UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

School of Informatics

Comparison of different techniques to parameter-efficient fine-tuning in deep learning

End of Degree Project

Bachelor's Degree in Informatics Engineering

AUTHOR: Moreno Arcas, Alejandro

Tutor: Juan Císcar, Alfonso

Cotutor: Civera Saiz, Jorge

Experimental director: Iranzo Sánchez, Jorge

ACADEMIC YEAR: 2023/2024

Resum

El camp de l'aprenentatge profund destaca com una de les àrees més dinàmiques i prometedores dins de la intel·ligència artificial. La pràctica més comuna per a construir sistemes d'aprenentatge profund implica l'ús de grans models prèviament entrenats, que després s'adapten a dominis o tasques particulars. Esta adaptació es realitza ajustant els paràmetres del model amb la finalitat d'optimitzar alguna mesura de rendiment adequada. Un exemple notable és la tècnica d'adaptació coneguda com *Low Rank Adaptation* (LoRA), que està demostrant oferir resultats prometedors en tasques relacionades amb la visió per computadora i el processament del llenguatge natural. En el marc d'este treball, ens proposem analitzar i contrastar diverses tècniques per a l'ajust eficient de paràmetres en el context de l'aprenentatge profund. Per això, es durà a terme una exploració exhaustiva de les diferents variants de LoRA aplicades a diverses tasques i models preentrenats. L'objectiu fonamental radica a identificar les tècniques més efectives per a millorar el rendiment després d'adaptar aquestos models.

Paraules clau: Intel·ligència artifical, aprenentatge profund, model preentrenat, adaptació de models, ajust eficient de paràmetres, LoRA

Resumen

El campo del aprendizaje profundo destaca como una de las áreas más dinámicas y prometedoras dentro de la inteligencia artificial. La práctica más común para construir sistemas de aprendizaje profundo implica el uso de grandes modelos previamente entrenados, que luego se adaptan a dominios o tareas particulares. Esta adaptación se realiza ajustando los parámetros del modelo con el fin de optimizar alguna medida de rendimiento adecuada. Un ejemplo notable es la técnica de adaptación conocida como *Low Rank Adaptation* (LoRA), que está demostrando ofrecer resultados prometedores en tareas relacionadas con la visión por computadora y el procesamiento del lenguaje natural. En el marco de este trabajo, nos proponemos analizar y contrastar diversas técnicas para el ajuste eficiente de parámetros en el contexto del aprendizaje profundo. Para ello se llevará a cabo una exploración exhaustiva de las diferentes variantes de LoRA aplicadas a diversas tareas y modelos preentrenados. El objetivo fundamental radica en identificar las técnicas más efectivas para mejorar el rendimiento tras adaptar estos modelos.

Palabras clave: Inteligencia artifical, aprendizaje profundo, modelo preentrenado, adaptación de modelos, ajuste eficiente de parámetros, LoRA

Abstract

The field of deep learning stands out as one of the most dynamic and promising areas within artificial intelligence. The most common practice for building deep learning systems involves the use of large pre-trained models, which are then adapted to particular domains or tasks. This adaptation is done by adjusting the model parameters with the purpose of optimizing some appropriate performance measure. A notable example is the adaptation technique known as low-rank adaptation (LoRA), which is proving to offer promising results in tasks related to computer vision and natural language processing. Within the framework of this work, we propose to analyze and contrast various techniques for efficient parameter fine-tuning in the context of deep learning. To this end, a comprehensive exploration of the different variants of LoRA applied to a variety of tasks and pre-trained models will be carried out. The fundamental objective lies in identifying the most effective techniques to improve performance after adapting these models.

Key words: Artificial intelligence, deep learning, pre-trained model, model adaptation, efficient parameter tuning, LoRA

Contents

Contents vii
List of Figures ix
List of Tables x

1 Introduction 1
1.1 Motivation . . . 1
1.2 Objectives . . . 1
1.3 Document structure . . . 2
2 Background 3
2.1 Machine learning . . . 3
2.2 Neural networks . . . 4
2.3 Transformers . . . 5
2.4 Large language models . . . 6
2.4.1 Phi 1.5 . . . 7
2.5 Transfer learning . . . 8
2.6 Parameter-efficient fine-tuning techniques . . . 8
3 Datasets and tasks 9
3.1 Introduction . . . 9
3.2 PromptSource . . . 9
3.3 BoolQ . . . 10
3.4 PIQA . . . 10
3.5 SIQA . . . 11
3.6 WinoGrande . . . 12
3.7 ARC . . . 13
3.8 OBQA . . . 14
4 LoRA, DoRA and LoKr 17
4.1 Introduction . . . 17
4.2 Orthonormal matrices . . . 17
4.3 LoRA . . . 18
4.4 LoKr . . . 19
4.5 DoRA . . . 19
4.6 Experiments . . . 20
4.7 Conclusions . . . 21
5 OFT: Orthogonal fine-tuning 23
5.1 Introduction . . . 23
5.2 OFT . . . 23
5.3 BOFT . . . 24
5.4 Experiments . . . 26
5.5 Conclusions . . . 27
6 SHOFT: SVD Householder orthonormal fine-tuning 29
6.1 Introduction . . . 29
6.2 Singular value decomposition . . . 29

