA. This difference is largely due to IA3's approach of using learned vectors rather than low-rank updates to a weight matrix, as in LoRA. This method keeps the number of trainable parameters much smaller. Additionally, while LoRA is applied across all linear layers in transformers, IA3 does not adjust the query layers, further reducing the number of trainable parameters. For instance, in the LLaMa 2 7B model, IA3 introduces just 1,581,056 trainable parameters, accounting for a mere 0.02% of the total model parameters (approximately 1.6M) whereas the same for LoRA with rank 8 is 0.2% (approximately 20M). This is in stark contrast to the much larger parameter counts associated with LoRA, highlighting IA3's efficiency.

4.7.3. PEFT Memory Overhead

This part focuses on the additional memory requirements for the PEFT methods. Table 4.4 provides insights into how the PEFT methods impact the size of the model. It reveals critical insights regarding the trade-offs between model size and the chosen PEFT configuration.

Focusing on LoRA and DoRA, Similar to the trainable parameters, it was observed that α does not affect the size. The model size increases with higher ranks. For the lowest rank, the additional memory required is only about 39 MB, which is negligible compared to the overall model size. This small increment is unlikely to affect fine-tuning on most hardware setups, making it a very efficient option when computational resources

PEFT Method	Model	Config	Post Quantization	+ PEFT	Difference
LoRA	LLaMa 2 7B	$r = 8$	3.51 GB	3.542 GB	38.648 MB
		$r = 16$	3.51 GB	3.579 GB	76.773 MB
		$r = 64$	3.51 GB	3.803 GB	305.523 MB
		$r = 256$	3.51 GB	4.696 GB	1220.523 MB
	LLaMa 2 70B	$r = 64$	32.85 GB	34.399 GB	1582.552 MB
IA3	LLaMa 2 7B	-	3.51 GB	3.5108 GB	6.554 MB
	LLaMa 2 70B	-	32.85 GB	32.89 GB	34.42 MB
	LLaMa 3 8B	-	5.21 GB	5.2144 GB	7.02 MB
DoRA	LLaMa 2 7B	$r = 8$	3.51 GB	3.545 GB	41.242 MB
		$r = 16$	3.51 GB	3.582 GB	79.367 MB
		$r = 64$	3.51 GB	3.805 GB	308.117 MB
		$r = 256$	3.51 GB	4.699 GB	1223.117 MB

Table 4.4.: Additional memory requirement of different PEFT methods for different configurations.

are constrained. The highest rank tested (256), results in a considerable memory overhead of 1221 MB. The memory requirements of DoRA closely match those of LoRA, with DoRA requiring only an additional of around 3 MB for each configuration As discussed, this extra 3 MB can be attributed to the additional magnitude vector of the pre-trained weights that DoRA fine-tunes. While this allows for more expressive adaptation during fine-tuning, the large memory footprint could pose challenges, particularly for environments with limited GPU memory. This investigation highlights that higher ranks result in much higher memory utilization. Selecting a lower rank could be required for applications with limited GPU RAM to prevent problems during fine-tuning.

When integrating IA3 layers into Quantized LLMs, the additional memory overhead remains minimal. For instance, when applied to a 70B parameter model, IA3 adds approximately 35 MB of memory, which is negligible relative to the overall model size. This efficiency is achieved because IA3 modifies only a few key parameters, leaving the vast majority of the model unchanged. As a result, the approach not only maintains the model's performance but also enables scalable deployment, particularly in memory-constrained environments.

Additionally, when these results are considered alongside model performance based on perplexity and downstream tasks, better decisions can be made regarding which configurations to use in different settings. This integrated approach enhances the overall

efficiency and effectiveness of fine-tuning strategies across various contexts. The subsequent sections will explore how these different PEFT configurations with Quantized models impact model performance, particularly using evaluation perplexity and 5-shot MMLU accuracy.

4.7.4. Evaluation Perplexity

In this subsection, we will briefly look at the evaluation perplexity of the experiments around different PEFT methods applied to Quantized LLMs. The outcome of these results will help us determine which combination of methods to use in different scenarios.

As outlined in section 4.5, Perplexity is a measure of how well a probability model predicts a sample. The basic intuition is that the lower the perplexity measure the better the language model is at modeling unseen sentences. In the experiments, perplexity serves as a metric for assessing fine-tuning performance across various methods, helping to identify optimal hyperparameter configurations. Each dataset was split into 80% training and 20% validation before training. The evaluation PPL is calculated on this 20% data which the model does not see during training. Table 4.5 shows the mean PPL across different datasets and methods.

	LLaMa 2 7B			LLaMa 3 8B		
Method	Unnatural	OASST	HH-RLHF	Unnatural	OASST	HH-RLHF
QLoRA	2.1424	11.1491	3.0468	2.373	13.7290	3.7129
Q-RsLoRA	2.2779	11.1123	2.9612	2.6201	12.4908	-
QLoRA + LoftQ	1.9432	10.3251	3.0988	-	-	-
Q-RsLoRA + LoftQ	2.1311	12.1469	2.9019	-	-	-
Q-IA3	1.5634	3.3831	4.1971	1.5776	3.576	5.4914
QDoRA	1.9866	9.1272	3.4329	-	-	-
QDoRA + RsLoRA	1.8669	10.6233	3.2737	-	-	-
QDoRA + LoftQ	1.693	10.5118	3.4153	-	-	-
HQQ-4bit-LoRA	1.6923	10.2474	-	-	-	-
HQQ-2bit-LoRA	1.887	15.4132	-	-	-	-

Table 4.5.: Mean Evaluation Perplexity for different combinations of PEFT and Quantization methods. The scores of QLoRA, QDoRA, and HQQ-LoRA are averages over different configurations whereas Q-IA3 is just a single configuration. The detailed results for each configuration can be found in the Appendix B.2.

Config	Model	Unnatural
QLoRA	LLaMa 2 70B	1.5344
Q-IA3	LLaMa 2 70B	1.4584

Table 4.6.: Evaluation Perplexity for QLoRA and QIA3 for LLaMa 2 70B model. For LoRA, $r = 64$ and $\alpha = 16$

For LLaMa 2 7B, the results indicate that validation perplexity is the lowest for Q-IA3 while it remains relatively consistent across all other methods with a variation of around ± 0.2 for Unnatural and HH-RLHF, while the same being around ± 1.0 for OASST. Notably, the OASST dataset exhibits higher perplexity compared to the other two datasets. A reason for this is the dataset size and quality. As we mentioned in section 4.2, OASST is significantly smaller than other datasets. Additionally, we don't take in entire conversations—just the initial replies. This affects the quality of the dataset. The IA3 method, known for its parameter efficiency, shows significant improvements, particularly with OASST, achieving a 67% reduction in perplexity for LLaMa2 compared to the combination of QLoRA and LoftQ. This highlights IA3's ability to effectively fine-tune models with less data, making it suitable for tasks where data is limited. As we scale up to 70B models in Table 4.6, the results are similar to as seen in 7B models.

Q-DoRA provides good competition to QLoRA while giving consistent perplexity scores across all 3 datasets. As RsLoRA is added to this mix, the perplexity worsens for OASST indicating a sensitivity to smaller datasets. As observed with LoRA, the addition of LoftQ contributes to improvement with DoRA as well.

HQQ in its 4-bit configuration offers strong competition to NF4+DQ, with HQQ-4bit-LoRA achieving better perplexity scores than QLoRA. However, the 2-bit version shows a performance drop, particularly on the OASST dataset. This suggests that fine-tuning in a 2-bit scenario may require more high-quality data to mitigate the increased quantization error and improve results.

Although we have fewer results for LLaMa 3, they show higher values as compared to LLaMa 2. This difference is unexpected, as one might assume that LLaMa 3 would outperform its predecessor. However, this could be explained by the higher quantization error observed in LLaMa 3, potentially contributing to the increased perplexity during evaluation, emphasizing the impact of quantization degradation.

Having examined various aspects so far, the next step is to evaluate the performance of the fine-tuned quantized models on a downstream task. This evaluation will provide insight into how these models perform in real-world applications.

4.7.5. Downstream Task - MMLU

The most important aspect of fine-tuning a model is to evaluate its performance on unseen downstream tasks. The MMLU benchmark, as mentioned in section 4.5, is widely used to evaluate LLMs. Table 4.7 displays the 5-shot MMLU results from the original QLoRA paper and figure 4.1, table 4.8 and table 4.9 display the 5-shot MMLU results for the experiments of this thesis.

Model	No fine-tuning	Unnatural	OASST	HH-RLHF
LLaMa 7B	35.1	41.9	36.6	34.9
LLaMa 65B	63.4	61.3	62.2	60.1

Table 4.7.: 5-Shot MMLU Accuracy of LLaMA finetuned on the corresponding datasets using QLoRA. Config used $r = 64$, $\alpha = 16$. Table and caption adapted from (Dettmers et al. 2024)

	LLaMa 2 7B			LLaMa 3 8B		
No fine-tuning	40.68			53.86		
Method	Unnatural	OASST	HH-RLHF	Unnatural	OASST	HH-RLHF
QLoRA	46.70	44.29	38.37	57.13	59.18	53.98
Q-RsLoRA	37.01	38.54	32.25	38.85	46.67	-
Q-RsLoRA*	42.74	42.82	34.69	47.43	55.39	-
QLoRA + LoftQ	47.57	43.86	40.16	-	-	-
Q-RsLoRA + LoftQ	41.73	42.41	34.37	-	-	-
Q-IA3	47.17	45.18	41.78	61.16	59.93	61.42
QDoRA	47.29	42.69	39.1	-	-	-
QDoRA + RsLoRA	41.95	42.07	36.12	-	-	-
QDoRA + LoftQ	47.24	44.64	39.99	-	-	-
HQQ-4bit-LoRA	44.84	43.87	-	-	-	-
HQQ-2bit-LoRA	31.43	25.1	-	-	-	-

Table 4.8.: Mean 5-shot MMLU Accuracy of fine-tuned quantized LLaMa Models for different combinations PEFT and Quantization methods. The first row shows the 5-shot MMLU Accuracy of the quantized models without fine-tuning. The row RsLoRA* excludes the configs of RsLoRA which were unstable. The detailed results for each configuration can be found in the Appendix B.2.

Config	Model	Unnatural
No fine-tuning	LLaMa 2 70B	67.12
QLoRA	LLaMa 2 70B	64.61
Q-IA3	LLaMa 2 70B	67.17

Table 4.9.: 5-shot MMLU Test Accuracy for QLoRA and QIA3 for LLaMa 2 70B model. For LoRA, $r = 64$ and $\alpha = 16$

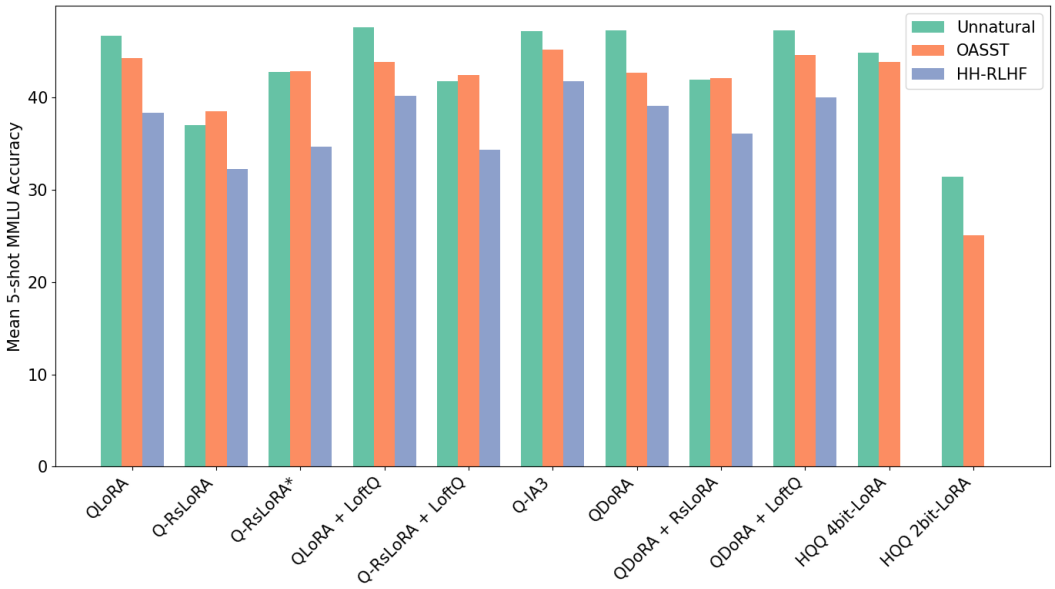

Figure 4.1.: Mean 5-shot MMLU Accuracy of fine-tuned quantized LLaMa Models for different combinations PEFT and Quantization methods. The first row shows the 5-shot MMLU Accuracy of the quantized models without fine-tuning.

QLoRA, RsLoRA, and LoftQ: The mean MMLU accuracy depicts a high-level overview of the performance of models in downstream tasks using different LoRA variations. Comparing each method with the base model without fine-tuning, shows that most of the methods with proper configurations tend to improve the results.

The results align with those from the official QLoRA experiments. For 7B model size, the QLoRA authors reported a significant improvement on the Unnatural Instructions dataset compared to the baseline without fine-tuning, a marginal improvement on OASST, and a slight decline on HH-RLHF. Shifting focus to the largest model size, a decline in MMLU performance was observed when using QLoRA. This could be due

to the large models having absorbed vast amounts of information, making them more sensitive to fine-tuning. This thesis observes a similar pattern of impact across these datasets.

For LLaMa 2, RsLoRA had a huge drop in accuracy while using high rank with high alpha, this can be seen in table B.3. Excluding these results from the mean shows that even RsLoRA enhances the model to perform better on downstream tasks (RsLoRA*). On the other hand, a drop in scores for LLaMa 3 is observed in RsLoRA. This can again be attributed to the fact that LLaMa 3 is very redundant to changes. Fine-tuning LLaMa 3 with more information and high-rank adapters doesn't lead to major improvements in performance.