6.3 Householder transformation . . . 30
6.3.1 CWY transform . . . 30
6.4 Insights and method . . . 31
6.5 Initialization . . . 32
6.6 Experiments . . . 33
6.6.1 Experiments on WinoGrande . . . 33
6.6.2 Experiments on BoolQ . . . 34
6.6.3 Experiments on PIQA . . . 34
6.6.4 Experiments on SIQA . . . 35
6.6.5 Experiments on ARC-e . . . 35
6.6.6 Experiments on ARC-c . . . 36
6.6.7 Experiments on OBQA . . . 36
6.6.8 Average results . . . 37
6.7 Conclusions . . . 37

7 RHOFT: Randomized SHOFT **39**
7.1 Introduction . . . 39
7.2 Randomized singular value decomposition . . . 39
7.3 SVD curve analysis . . . 40
7.4 Insights and methods . . . 41
7.5 Experiments . . . 43
7.5.1 Experiments on WinoGrande . . . 43
7.5.2 Experiments on BoolQ . . . 44
7.5.3 Experiments on PIQA . . . 44
7.5.4 Experiments on SIQA . . . 45
7.5.5 Experiments on ARC-e . . . 45
7.5.6 Experiments on ARC-c . . . 46
7.5.7 Experiments on OBQA . . . 46
7.5.8 Average results . . . 47
7.5.9 Effect of k . . . 47
7.5.10 Speed comparison . . . 48
7.6 Conclusions . . . 48

8 Conclusions **49**
8.1 Summary of work done . . . 49
8.2 Objectives achieved . . . 49
8.3 Future work . . . 50

Bibliography **51**

Appendix

A Hyperparameters **53**

List of Figures

2.1 Representation of the perceptron as a neuron. . . . 4
2.2 Representation of a multilayer perceptron 4
2.3 Representation of the Transformer architecture. . . . 5

4.1 LoRA reparametrization. . . . 18
4.2 LoRA geometric interpretation. . . . 18
4.3 DoRA reparametrization. . . . 19
4.4 Comparative average for LoRA and DoRA 20

5.1 OFT reparametrization. . . . 24
5.2 Butterfly structure for $d = 4$ 25
5.3 Comparative average for LoRA, DoRA and OFT 26

6.1 Householder transformation on $\mathbb{R}^3$ 30
6.2 SHOFT reparametrization. . . . 31
6.3 Comparative average on WinoGrande 33
6.4 Comparative average on BoolQ 34
6.5 Comparative average on PIQA 34
6.6 Comparative average on SIQA 35
6.7 Comparative average on ARC-e 35
6.8 Comparative average on ARC-c 36
6.9 Comparative average on OBQA 36
6.10 Comparative average for DoRA and SHOFT 37

7.1 SVD curve of Query matrix 40
7.2 SVD curve of Key matrix 40
7.3 SVD curve of Value matrix 41
7.4 RHOFT reparametrization. . . . 42
7.5 Hypersphere on $\mathbb{R}^3$ 42
7.6 Hypersphere on $\mathbb{R}^2$ 42
7.7 Comparative average on WinoGrande 43
7.8 Comparative average on BoolQ 44
7.9 Comparative average on PIQA 44
7.10 Comparative average on SIQA 45
7.11 Comparative average on ARC-e 45
7.12 Comparative average on ARC-c 46
7.13 Comparative average on OBQA 46
7.14 Comparative average for DoRA, SHOFT and RHOFT 47
7.15 Average accuracy on all datasets varying k 47

List of Tables

3.1 Baseline accuracy of Phi's 1.5 models on BoolQ. 10
3.2 Baseline accuracy of Phi's 1.5 models on PIQA. 11
3.3 Baseline accuracy of Phi's 1.5 models on SIQA. 12
3.4 Baseline accuracy of Phi's 1.5 models on WinoGrande. 13
3.5 Baseline accuracy of Phi's 1.5 models on ARC-Easy. 13
3.6 Baseline accuracy of Phi's 1.5 models on ARC-Challenge. 14
3.7 Baseline accuracy of Phi's 1.5 models on OBQA. 15

7.1 Average time on the commonsense reasoning tasks 48
7.2 Time performance on the commonsense reasoning tasks 48

A.1 Hyperparameter configurations for Phi-1.5 on the commonsense reasoning tasks. 53

List of Acronyms

ARC AI2 Reasoning Challenge.

BOFT Orthogonal fine-tuning via butterfly factorization.

DoRA Weight-Decomposed Low-Rank Adaptation.

LLM Large language model.

LoKr Low Rank Adaptation with Kronecker product.

LoRA Low Rank Adaptation.

MLLP Machine Learning and Language Processing.

NLP Natural Language Processing.

OBQA OpenBookQA.

OFT Orthogonal fine-tuning.

PEFT Parameter-efficien fine-tuning.

RHOFT Randomized SVD Householder Orthonormal fine-tuning.

SHOFT SVD Householder Orthonormal fine-tuning.

SVD Singular value decomposition.

UT Upper Triangular.

VRAIN Valencian Research Institute for Artificial Intelligence.

CHAPTER 1
Introduction

1.1 Motivation

Parameter-efficient fine-tuning (PEFT) techniques aim to reduce the amount of parameters demanded in order to fine-tune machine learning models without loosing performance. These techniques are highly needed nowadays with models with billions of parameters which require computational power in order to perform full fine-tuning.

PEFT enable scalable solutions for large models, facilitate multi-task learning, and support real-time applications. They also offer economic benefits by lowering costs and increasing accessibility to advanced AI technology. Additionally, efficient fine-tuning can lead to improved generalization, robustness, and interpretability of models.

This project is developed under the framework of a collaboration scholarship with the *Machine Learning and Language Processing* (MLLP) research group[1] of the *Valencia Research Institute on Artificial Intelligence*[2] (VRAIN). In addition, this project is also enriched with the background gained from being currently taking a degree in Mathematics in parallel with this degree.