Performance variations across the three different datasets are evident. For both models, the HH-RLHF dataset seems to contribute minimally to improving MMLU scores, with most scenarios even showing a decline. A possible reason for this could be that the LLaMa models might have already been exposed to similar types of data during their pre-training. HH-RLHF dataset is used to train models to give clean, healthy, and non-abusive responses. This general-purpose nature may result in less effective enhancement of the specific skills measured by MMLU.

Among all the different combinations, LoRA when applied with LoftQ initialization performs the best on average. It had already been established by various researchers that QLoRA was the state-of-the-art technique to fine-tune models in low-bit and in a parameter-efficient way. Combining this with an initialization that is aimed at reducing quantization error improves performance.

Q-IA3: The analysis now shifts to the evaluation of IA3. Despite the minimal number of parameters involved and negligible memory overhead, it remains crucial to evaluate whether fine-tuning such a small set of parameters can still yield effective results. Similar to perplexity, the 5-Shot MMLU Accuracy Scores of IA3 are compared to the other combinations. It is observed, that IA3 performs excellently. IA3 demonstrates robust performance across datasets, showing significant improvement over the no-fine-tuning baseline. Particularly for LLaMa 2, the MMLU accuracy of IA3 on the Unnatural Instructions dataset is consistent with the QLoRA + LoftQ combination. However, IA3 surpasses this combination on the OASST and HH-RLHF datasets, showcasing its adaptability and strength across diverse tasks. Similarly, when applied to LLaMa3, IA3 achieves impressive accuracy, outperforming other configurations and further emphasizing its effectiveness in fine-tuning large models. Even after scaling to a larger model of LLaMa 2 with 70B parameters, IA3 performed better with a minimal number of parameters to fine-tune.

IA3 continues to stand out by requiring the fewest parameters for fine-tuning while

maintaining strong performance, as evidenced by both MMLU accuracy and validation perplexity comparisons. Given these findings, IA3 emerges as the optimal choice among the evaluated PEFT methods seen so far, offering a compelling balance of efficiency and accuracy.

Q-DoRA: As observed with perplexity, Q-DoRA also demonstrates consistent performance on MMLU. However, it falls slightly short in the small dataset scenario of OASST. When combined with rank-stabilizing LoRA, Q-DoRA significantly impacts MMLU performance on the Unnatural Instructions and HH-RLHF datasets. This suggests that the rank-stabilizing factor is not a favorable combination with DoRA.

HQQ: In terms of downstream tasks, HQQ-4bit demonstrates consistent performance compared to NF4+DQ. Although the 2-bit version significantly reduces memory usage, it results in a substantial drop in scores, likely due to increased quantization error. This outcome indicates that further research is needed to enhance performance in lower-bit quantization scenarios.

4.7.6. Memory Footprint

Quantization	PEFT Method	LLaMa 2 7B	LLaMa 2 70B
NF4+DQ	LoRA	7.43 GB	48.48 GB
	RsLoRA	7.43 GB	-
	LoRA + LoftQ	7.43 GB	-
	RsLoRA + LoftQ	7.43 GB	-
	DoRA	8.97 GB	-
	DoRA + RsLoRA	8.97 GB	-
	DoRA + LoftQ	9.02 GB	-
	IA3	7.63 GB	44.58 GB
HQQ-4bit	LoRA	7.34 GB	-
HQQ-2bit	LoRA	5.99 GB	-

Table 4.10.: LLaMa 2 - Average GPU Consumption for different techniques over 500 steps. For LoRA-related methods, the rank used is 64 and alpha 16.

The average memory footprint for all different Q-PEFT training experiments using LLaMA 2 as the base model can be seen in table 4.10. Since all experiments were running on shared servers it was quite difficult to keep track of the memory footprint of a specific training run. Hence, the results show the average consumption of over 500 steps of fine-tuning on Unnatural core for different configurations on Nvidia RTX A6000 (48 GB) GPUs.

While using the NF4 and Double Quantization techniques, all the variations of the LoRA experiments have the same memory footprint. This shows that RsLoRA and LoftQ do not add any overhead consumption. For LLaMa 2 7B, DoRA has around 1.5 GB of additional memory footprint when compared to LoRA. This can be attributed to the additional magnitude vector of the decomposed pre-trained model which is tuned in DoRA as explained in 3.5. IA3 has a memory consumption of 7.63 GB, comparable to LoRA. However, with a 70B model, LoRA incurs a significantly higher memory overhead, exceeding IA3 by 3 GB. Additionally, it is seen that tuning a 70B model with QLoRA would need a slightly larger GPU. The comparison between HQQ-4bit and NF4 + DQ shows minimal difference in memory footprint. As anticipated, HQQ-2bit requires even less memory.

4.7.7. Average Runtime

The average runtime for all different Q-PEFT training experiments using LLaMA 2 as the base model can be seen in table 4.11. The data reflects the average runtime over 500 fine-tuning steps on the Unnatural Instructions dataset across different configurations. These experiments were conducted on Nvidia RTX A6000 (48 GB) GPUs.

Quantization	PEFT Method	LLaMa 2 7B	LLaMa 2 70B
NF4+DQ	LoRA	2h 15m	18h 40m
	RsLoRA	2h 20m	-
	LoRA + LoftQ	2h 25m	-
	RsLoRA + LoftQ	2h 30m	-
	DoRA	5h 30m	-
	DoRA + RsLoRA	5h 30m	-
	DoRA + LoftQ	5h 40m	-
	IA3	2h	17h 10m
HQQ-4bit	LoRA	2h 15m	-
HQQ-2bit	LoRA	2h 15m	-

Table 4.11.: LLaMa 2 - Average Runtime for different techniques over 500 steps. For LoRA-related methods, the rank used is 64 and alpha 16.

For the 7B models, LoRA and IA3 completed 500 steps in approximately 2 hours. RsLoRA required just 5 minutes longer, while LoftQ added 10 minutes due to its initial weight initialization to reduce quantization error. DoRA, however, was the slowest, taking more than twice the time of the other methods. One possible reason could be that the decomposition of the weight matrices into magnitude and direction and the fine-tuning of them could require much more computing time. This raises concerns,

as scaling up for more steps could make DoRA excessively time-consuming, potentially making full model fine-tuning a more practical option. For the 70B model, LoRA requires an additional hour and a half compared to IA3 to complete 500 steps. When scaled to larger models and more steps, IA3 demonstrates greater efficiency due to the smaller size of the parameters it tunes.

The table also suggests that the choice of quantization technique does not significantly impact runtime, indicating that the added computational overhead from different quantization methods is minimal.

4.7.8. Summary

Having explored the various aspects of the results from different combinations of Quantization and PEFT methods, it is evident that while QLoRA remains state-of-the-art, other methods also demonstrate comparable performance. Q-IA3, in particular, emerges as a strong competitor. In our experiments, Q-IA3 exhibited fewer trainable parameters, lower memory overhead, consistent perplexity, better MMLU scores, and a similar memory footprint and runtime compared to LoRA, making it a viable alternative for efficient fine-tuning.