1.2 Objectives

The main objectives of this work are the following:

1. To gather and apply all the concepts studied during this degree alongside with state-of-the-art PEFT techniques.
2. To interpret the underlying mathematical properties of state-of-the-art PEFT techniques.
3. To design a PEFT technique from a theoretical perspective.
4. To evaluate the performance of the designed technique alongside with other PEFT techniques.

[1] https://mllp.upv.es
[2] https://vrain.upv.es

1.3 Document structure

This document is divided into eight chapters. This first chapter introduces the motivation of this project, describes the objectives and explains the structure of this work. Next, Chapter 2 contains an introduction to machine learning, starting from the general landscape to the state-of-the-art model structures and techniques. Then, Chapter 3 enumerates and describes the datasets and tasks that are used for training and evaluating the PEFT techniques.

The following four chapters are devoted to describe and compare different PEFT techniques. Chapter 4 presents LoRA with some of its variants, and performs experiments to compare them. Then, Chapter 5 mainly describes orthogonal fine-tuning (OFT) and evaluates it with other PEFT techniques.

After that, Chapters 6 and 7 show the designed PEFT techniques. Chapter 6 explains and evaluates the designed method inspired by those commented in previous chapters. Chapter 7 improves the proposed method by deriving a generalization. Finally, Chapter 8 summarizes the work done, gives some concluding remarks, and outlines possible futures lines of work and investigation.

The reader is recommended to read the document sequentially after Chapter 3, since each chapter introduces and explains concepts that will be used and mentioned in the following ones.

CHAPTER 2
Background

2.1 Machine learning

Machine learning is a field that focuses on creating computer algorithms that can learn to solve tasks on their own by recognizing patterns in data. Firstly, it is compulsory to define what we refer when we say that a machine program is 'learning' [8]:

Definition 1 *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

The first decision we need to make is to pick the kind of training our system will use to learn. The type of training experience we choose can greatly affect how well the system learns. An important factor is whether the training gives direct feedback or indirect feedback on the choices the system makes.

In this context, the general way of 'learning' is supervised learning. Supervised learning is a fundamental approach in the field of machine learning where a model is trained using labeled data. In this context, 'labeled data' means that each training example is paired with the expected output label. The goal of the model is to learn a mapping from inputs to outputs that can be used to make predictions on new, unseen data.

2.2 Neural networks

The term 'neural network' has its origins in attempts to find mathematical representations of information processing in biological systems [9]. The main idea behind neural networks is the 'neuron' itself, which is often called as perceptron. The perceptron is a non-linear function $f : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R} \to \{0, 1\}$ defined as follows

$$f(\boldsymbol{x}; \boldsymbol{w}, b) = \begin{cases} 1 & \text{if } \boldsymbol{w}^t \boldsymbol{x} + b > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

where $\boldsymbol{w}, b$ are parameters and $\boldsymbol{x}$ is the input of the perceptron.

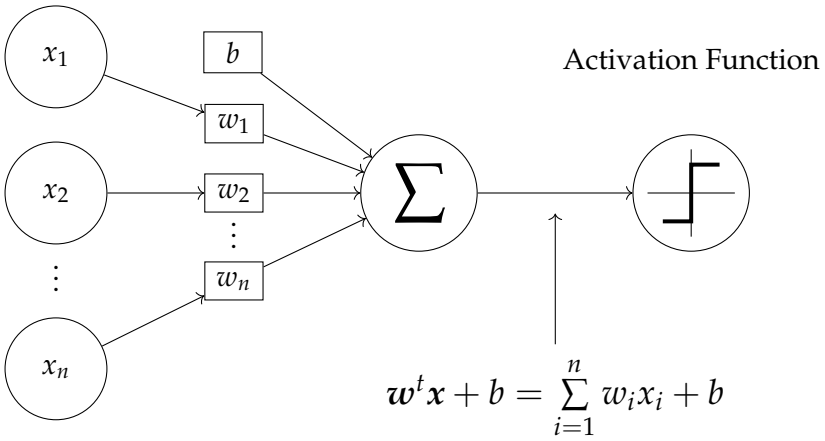

Figure 2.1: Representation of the perceptron as a neuron.

Figure 2.2 shows the representation of the perceptron for classifying a n dimensional vector $\boldsymbol{x}$. However, the most successful model of this kind is the feed-forward neural network, which is also called the multilayer perceptron.

This model gained popularity thanks to the increment of the computational power available. The neurons are grouped into layers, and each neuron process the entry to produce an output, which is passed to the next layer. Input layers takes the input vector, while hidden layers are used to process the input vector and the output layer returns the processed result.

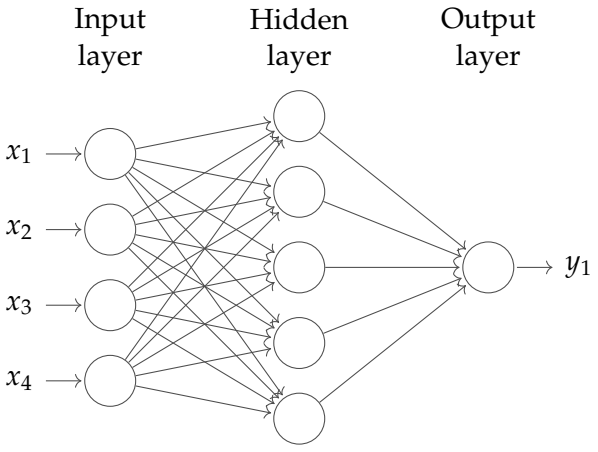

Figure 2.2: Representation of a multilayer perceptron

As mentioned, the neural network model comprises two stages of processing, each of which resembles the perceptron model, and for this reason the neural network is also known as the multilayer perceptron [5], represented in Figure 2.2. A key difference compared to the perceptron, however, is that the neural network employs differentiable nonlinearities in the hidden units, whereas the perceptron uses step-function nonlinearities. This means that the neural network function is differentiable with respect to the network parameters, and this property plays a central role in network training.

Neural networks use different architectures depending on the type of data. Multilayer perceptrons are good for tabular data because they have fully connected layers that capture patterns in structured datasets. Convolutional neural networks are designed for images, recognizing spatial features. Recurrent neural networks, work well with sequential data by maintaining temporal dependencies. As data becomes more complex, Transformers have become popular due to their self-attention mechanisms that learn dependencies on a sequence.

2.3 Transformers

The Transformer is a neural network architecture [3] that relies on the so called self-attention mechanism to compute dependencies in a sequence. Self-attention assigns weights to different parts of the sequence based on how they relate to each other. This helps the model understand which elements in the sequence are important.

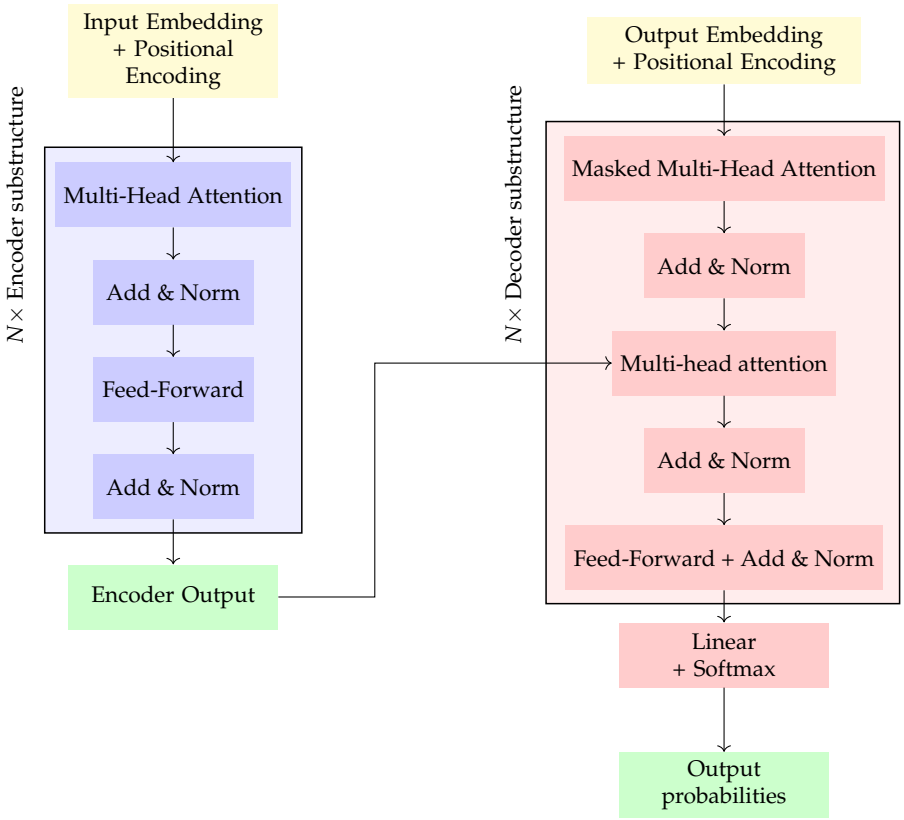

Figure 2.3: Representation of the Transformer architecture.

Transformers have two important substructures that can be observed in Figure 2.3. The first substructure is the encoder architecture. It consists of a stack of identical layers, each comprising two main sub-layers: a multi-head self-attention mechanism and a fully connected feed-forward network. Inputs are first converted into embeddings, which are then augmented with positional encodings to retain sequence information. In each encoder layer, the self-attention mechanism allows the model to focus on different parts of the input sequence, capturing relationships between words. The equation that models the self-attention mechanism A is the following:

$$A(Q, K, V) = \mathcal{S}\left(\frac{QK^t}{\sqrt{d_k}}\right)V \tag{2.2}$$

where Q, K, V are named as Query, Key and Value matrices, d_k is the dimension of the matrices, and $\mathcal{S}$ is the softmax function.

The multi-head approach enables the model to attend to information from various representation subspaces [3]. Following the attention mechanism, a feed-forward network processes each position independently. Each of these sub-layers is followed by residual connections and layer normalization to stabilize training and facilitate gradient flow, ensuring that the model can learn effectively even with deep architectures.

The second substructure is the decoder architecture in the Transformer model, which mirrors the encoder but includes additional mechanisms to handle the auto-regressive nature of sequence generation. Each decoder layer consists of three main sub-layers: a masked multi-head self-attention mechanism, a multi-head attention mechanism over the encoder's output, and a position-wise fully connected feed-forward network. The masked self-attention sub-layer ensures that predictions for a given position depend only on known outputs for preceding positions, preserving the sequence order. The second attention sub-layer performs multi-head attention over the encoder's output, allowing the decoder to focus on relevant parts of the input sequence. This is followed by a feed-forward network for further processing. Each sub-layer in the decoder also has residual connections and layer normalization to maintain training stability and efficiency. The final output is passed through a linear transformation and a softmax layer to generate the probability distribution over the target vocabulary.