On the quantization side, HQQ-4bit also shows promise as an alternative to NF4+DQ, delivering solid performance. However, while HQQ-2bit allows models to be quantized to even smaller sizes, the significant drop in MMLU scores indicates that further research is necessary to optimize low-bit quantization methods. This suggests that while HQQ-2bit offers potential benefits in reducing model size, its impact on performance needs to be carefully addressed before it can be considered a reliable option.

5. Conclusion

5.1. Summary

In this thesis, the primary objective was to optimize and critically evaluate LLMs through quantization and PEFT methods. The aim was to replicate and extend QLoRA's success and assess its effectiveness and performance across different PEFT strategies and LLMs.

To accomplish this, the LLaMa series of models was selected as the base models for fine-tuning. Different datasets representing different domains of Natural Language Generation tasks were then selected. Following this, a comprehensive review of Quantization and PEFT methods was conducted, leading to the selection of a subset of methods for experimentation. A codebase and repository were set up for the experiments. Initial small-scale experiments were conducted to determine the best set of hyperparameter configurations for different fine-tuning combinations. Following this, the LLaMa models were fine-tuned using various combinations of Quantization and PEFT methods. These combinations were critically evaluated based on several criteria, including the impact of quantization, the number of trainable parameters, memory overhead, memory footprint, runtime, evaluation perplexity, and performance on a standardized downstream task.

The experiments successfully replicated results as observed in (Dettmers et al. 2024). The experimental results highlight why QLoRA is considered state-of-the-art for efficiently fine-tuning large language models. QLoRA performed exceptionally well across all evaluated criteria, including evaluation perplexity and consistent results on the MMLU task. Additionally, it demonstrated strong performance in terms of memory requirements and runtime, making it a highly effective and resource-efficient method for fine-tuning. The addition of LoftQ to QLoRA slightly enhanced its performance by focusing on minimizing the initial quantization error. This improvement resulted in better accuracy and stability during fine-tuning, making the combination of QLoRA and LoftQ even more effective across various tasks and evaluation metrics.

Q-IA3 consistently matched QLoRA's performance, while improving the performance in terms of MMLU. For example, on the LLaMa 3 8B model, Q-IA3 achieved scores of 61.16 on the Unnatural dataset, 59.93 on OASST, and 61.42 on HHRLHF, which are either on par with or slightly better than the QLoRA results. Q-IA3 delivered top results across all datasets while requiring the fewest trainable parameters. Its memory footprint

and runtime were nearly identical to those of QLoRA. IA3, while promising, has not yet reached the popularity of LoRA and may require further research, practical implementation, and optimization to become a stronger competitor. Its unique advantages, such as lower parameter requirements, highlight its potential, but broader adoption and more real-world applications are needed to establish its viability as a mainstream fine-tuning method for large language models.

Q-DoRA also delivered consistent results compared to QLoRA, but its efficiency is questionable due to the training time, which was more than twice as long. This extended runtime makes Q-DoRA less practical for scenarios where quick fine-tuning is essential, despite its strong performance across various tasks.

For a long time, Normal Float 4 and Double Quantization have been the state-of-the-art techniques. However, this thesis has observed that HQQ also delivers consistent results, suggesting that further research into lower-bit scenarios could enhance this technique. Although not tested in this thesis, HQQ also offers the flexibility to quantize each layer or module with a different configuration.

Overall, the integration of parameter-efficient fine-tuning techniques (PEFTs) with quantization significantly enhances the efficiency of large language models, making them more accessible and practical for a wider range of users. These methods reduce the number of trainable parameters, lower memory requirements, and maintain strong performance across various tasks, striking a balance between computational demands and model effectiveness. As these approaches continue to evolve, they hold the potential to further democratize AI technology and drive innovation across diverse fields.

Limitations: The use of shared GPUs on the server sometimes impacted access and computational power. Dedicated CPUs/GPUs would enable more consistent and extensive experimentation. Additionally, storage limitations on the servers hindered the ability to download larger models.

5.2. Future Work

While this thesis provides a valuable exploration of a few Quantization and PEFT combinations, it only begins to explore the depth of this domain. As this work was being written, new Quantization and PEFT methods continued to emerge, some of which were discussed in the background sections (2.5 and 2.6). These individual methods must be tested as combinations for improvements in fine-tuning and deploying LLMs in a low-resource environment. Additionally, greater emphasis could be placed on testing these method combinations with even larger models, provided sufficient resources are avail-

able. Additionally, exploring HQQ's flexibility to quantize each layer or module with different configurations could be a promising area for future research.

Moving forward, a key area for improvement is the evaluation of these fine-tuned LLMs across a diverse range of downstream tasks. Given the variety of domains in which LLMs are fine-tuned, it is essential to evaluate them on tasks that closely align with the specific purposes for which they were fine-tuned.

Bibliography

Achiam, J., S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

An, S., Y. Li, Z. Lin, Q. Liu, B. Chen, Q. Fu, W. Chen, N. Zheng, and J.-G. Lou (2022). Input-tuning: Adapting unfamiliar inputs to frozen pretrained models. *arXiv preprint arXiv:2203.03131*.

Badri, H. and A. Shaji (2023, November). Half-quadratic quantization of large machine learning models.

Bai, Y., A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan, N. Joseph, S. Kadavath, J. Kernion, T. Conerly, S. E. Showk, N. Elhage, Z. Hatfield-Dodds, D. Hernandez, T. Hume, S. Johnston, S. Kravec, L. Lovitt, N. Nanda, C. Olsson, D. Amodei, T. B. Brown, J. Clark, S. McCandlish, C. Olah, B. Mann, and J. Kaplan (2022). Training a helpful and harmless assistant with reinforcement learning from human feedback. *CoRR abs/2204.05862*.

Bengio, Y., R. Ducharme, and P. Vincent (2000). A neural probabilistic language model. *Advances in neural information processing systems 13*.

Bengio, Y., P. Simard, and P. Frasconi (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks 5*(2), 157–166.

Brown, P. F., V. J. Della Pietra, P. V. Desouza, J. C. Lai, and R. L. Mercer (1992). Class-based n-gram models of natural language. *Computational linguistics 18*(4), 467–480.

Brown, T., B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems 33*, 1877–1901.

Brown, T. B., B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei (2020). Language models are few-shot learners. In

Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Chang, Y., X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, et al. (2024). A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology 15*(3), 1–45.

Chen, B., T. Dao, E. Winsor, Z. Song, A. Rudra, and C. Ré (2021). Scatterbrain: Unifying sparse and low-rank attention. *Advances in Neural Information Processing Systems 34*, 17413–17426.

Chen, S. F. and J. Goodman (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language 13*(4), 359–394.

Chen, T., T. Ding, B. Yadav, I. Zharkov, and L. Liang (2023). Lorashear: Efficient large language model structured pruning and knowledge recovery. *ArXiv abs/2310.18356*.

Chen, X., T. Chen, W. Chen, A. H. Awadallah, Z. Wang, and Y. Cheng (2023, July). DSEE: Dually sparsity-embedded efficient tuning of pre-trained language models. In A. Rogers, J. Boyd-Graber, and N. Okazaki (Eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Toronto, Canada, pp. 8208–8222. Association for Computational Linguistics.