2.4 Large language models

Large language models (LLMs) [10, 11] represent an advancement in computational linguistics, born by the demands of human-machine natural language interaction. These models, composed by neural networks and trained on vast datasets, serve as versatile tools capable of executing complex language tasks such as translation, summarization, and conversational interactions.

Their emergence marks a transformative milestone in artificial intelligence, fueled by advancements in transformer architectures, increased computational prowess, and the availability of extensive training data. LLMs stand as large-scale, pre-trained statistical models that approximate human-level proficiency across diverse linguistic tasks. This achievement represents the culmination of decades of research across four distinct waves: statistical models, neural models, pre-trained models, and finally, the powerful LLMs of today. LLMs are mainly divided into three categories regarding its structure:

In encoder-only LLMs, each attention layer can access all words in the original sentence at every stage. The pre-training process typically involves corrupting a given sentence, such as by masking random words, and then tasking the model with reconstructing the original sentence. Encoder models excel at tasks that require understanding the entire sequence, such as sentence classification, named entity recognition, and extractive question answering. One prominent example of an encoder-only model is BERT (Bidirectional Encoder Representations from Transformers) [12].

In decoder-only models, each attention layer at each stage can only access words that precede the current word in the sentence. These models are also known as auto-regressive models. Pre-training for decoder-only models usually involves predicting the next word or token in a sequence. They are particularly effective for tasks involving text generation. GPT (Generative Pre-trained Transformer) [13, 14] models are notable examples of this category.

In encoder-decoder models, they utilize both an encoder and a decoder. At each stage, the encoder's attention layers can access all words in the original sentence, whereas the decoder's attention layers can only access words that precede a given word in the input. These models are typically pre-trained using objectives similar to those of encoder or decoder models, but with more complex formulations. For instance, some models are pre-trained by replacing random spans of text (which can contain several words) with a single masked special word, and the objective is then to predict the original text that this mask word replaces. Encoder-decoder models are well-suited for tasks that involve generating new sentences based on a given input, such as summarization, translation, or generative question answering. One good example would be BART [15].

2.4.1. Phi 1.5

Phi 1.5 [16] is a decoder-only LLM with 1.3 billion parameters, trained on a dataset of 30 billion tokens. Its architecture is based on a Transformer with 24 layers, 32 attention heads per layer, and each head having a dimension of 64. The model utilizes rotary embeddings and a context length of 2048 tokens, with training efficiency enhanced by flash-attention techniques. The tokenizer used is from the codegen-mono model [18].

The training data for Phi 1.5 includes Phi 1's [17] dataset (7 billion tokens) and an additional 20 billion tokens of synthetically generated "textbook-like" data, designed to teach common sense reasoning and general knowledge. The data generation process was carefully designed, using 20,000 selected topics and web data samples to ensure diversity. The non-synthetic portion of the training data consists of 6 billion tokens from a filtered code dataset. From the training dataset, three models were trained: phi-1.5-web-only, phi-1.5-web and phi-1.5. The first one was trained only using internet information, while the last one was using the training data commented before. The model in the middle was a mix of both types. For our purpose, we will employ phi-1.5 for performing the experiments.

Phi 1.5 achieves performance on common sense reasoning benchmarks comparable to models ten times its size, demonstrating significant efficiency. The use of synthetic data helps address issues of toxic and biased content generation. Additionally, a variant called Phi 1.5-web incorporates filtered web data to enhance performance further. The open-sourcing of Phi 1.5 aims to facilitate research on critical issues such as in-context learning, mechanistic interpretability, and mitigation strategies for hallucinations, toxic content, and biased outputs [16].

The model's capabilities suggest that high-level performance can be achieved in smaller LLMs, promoting more efficient and environmentally sustainable AI systems. Future plans for Phi 1.5 include expanding its synthetic dataset to cover more topics and fine-tuning the model for specific tasks. This model indicates that achieving capabilities comparable to larger models at a one-billion-parameter scale is a realistic goal [16].

2.5 Transfer learning

To reach high-level performance efficiently, using existing model knowledge becomes crucial, and this is where transfer learning becomes crucial. Transfer learning [7] aims to solve a new problem by leveraging the similarity of data (task or models) between the old problem and the new one to perform knowledge transfer. The concept of transfer learning was originally born in psychology and pedagogy (Bray, 1928). It is also called "Learning Transfer" by psychologists, indicating the influence of one learning process to another. This even happens naturally in our daily lives. For instance, if we can play badminton, then we can learn to play tennis since playing badminton and tennis share some similar strategy and tricks

In recent years, there has been a growing interest in pre-training large language models. Researchers have found that by collecting large datasets, whether labeled or not, we can pre-train these models using self-supervised learning methods. These pre-trained models can then be fine-tuned for specific tasks in natural language processing (NLP) to boost their performance.

2.6 Parameter-efficient fine-tuning techniques

With the continuous growth in the number of parameters of transformer-based pretrained language models, particularly the emergence of large language models (LLMs) with billions of parameters, many NLP tasks have demonstrated remarkable success.

However, the very large size and high computational demands of these models create challenges when adapting them to specific tasks, especially in environments with limited computing power. Parameter-efficient fine-tuning (PEFT) [1] offers a solution by reducing the number of trainable parameters and memory usage while achieving similar performance to full fine-tuning. This has led to increased development of PEFT techniques, particularly for fine-tuning LLMs.