Chen, X., N. Zhang, X. Xie, S. Deng, Y. Yao, C. Tan, F. Huang, L. Si, and H. Chen (2022). Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. In *Proceedings of the ACM Web conference 2022*, pp. 2778–2788.

Chen, Y., S. Qian, H. Tang, X. Lai, Z. Liu, S. Han, and J. Jia (2023). Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*.

Cho, K., B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Dettmers, T., M. Lewis, Y. Belkada, and L. Zettlemoyer (2022a). Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems 35*, 30318–30332.

Dettmers, T., M. Lewis, Y. Belkada, and L. Zettlemoyer (2022b). Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*.

Dettmers, T., M. Lewis, S. Shleifer, and L. Zettlemoyer (2022). 8-bit optimizers via block-wise quantization.

Dettmers, T., A. Pagnoni, A. Holtzman, and L. Zettlemoyer (2024). Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems 36*.

Dettmers, T. and L. Zettlemoyer (2023). The case for 4-bit precision: k-bit inference scaling laws. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ding, N., Y. Qin, G. Yang, F. Wei, Z. Yang, Y. Su, S. Hu, Y. Chen, C.-M. Chan, W. Chen, et al. (2023). Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence 5*(3), 220–235.

Frantar, E., S. Ashkboos, T. Hoefler, and D. Alistarh (2023). OPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*.

Fu, Z., H. Yang, A. M.-C. So, W. Lam, L. Bing, and N. Collier (2023). On the effectiveness of parameter-efficient fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Volume 37, pp. 12799–12807.

Gao, Z.-F., P. Liu, W. X. Zhao, Z.-Y. Lu, and J.-R. Wen (2022, October). Parameter-efficient mixture-of-experts architecture for pre-trained language models. In *Proceedings of the 29th International Conference on Computational Linguistics*, Gyeongju, Republic of Korea, pp. 3263–3273. International Committee on Computational Linguistics.

Grefenstette, E., P. Blunsom, et al. (2014). A convolutional neural network for modelling sentences. In *The 52nd Annual Meeting of the Association for Computational Linguistics, Baltimore, Maryland*, Volume 1, pp. 655–665.

Guo, H., P. Greengard, E. Xing, and Y. Kim (2024). LQ-loRA: Low-rank plus quantized matrix decomposition for efficient language model finetuning. In *The Twelfth International Conference on Learning Representations*.

Han, S., H. Mao, and W. J. Dally (2016). Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In Y. Bengio and Y. LeCun (Eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Hochreiter, S. and J. Schmidhuber (1997). Long short-term memory. *Neural computation 9*(8), 1735–1780.

Honovich, O., T. Scialom, O. Levy, and T. Schick (2023, July). Unnatural instructions: Tuning language models with (almost) no human labor. In A. Rogers, J. Boyd-Graber, and N. Okazaki (Eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Toronto, Canada, pp. 14409–14428. Association for Computational Linguistics.

Houlsby, N., A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly (2019, 09–15 Jun). Parameter-efficient transfer learning for NLP. In K. Chaudhuri and R. Salakhutdinov (Eds.), *Proceedings of the 36th International Conference on Machine Learning*, Volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799. PMLR.

Howard, J. and S. Ruder (2018). Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Hu, E. J., Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Huang, W., X. Ma, H. Qin, X. Zheng, C. Lv, H. Chen, J. Luo, X. Qi, X. Liu, and M. Magno (2024). How good are low-bit quantized llama3 models? an empirical study. *CoRR abs/2404.14047*.

Jacob, B., S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2704–2713.

Jeon, H., Y. Kim, and J. joon Kim (2024). L4q: Parameter efficient quantization-aware fine-tuning on large language models.

Kalajdzievski, D. (2023). A rank stabilization scaling factor for fine-tuning with lora. *arXiv preprint arXiv:2312.03732*.

Kaplan, J., S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Karimi Mahabadi, R., J. Henderson, and S. Ruder (2021). Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems 34*, 1022–1035.

Karimi Mahabadi, R., S. Ruder, M. Dehghani, and J. Henderson (2021). Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Annual Meeting of the Association for Computational Linguistics*.

Kim, J., J. H. Lee, S. Kim, J. Park, K. M. Yoo, S. J. Kwon, and D. Lee (2024). Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization. *Advances in Neural Information Processing Systems 36*.

Kim, M., S. Lee, S.-J. Hong, D.-S. Chang, and J. Choi (2022). Understanding and improving knowledge distillation for quantization aware training of large transformer encoders. In Y. Goldberg, Z. Kozareva, and Y. Zhang (Eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Abu Dhabi, United Arab Emirates, pp. 6713–6725. Association for Computational Linguistics.

Kim, Y. (2014, October). Convolutional neural networks for sentence classification. In A. Moschitti, B. Pang, and W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, pp. 1746–1751. Association for Computational Linguistics.

Koohpayegani, S. A., K. Navaneet, P. Nooralinejad, S. Kolouri, and H. Pirsiavash. Nola: Compressing lora using linear combination of random basis. In *The Twelfth International Conference on Learning Representations*.

Köpf, A., Y. Kilcher, D. von Rütte, S. Anagnostidis, Z. R. Tam, K. Stevens, A. Barhoum, D. Nguyen, O. Stanley, R. Nagyfi, et al. (2024). Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems 36*.

Kopiczko, D. J., T. Blankevoort, and Y. M. Asano (2024). VeRA: Vector-based random matrix adaptation. In *The Twelfth International Conference on Learning Representations*.

Kummer, L., K. Sidak, T. Reichmann, and W. Gansterer (2023). *Adaptive Precision Training (AdaPT): A dynamic quantized training approach for DNNs*, pp. 559–567.

Lester, B., R. Al-Rfou, and N. Constant (2021). The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Li, X. L. and P. Liang (2021). Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Li, Y., Y. Yu, C. Liang, N. Karampatziakis, P. He, W. Chen, and T. Zhao (2024). Loftq: LoRA-fine-tuning-aware quantization for large language models. In *The Twelfth International Conference on Learning Representations*.

Li, Z., X. Liu, B. Zhu, Z. Dong, Q. Gu, and K. Keutzer (2024). QFT: Quantized full-parameter tuning of LLMs with affordable resources.

Lian, D., D. Zhou, J. Feng, and X. Wang (2022). Scaling & shifting your features: A new baseline for efficient model tuning. *Advances in Neural Information Processing Systems 35*, 109–123.

Lin, J., J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han (2024). Awq: Activation-aware weight quantization for llm compression and acceleration. In *MLSys*.

Lin, Y., Y. Li, T. Liu, T. Xiao, T. Liu, and J. Zhu (2021). Towards fully 8-bit integer inference for the transformer model. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20.

Liu, H., D. Tam, M. Muqeeth, J. Mohta, T. Huang, M. Bansal, and C. A. Raffel (2022). Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems 35*, 1950–1965.

Liu, J., R. Gong, X. Wei, Z. Dong, J. Cai, and B. Zhuang (2024). QLLM: Accurate and efficient low-bitwidth quantization for large language models. In *The Twelfth International Conference on Learning Representations*.

Liu, S.-y., Z. Liu, X. Huang, P. Dong, and K.-T. Cheng (2023). Llm-fp4: 4-bit floating-point quantized transformers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Liu, S.-Y., C.-Y. Wang, H. Yin, P. Molchanov, Y.-C. F. Wang, K.-T. Cheng, and M.-H. Chen (2024). DoRA: Weight-decomposed low-rank adaptation.

Liu, W., Z. Qiu, Y. Feng, Y. Xiu, Y. Xue, L. Yu, H. Feng, Z. Liu, J. Heo, S. Peng, Y. Wen, M. J. Black, A. Weller, and B. Schölkopf (2024). Parameter-efficient orthogonal finetuning via butterfly factorization. In *ICLR*.

Liu, Y., M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ma, Y., H. Li, X. Zheng, F. Ling, X. Xiao, R. Wang, S. Wen, F. Chao, and R. Ji (2024). Affinequant: Affine transformation quantization for large language models. In *The Twelfth International Conference on Learning Representations*.

Markov, I., A. Vladu, Q. Guo, and D. Alistarh (2023). Quantized distributed training of large models with convergence guarantees. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Micikevicius, P., S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu (2018). Mixed precision training. In *International Conference on Learning Representations*.

Mikolov, T., M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur (2010). Recurrent neural network based language model. In *Interspeech*, Volume 2, pp. 1045–1048. Makuhari.

Min, S., X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer (2022, December). Rethinking the role of demonstrations: What makes in-context learning work? In Y. Goldberg, Z. Kozareva, and Y. Zhang (Eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Abu Dhabi, United Arab Emirates, pp. 11048–11064. Association for Computational Linguistics.

Narayanan, D., M. Shoeybi, J. Casper, P. LeGresley, M. Patwary, V. Korthikanti, D. Vainbrand, P. Kashinkunti, J. Bernauer, B. Catanzaro, et al. (2021). Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–15.

Ouyang, L., J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. (2022). Training language models to follow instructions with human feedback. *Advances in neural information processing systems 35*, 27730–27744.

Pan, S. J. and Q. Yang (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering 22*(10), 1345–1359.

Park, G., B. park, M. Kim, S. Lee, J. Kim, B. Kwon, S. J. Kwon, B. Kim, Y. Lee, and D. Lee (2024). LUT-GEMM: Quantized matrix multiplication based on LUTs for efficient inference in large-scale generative language models. In *The Twelfth International Conference on Learning Representations*.

Polino, A., R. Pascanu, and D. Alistarh (2018). Model compression via distillation and quantization. In *International Conference on Learning Representations*.

Radford, A., K. Narasimhan, T. Salimans, I. Sutskever, et al. (2018). Improving language understanding by generative pre-training.

Radford, A., J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog 1*(8), 9.

Sanh, V., A. Webson, C. Raffel, S. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, A. Raja, M. Dey, M. S. Bari, C. Xu, U. Thakker, S. S. Sharma, E. Szczechla, T. Kim, G. Chhablani, N. Nayak, D. Datta, J. Chang, M. T.-J. Jiang, H. Wang, M. Manica, S. Shen, Z. X. Yong, H. Pandey, R. Bawden, T. Wang, T. Neeraj, J. Rozen, A. Sharma, A. Santilli, T. Fevry, J. A. Fries, R. Teehan, T. L. Scao, S. Biderman, L. Gao, T. Wolf, and A. M. Rush (2022). Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

Scialom, T., T. Chakrabarty, and S. Muresan (2022, December). Fine-tuned language models are continual learners. In Y. Goldberg, Z. Kozareva, and Y. Zhang (Eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Abu Dhabi, United Arab Emirates, pp. 6107–6122. Association for Computational Linguistics.

Shen, S., Z. Dong, J. Ye, L. Ma, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer (2020). Q-bert: Hessian based ultra low precision quantization of bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Volume 34, pp. 8815–8821.

Shi, Z. and A. Lipani (2024). DePT: Decomposed prompt tuning for parameter-efficient fine-tuning. In *The Twelfth International Conference on Learning Representations*.

Sun, C., X. Qiu, Y. Xu, and X. Huang (2019). How to fine-tune bert for text classification? In *Chinese computational linguistics: 18th China national conference, CCL 2019, Kunming, China, October 18–20, 2019, proceedings 18*, pp. 194–206. Springer.

Tao, C., L. Hou, W. Zhang, L. Shang, X. Jiang, Q. Liu, P. Luo, and N. Wong (2022, May). Compression of generative pre-trained language models via quantization. In S. Muresan, P. Nakov, and A. Villavicencio (Eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Dublin, Ireland, pp. 4821–4836. Association for Computational Linguistics.

Tayaranian Hosseini, M., A. Ghaffari, M. S. Tahaei, M. Rezagholizadeh, M. Asgharian, and V. Partovi Nia (2023, May). Towards fine-tuning pre-trained language models with integer forward and backward propagation. In A. Vlachos and I. Augenstein (Eds.), *Findings of the Association for Computational Linguistics: EACL 2023*, Dubrovnik, Croatia, pp. 1912–1921. Association for Computational Linguistics.

Team, G., T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, P. Tafti, L. Hussenot, P. G. Sessa, A. Chowdhery, A. Roberts, A. Barua, A. Botev, A. Castro-Ros, A. Slone, A. Héliou, A. Tacchetti, A. Bulanova, A. Paterson, B. Tsai, B. Shahriari, C. L. Lan, C. A. Choquette-Choo, C. Crepy, D. Cer, D. Ippolito, D. Reid, E. Buchatskaya, E. Ni, E. Noland, G. Yan, G. Tucker, G.-C. Muraru, G. Rozhdestvenskiy, H. Michalewski, I. Tenney, I. Grishchenko, J. Austin, J. Keeling, J. Labanowski, J.-B. Lespiau, J. Stanway, J. Brennan, J. Chen, J. Ferret, J. Chiu, J. Mao-Jones, K. Lee, K. Yu, K. Millican, L. L. Sjoesund, L. Lee, L. Dixon, M. Reid, M. Mikuła, M. Wirth, M. Sharman, N. Chinaev, N. Thain, O. Bachem, O. Chang, O. Wahltinez, P. Bailey, P. Michel, P. Yotov, R. Chaabouni, R. Comanescu, R. Jana, R. Anil, R. McIlroy, R. Liu, R. Mullins, S. L. Smith, S. Borgeaud, S. Girgin, S. Douglas, S. Pandya, S. Shakeri, S. De, T. Klimenko, T. Hennigan, V. Feinberg, W. Stokowiec, Y. hui Chen, Z. Ahmed, Z. Gong, T. Warkentin, L. Peran, M. Giang, C. Farabet, O. Vinyals, J. Dean, K. Kavukcuoglu, D. Hassabis, Z. Ghahramani, D. Eck, J. Barral, F. Pereira, E. Collins, A. Joulin, N. Fiedel, E. Senter, A. Andreev, and K. Kenealy (2024). Gemma: Open models based on gemini research and technology.

Touvron, H., T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample (2023). Llama: Open and efficient foundation language models.