CHAPTER 3
Datasets and tasks

3.1 Introduction

This chapter is devoted to present the datasets required to train and evaluate the different PEFT methods. Section 3.2 presents a library for managing prompt templates from the different datasets. The following sections describe all the datasets employed with its structure, and give a prompt example. In addition, it also remarks the benchmarks obtained from the Phi-1.5 model described in Subsection 2.4.1.

3.2 PromptSource

PromptSource [19] is a system designed for creating, sharing, and using natural language prompts. A prompt is an input or instruction given to a language model to guide its response. It sets the context or specifies the task for the model to generate relevant output. Prompts function by mapping an example from a dataset to a natural language input and target output. Employing prompts to train and query language models is a burgeoning area in NLP, requiring new tools that enable users to develop and refine these prompts collaboratively. PromptSource addresses the emergent challenges in this new setting with

- A templating language for defining data-linked prompts.
- An interface that lets users quickly iterate on prompt development by observing outputs of their prompts on many examples.
- A community driven set of guidelines for contributing new prompts to a common pool. Over 2,000 prompts for roughly 170 datasets are already available in PromptSource.

The process of prompt engineering is essential for successful deployment, as choices in prompting can significantly impact downstream predictions, especially in zero-shot settings. Consequently, there is an increasing demand for tools that facilitate the creation of prompt corpora.

For our purpose, PromptSource will be employed for generating the corresponding prompts and facilitate experimental reproducibility. Due to the prompt lack in all works [26, 30, 16], we will select the ones which gives better results.

3.3 BoolQ

Understanding which facts can be inferred as true or false from a text is a crucial aspect of natural language understanding. Often, these inferences extend beyond the explicit statements in the text. *BoolQ* [20] is a dataset consisting of naturally occurring yes/no questions. These questions are particularly challenging and demand a broad spectrum of inference skills to answer correctly.

The dataset contains 15,942 examples, 9,427 for training a and 3,270 for testing. It consists of 3 columns for generating the prompts:

- *passage*: the passage that needs to be understood.
- *question*: the question which requires passage comprehension to be answered correctly.
- *answer*: the correct solution.

For our training and evaluating purposes, the 'yes_no_question' template from PromptSource was employed. An example of prompt would be the following:

> Text: *Announced in April 2000 at the New York Auto Show and arriving in late 2000 in Japan and January 2001 in North America, the Highlander became one of the first car-based mid-size SUV or mid-size crossovers. The Highlander is the crossover counterpart to the more rugged, truck-based midsize 4Runner and became Toyota's best-selling SUV before being surpassed by the smaller RAV4 in 2006. In Japan, the Kluger is exclusive to dealership network called Toyota NETZ as a larger alternative to the RAV4.*
>
> Answer the following yes/no question: *is the toyota highlander on a truck frame*? Yes or no?

From the technical report of Phi 1.5 [16], the baseline accuracy results presented using zero-shot evaluation on this dataset are in Table 3.1.

Table 3.1: Baseline accuracy of Phi's 1.5 models on BoolQ.

phi-1.5-web-only (1.3B)	phi-1.5-web (1.3B)	phi-1.5 (1.3B)
63.2	72.8	75.8

3.4 PIQA

PIQA [21] is a dataset designed to benchmark advancements in physical commonsense understanding. The core task involves multiple-choice question answering: given a question q and two possible solutions s_1 and s_2, a model must select the most appropriate solution, with exactly one being correct.

PIQA's aim is to offer insights and a benchmark for progress towards language representations that encompass knowledge typically acquired through seeing or experiencing, thereby facilitating the development of language models beneficial beyond the natural language processing community.

The dataset contains 16,000 examples for training, 2,000 for development and 3,000 for testing. It consists of 4 columns for generating the prompts:

- *goal*: the question which requires physical commonsense to be answered correctly.
- *sol1*: the first possible solution.
- *sol2*: the second possible solution.
- *label*: the correct solution. 0 refers to sol1 and 1 refers to sol2.

For our training and evaluating purposes, the 'what_is_the_correct_ending' template from PromptSource was employed. An example of prompt would be the following:

> Goal: *how do you go to sleep*
>
> Which is the correct ending?
> *- open your eyes and sit on your bed*
> *- close your eyes and lay in bed.*
>
> Answer:

From the technical report of Phi 1.5 [16] the baseline accuracy results presented using zero-shot evaluation on this dataset are in Table 3.2.

Table 3.2: Baseline accuracy of Phi's 1.5 models on PIQA.

phi-1.5-web-only (1.3B)	phi-1.5-web (1.3B)	phi-1.5 (1.3B)
74.3	77.0	76.6

3.5 SIQA

SIQA [22] is a standardized benchmark for assessing commonsense reasoning in social contexts. It encompasses different types of inferences related to people's actions described within various situational scenarios.

The benchmark includes multiple-choice questions, each offering three possible answers. The questions and answers are collected through a three-phase crowd-sourcing process designed to gather the context, formulate the question, and generate a set of both correct and incorrect answers.

The dataset contains 35,364 examples, 33,410 for training and 3,000 for testing. It consists of 6 columns for generating the prompts:

- *context*: the social context commonsense needed to answer correctly.
- *question*: the question in relation to the context given.
- *answerA*: the first possible solution.
- *answerB*: the second possible solution.
- *answerC*: the third possible solution.

- *label*: the correct solution. 0 refers to answerA, 1 refers to answerB and 2 refers to answerC.

For our training and evaluating purposes, the 'Show choices and generate index' template from PromptSource was employed. An example of prompt would be the following:

> Context: *Ash redeemed themselves after retaking the test they failed.*
>
> Question: *How will Ash feel as a result*?
>
> Which one of these answers best answers the question according to the context?
>
> A: *relieved*
>
> B: *accomplished*
>
> C: *proud*

From the technical report of Phi 1.5 [16] the baseline accuracy results presented using zero-shot evaluation on this dataset are in Table 3.3.

Table 3.3: Baseline accuracy of Phi's 1.5 models on SIQA.

phi-1.5-web-only (1.3B)	phi-1.5-web (1.3B)	phi-1.5 (1.3B)
41.4	53.0	52.6

3.6 WinoGrande

WinoGrande [23] is a substantial dataset comprising 44,000 problems, designed to enhance both the scale and difficulty of the dataset. These are pronoun resolution problems that are straightforward for humans but challenging for machines that depend solely on statistical patterns without genuine commonsense reasoning abilities. The dataset is presented in a fill-in-the-blank format, where the blank corresponds to a mention of one of the two names in the given context.

The sub-dataset used is 'winogrande_l', which contains 10,234 examples for training and 1,767 for testing. It consists of 4 columns for generating the prompts:

- *sentence*: the sentence in which appears an underscore to be filled.
- *option1*: the first possible solution.
- *option2*: the second possible solution.
- *answer*: the correct solution.

For our training and evaluating purposes, the 'underscore refer to' template from PromptSource was employed. An example of prompt would be the following:

> *Kenneth went cheap on the gemstone present for Michael and _ was understanding about being a cheapskate.*
> What does the _ in the above sentence refer to? *Kenneth* or *Michael*?

From the technical report of Phi 1.5 [16] the baseline accuracy results presented using zero-shot evaluation on this dataset are in Table 3.4.

Table 3.4: Baseline accuracy of Phi's 1.5 models on WinoGrande.

phi-1.5-web-only (1.3B)	phi-1.5-web (1.3B)	phi-1.5 (1.3B)
60.4	74.0	73.4

3.7 ARC

The AI2 Reasoning Challenge (ARC) [24] demands significantly more advanced knowledge and reasoning capabilities compared to earlier benchmarks like SQuAD or SNLI. The ARC dataset is divided into a Challenge Set and an Easy Set. The Challenge Set consists exclusively of questions that were incorrectly answered by both a retrieval-based algorithm and a word co-occurrence algorithm. This dataset features only natural, grade-school level science questions.

One of sub-dataset used is ARC-Easy, which contains 1,119 examples for training and 1,172 for testing. It consists of 3 columns for generating the prompts:

- *question_stem*: the question to be answered.
- *choices*: a dictionary which contain pairs of *text* and *labels* to identify them.
- *answerKey*: the key of the correct solution.

For our training and evaluating purposes, the 'pick_the_most_correct_option' template from PromptSource was employed. An example of prompt would be the following:

> Pick the most correct option to answer the following question.
>
> *Which of the following has the greatest direct influence on movement of the lithosphere*?
>
> Options:
>
> - A: *the Sun*
>
> - B: *the Moon*
>
> - C: *Earth's core*
>
> - D: *Earth's mantle*

From the technical report of Phi 1.5 [16] the baseline accuracy results presented using zero-shot evaluation on this dataset are in Table 3.5.

Table 3.5: Baseline accuracy of Phi's 1.5 models on ARC-Easy.

phi-1.5-web-only (1.3B)	phi-1.5-web (1.3B)	phi-1.5 (1.3B)
66.6	76.1	75.6

The other sub-dataset used is ARC-Challenge, which contains 2,251 examples for training and 2,376 for testing. It contains the same columns as ARC-Easy. For our training and evaluating purposes, the 'qa_options' template from PromptSource was employed. An example of prompt would be the following:

Which investigation would require the longest period of observation?

Options:
- *chicken eggs hatching*
- *paint fading to a lighter color*
- *apples ripening on the tree*
- *boulders weathering into gravel*

From the technical report of Phi 1.5 [16] the baseline accuracy results presented using zero-shot evaluation on this dataset are in Table 3.6.

Table 3.6: Baseline accuracy of Phi's 1.5 models on ARC-Challenge.

phi-1.5-web-only (1.3B)	phi-1.5-web (1.3B)	phi-1.5 (1.3B)
32.9	44.9	44.4

3.8 OBQA

OpenBookQA [25] is a question-answering dataset designed to resemble open book exams, which are used to assess human understanding of a subject. The accompanying "open book" consists of 1326 elementary-level science facts. Approximately 6000 questions in the dataset challenge users to demonstrate their understanding of these facts and their ability to apply them to new situations.

While existing QA datasets over documents or knowledge bases are generally self-contained and focus on linguistic understanding, OpenBookQA delves into a more profound comprehension of both the subject matter and the language in which it is conveyed.

The main dataset contains 4,957 examples for training and 500 for testing. It consists of 3 columns for generating the prompts:

- *question_stem*: the question to be answered.
- *choices*: a dictionary which contain pairs of *text* and *labels* to identify them.
- *answerKey*: the key of the correct solution.

For our training and evaluating purposes, the 'which_correct' template from PromptSource was employed. An example of prompt would be the following:

Green parts of a life form absorb

Which is the correct answer?
- *carbon dioxide*
- *light*
- *oxygen*
- *water*

From the technical report of Phi 1.5 [16] the baseline accuracy results presented using zero-shot evaluation on this dataset are in Table 3.7.