Üstün, A., A. Bisazza, G. Bouma, G. van Noord, and S. Ruder (2022). Hyper-x: A unified hypernetwork for multi-task multilingual transfer. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Valipour, M., M. Rezagholizadeh, I. Kobyzev, and A. Ghodsi (2022). Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. *arXiv preprint arXiv:2210.07558*.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin (2017). Attention is all you need. *Advances in neural information processing systems 30*.

Wang, X., T. Chen, Q. Ge, H. Xia, R. Bao, R. Zheng, Q. Zhang, T. Gui, and X. Huang (2023, December). Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, Singapore, pp. 10658–10671. Association for Computational Linguistics.

Wei, J., M. Bosma, V. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le (2022). Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Wei, J., Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, et al. (2022). Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Wu, X., C. Li, R. Y. Aminabadi, Z. Yao, and Y. He (2023). Understanding int4 quantization for language models: latency speedup, composability, and failure cases. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Xu, Y., Y. Li, S. Zhang, W. Wen, B. Wang, W. Dai, Y. Qi, Y. Chen, W. Lin, and H. Xiong (2019). Trained rank pruning for efficient deep neural networks. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*, pp. 14–17. IEEE.

Xu, Y., L. Xie, X. Gu, X. Chen, H. Chang, H. Zhang, Z. Chen, X. ZHANG, and Q. Tian (2024). QA-loRA: Quantization-aware low-rank adaptation of large language models. In *The Twelfth International Conference on Learning Representations*.

Yang, X., J. Y. Huang, W. Zhou, and M. Chen (2023). Parameter-efficient tuning with special token adaptation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 865–872.

Yao, Z., R. Yazdani Aminabadi, M. Zhang, X. Wu, C. Li, and Y. He (2022). Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), *Advances in Neural Information Processing Systems*, Volume 35, pp. 27168–27183. Curran Associates, Inc.

Yuan, Z., Y. Shang, and Z. Dong (2024). PB-LLM: Partially binarized large language models. In *The Twelfth International Conference on Learning Representations*.

Zafrir, O., G. Boudoukh, P. Izsak, and M. Wasserblat (2019, December). Q8bert: Quantized 8bit bert. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC2-NIPS)*. IEEE.

Zaken, E. B., S. Ravfogel, and Y. Goldberg (2021). Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*.

Zhang, M., H. Chen, C. Shen, Z. Yang, L. Ou, X. Yu, and B. Zhuang (2024). LoRAPrune: Pruning meets low-rank parameter-efficient fine-tuning.

Zhao, Y., C.-Y. Lin, K. Zhu, Z. Ye, L. Chen, S. Zheng, L. Ceze, A. Krishnamurthy, T. Chen, and B. Kasikci (2024). Atom: Low-bit quantization for efficient and accurate llm serving. In P. Gibbons, G. Pekhimenko, and C. D. Sa (Eds.), *Proceedings of Machine Learning and Systems*, Volume 6, pp. 196–209.

Zhu, Z., Y. Dong, and Z. Zhao (2023). Learning low-rank representations for model compression. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9. IEEE.

A. Program Code / Resources

The entire source code of the thesis can be found in the GitHub Repository - https://github.com/Sharan1712/quantization_peft

The official QLoRA source code was used as the base and it was updated to facilitate other PEFT and quantization techniques that were needed for the thesis.

The repository consists of a Readme.Md file which has the details of replication.

B. Additional Experimental Results

This chapter consists of extra details related to the experiments conducted in the thesis.

B.1. Hyperparameter Settings

The table B.1 focuses on the hyperparameter configurations for different fine-tuning runs for different LLMs and datasets. Most of the settings are directly replicated from the QLoRA paper and source code.

	LLaMa 2 7B/LLaMa 2 8B			LLaMa 2 70B
Hyperparameter	Unnatural	OASST	HH-RLHF	Unnatural
Batch Size	8	32	16	8
Steps	3000	1000	3000	2000
Max Source Length	384	-	-	384
Max Target Length	128	512	768	128
Learning Rate	0.0002	0.0002	0.0002	0.0002
LR Scheduler	Constant	Constant	Constant	Constant
Weight Decay	0.001	0.001	0.001	0.001

Table B.1.: Training hyperparameters for applying PEFT on Quantized LLMs on different datasets and across model sizes.

B.2. Detailed Results

	LLaMa2			LLaMa3		
LoRA Config	Unnatural	OASST	HH-RLHF	Unnatural	OASST	HH-RLHF
$r = 64$, $\alpha = 16$	2.3701	15.7632	3.4377	2.5026	22.7634	4.0311
$r = 64$, $\alpha = 16$ + RsLoRA	2.0637	12.7108	2.4762	2.2161	11.6514	-
$r = 64$, $\alpha = 16$ + LoftQ	1.9078	8.0551	3.4419	-	-	-
$r = 64$, $\alpha = 16$ + LoftQ + RsLoRA	2.0989	12.9316	2.5861	-	-	-
$r = 8$, $\alpha = 16$	2.3158	12.0986	3.6142	2.4358	7.0868	4.4494
$r = 8$, $\alpha = 16$ + RsLoRA	1.9611	13.3002	3.3981	1.9712	10.3298	-
$r = 8$, $\alpha = 16$ + LoftQ	2.2766	7.2956	3.6309	-	-	-
$r = 8$, $\alpha = 16$ + LoftQ + RsLoRA	2.1042	12.9427	3.4141	-	-	-
$r = 16$, $\alpha = 32$	2.0806	14.8292	3.2428	2.4503	11.8029	3.9345
$r = 16$, $\alpha = 32$ + RsLoRA	1.9545	10.8588	3.0265	2.1954	12.3694	-
$r = 16$, $\alpha = 32$ + LoftQ + RsLoRA	2.2322	12.9507	3.1276	-	-	-
$r = 64$, $\alpha = 128$	2.0137	11.7592	2.5184	2.2635	12.7997	3.1769
$r = 64$, $\alpha = 128$ + RsLoRA	2.7778	9.5901	3.3966	4.3249	16.7684	-
$r = 64$, $\alpha = 128$ + LoftQ	1.665	13.4349	2.5697	-	-	-
$r = 256$, $\alpha = 128$	1.9317	11.3912	2.4207	2.2128	14.1924	2.973
$r = 256$, $\alpha = 128$ + RsLoRA	2.8096	3256.7863	149.7744	172.2573	857.3021	-
$r = 256$, $\alpha = 128$ + LoftQ	1.9234	12.5148	2.7525	-	-	-
$r = 256$, $\alpha = 32$ + RsLoRA	2.1005	9.1015	3.0795	2.3927	11.3352	-
$r = 256$, $\alpha = 32$ + LoftQ + RsLoRA	2.089	9.7625	2.4801	-	-	-

Table B.2.: Evaluation Perplexity across different configurations of Q-PEFT methods.

	LLaMa2			LLaMa3		
QLoRA Config	Unnatural	OASST	HH-RLHF	Unnatural	OASST	HH-RLHF
$r = 64$, $\alpha = 16$	49.29	43.39	39.16	60.78	60.42	56.74
$r = 64$, $\alpha = 16$ + RsLoRA	43.39	44.29	35.39	55.52	58.44	
$r = 64$, $\alpha = 16$ + LoftQ	49.14	44.39	40.01	-	-	-
$r = 64$, $\alpha = 16$ + LoftQ + RsLoRA	42.31	45.11	37.2	-	-	-
$r = 8$, $\alpha = 16$	48.96	42.46	42.05	59.66	57.2	56.36
$r = 8$, $\alpha = 16$ + RsLoRA	49.14	42.01	39.37	56.78	57.48	
$r = 8$, $\alpha = 16$ + LoftQ	46.79	43.54	40	-	-	-
$r = 8$, $\alpha = 16$ + LoftQ + RsLoRA	47.16	43.59	36.05	-	-	-
$r = 16$, $\alpha = 32$	46.14	44.65	38.17	58.27	59.67	53.67
$r = 16$, $\alpha = 32$ + RsLoRA	41.56	42.63	32.02	51.59	57.4	
$r = 16$, $\alpha = 32$ + LoftQ + RsLoRA	42.73	40.77	32.11	-	-	-
$r = 64$, $\alpha = 128$	43.73	46.3	36.07	52.11	59.16	50.54
$r = 64$, $\alpha = 128$ + RsLoRA	23.81	29.81	29.24	25.39	32.76	
$r = 64$, $\alpha = 128$ + LoftQ	47.28	43.43	38.75	-	-	-
$r = 256$, $\alpha = 128$	45.39	43.86	36.4	54.84	59.47	52.58
$r = 256$, $\alpha = 128$ + RsLoRA	27.28	25.87	25.51	24.01	25.72	
$r = 256$, $\alpha = 128$ + LoftQ	47.05	44.09	41.89	-	-	-
$r = 256$, $\alpha = 32$ + RsLoRA	36.87	42.34	31.99	25.81	48.22	
$r = 256$, $\alpha = 32$ + LoftQ + RsLoRA	34.71	40.17	32.1	-	-	-

Table B.3.: 5-shot MMLU Test Accuracy for LLaMa Models fine-tuned with different LoRA methods on different datasets. The quantization method used here is NF4 + DQ

The detailed perplexity and MMLU results for QLoRA and variants in table B.2 and B.3 offer valuable insights into the most effective LoRA hyperparameter combinations for different datasets and variations. By identifying underperforming configurations, these results allow for the exclusion of less optimal setups from future analyses.

	LLaMa 2		
DoRA Config	Unnatural	OASST	HH-RLHF
$r = 64$, $\alpha = 16$	2.5031	7.9429	3.6838
$r = 64$, $\alpha = 16$ + RsLoRA	2.1689	14.4945	2.6131
$r = 64$, $\alpha = 16$ + LoftQ	1.6519	8.3619	3.6622
$r = 8$, $\alpha = 16$	2.0004	6.8301	3.7683
$r = 8$, $\alpha = 16$ + RsLoRA	1.6834	9.2414	3.7827
$r = 8$, $\alpha = 16$ + LoftQ	1.6492	6.8758	3.7682
$r = 16$, $\alpha = 32$ + RsLoRA	1.6675	10.1681	3.6026
$r = 64$, $\alpha = 128$	1.6887	12.9143	3.4741
$r = 64$, $\alpha = 128$ + LoftQ	1.7405	12.0702	3.1668
$r = 256$, $\alpha = 128$	1.754	8.8216	2.8053
$r = 256$, $\alpha = 128$ + LoftQ	1.7303	14.7391	3.064
$r = 256$, $\alpha = 32$ + RsLoRA	1.9479	8.5892	3.0962

Table B.4.: The detailed Evaluation Perplexity for different configurations for DoRA.

	LLaMa 2 7B		
DoRA Config	Unnatural	OASST	HH-RLHF
$r = 64$, $\alpha = 16$	48.63	42.56	41.12
$r = 64$, $\alpha = 16$ + RsLoRA	45.15	44.35	37.93
$r = 64$, $\alpha = 16$ + LoftQ	49.21	43.92	40.79
$r = 8$, $\alpha = 16$	48.61	42.27	40.01
$r = 8$, $\alpha = 16$ + RsLoRA	48.92	41.83	41.98
$r = 8$, $\alpha = 16$ + LoftQ	48.72	43.52	41.46
$r = 16$, $\alpha = 32$ + RsLoRA	46.04	41.05	33.47
$r = 64$, $\alpha = 128$	46.27	42.08	36.94
$r = 64$, $\alpha = 128$ + LoftQ	44.23	45.99	39.2
$r = 256$, $\alpha = 128$	45.65	43.85	38.32
$r = 256$, $\alpha = 128$ + LoftQ	46.79	45.12	38.5
$r = 256$, $\alpha = 32$ + RsLoRA	27.68	40.89	31.26

Table B.5.: 5-shot MMLU Test Accuracy for LLaMa 2 7B Model fine-tuned with different DoRA combinations on different datasets. The quantization method used here is NF4 + DQ

The detailed perplexity and MMLU results for DoRA in table B.4 and B.5 offer valu-

able insights into the most effective combinations for different datasets and variations.

HQQ Bits	LoRA Config	Unnatural	OASST
4-bit	$r = 64$, $\alpha = 16$	1.6352	8.7849
2-bit	$r = 64$, $\alpha = 16$	1.8301	8.3133
4-bit	$r = 8$, $\alpha = 16$	1.6473	6.7715
2-bit	$r = 8$, $\alpha = 16$	1.8289	10.0195
4-bit	$r = 64$, $\alpha = 128$	1.7369	12.7801
2-bit	$r = 64$, $\alpha = 128$	1.9199	20.8165
4-bit	$r = 256$, $\alpha = 128$	1.7496	12.6529
2-bit	$r = 256$, $\alpha = 128$	1.9691	22.5033

Table B.6.: The detailed Evaluation Perplexity for different configurations of HQQ-LoRA.

HQQ Bits	LoRA Config	Unnatural	OASST
4-bit	$r = 64$, $\alpha = 16$	46.28	43.18
2-bit	$r = 64$, $\alpha = 16$	32.48	24.95
4-bit	$r = 8$, $\alpha = 16$	46.36	43.58
2-bit	$r = 8$, $\alpha = 16$	33.29	25.03
4-bit	$r = 64$, $\alpha = 128$	43.15	44.33
2-bit	$r = 64$, $\alpha = 128$	31.92	23.62
4-bit	$r = 256$, $\alpha = 128$	43.55	44.39
2-bit	$r = 256$, $\alpha = 128$	28.03	26.79

Table B.7.: 5-shot MMLU Test Accuracy for LLaMa 2 7B Model fine-tuned with different HQQ-LoRA combinations on different datasets.

The detailed perplexity and MMLU results for HQQ with LoRA in table B.6 and B.7 offer valuable insights into the effect of quantizing into lower bits for different datasets and hyperparameters.

Ehrenwörtliche Erklärung

Ich versichere, dass ich die beiliegende Bachelor-, Master-, Seminar-, oder Projektarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und in der untenstehenden Tabelle angegebenen Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Declaration of Used AI Tools

Tool	Purpose	Where?	Useful?
ChatGPT	Rephrasing	Throughout	+
DeepL	Translation	Throughout	+
Quillbot	Paraphrasing	Throughout	+

Unterschrift
Mannheim, den 03. 09. 2024