Table 3.7: Baseline accuracy of Phi's 1.5 models on OBQA.

phi-1.5-web-only (1.3B)	phi-1.5-web (1.3B)	phi-1.5 (1.3B)
27.4	36.0	37.2

CHAPTER 4
LoRA, DoRA and LoKr

4.1 Introduction

This chapter introduces some PEFT techniques belonging to the LoRA family. As explained in Section 1.2, one of the goals of this work is to compare some fine-tuning techniques in terms of performance after fine-tuning Phi-1.5 model. Section 4.2 is dedicated to introduce orthonormal matrices, needed for understanding the next sections and chapters. Section 4.3 introduces LoRA, remarking its advantages against other fine-tuning techniques. Section 4.4 presents LoKr, a variant of LoRA for fine-tuning. Section 4.5 shows DoRA, a variant of LoRA, and discusses some of its improvements. Section 4.6 performs experiments on the tasks commented in Chapter 3 and compare the results. Finally, section 4.7 is devoted to give detailed conclusions of this chapter.

4.2 Orthonormal matrices

One important issue concerning fine-tuning is the possibility to separate direction and magnitude during training [30, 31, 32]. It is easy to see that the scaling transformation can be done correctly using a scalar matrix. However, distance-preserving direction changes can only be achieved through orthonormal matrices [4], which are matrices belonging to the special orthogonal group $SO(n)$, defined as follows:

$$SO(n) = \{U \in GL(n) \mid UU^t = U^tU = I \ \wedge \ \det(U) = 1\} \tag{4.1}$$

where $GL(n)$ is the group containing all the possible $n \times n$ invertible matrices. All matrices belonging to $SO(n)$ are also called unitary matrices. This group is also called the rotation group, since it generalizes rotations of vectors in higher dimensions.

Unlike other matrix groups, $SO(n)$ is not closed over $+$ operation. That means, for instance, given any two matrices $M, N \in SO(n)$, $M + N$ does not always belong to $SO(n)$. $SO(n)$ is only closed over the matrix multiplication operation. This group is also named as the group of distance-preserving transformations of a Euclidean space of dimension n that preserves a fixed point.

4.3 LoRA

Low Rank Adaptation (LoRA) [26, 27] on large models marked a before and after in fine-tuning techniques. This method freezes the pretrained model weights and injects trainable rank decomposition matrices into each targeted layer, greatly reducing the number of trainable parameters for downstream tasks. Considering the targeted matrix $M \in \mathbb{R}^{m\times n}$, we define the new matrix $M' \in \mathbb{R}^{m\times n}$ as follows:

$$M' = M + \Delta M = M + BA \tag{4.2}$$

where $A \in \mathbb{R}^{r\times n}, B \in \mathbb{R}^{m\times r}$ are trainable matrices and M is frozen during training. The core idea behind this method is that the change in weights during model adaptation also has a low 'intrinsic rank'.

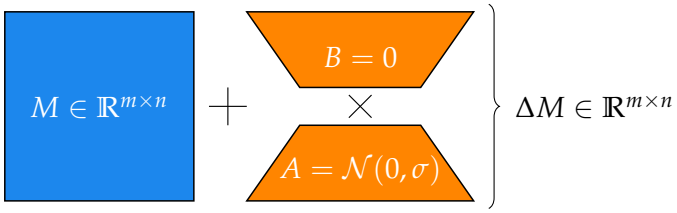

Figure 4.1: LoRA reparametrization.

For initialization, a random Gaussian distribution is used for A, while B is initialized to zero, ensuring that $\Delta M = BA = \mathbf{0}$ at the start of training. This approach is necessary to begin training with $M' = M$, as observed in Figure 4.1. Additionally, ΔM is scaled by $\frac{\alpha}{r}$, where α is a constant relative to r. This scaling reduces the need to retune hyperparameters when r varies.

One interesting perspective of LoRA's method is to see it as a computed bias vector obtained by $b(x) = \Delta Mx$ and then added to the original vector $y = Mx$.

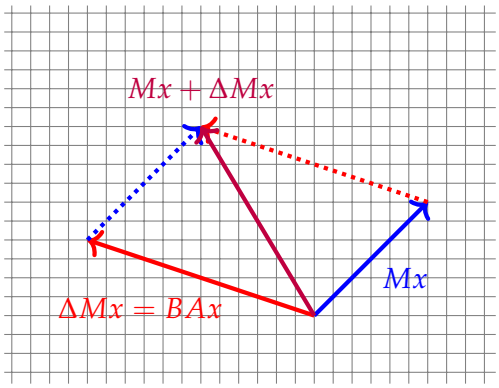

Figure 4.2: LoRA geometric interpretation.

Thus, the result is the sum of both vectors, which is represented in Figure 4.2. The precision of the computed bias improves as the intrinsic dimension (rank) increases. Yet, empirical studies indicate that simply increasing the rank doesn't consistently yield optimal outcomes [26, 30] in LLMs. These implications hint at the necessity to consider additional factors, like data volume, task complexity, etc., when determining the intrinsic rank r.

LoRA presents notable advancements in machine learning by targeting specific model parameters for modification, significantly reducing computational and storage demands. This focused adjustment not only enhances efficiency but also preserves the original model's knowledge, resulting in improved performance across tasks.

Moreover, LoRA's methodology enhances model scalability, facilitating smooth integration and customization across various applications. Its capacity to maintain high performance while minimizing resource consumption positions LoRA as an appealing solution for deploying complex models in resource-limited environments [26].

4.4 LoKr

Other PEFT methods, such as the Low-Rank Kronecker Product (LoKr) [28, 29], tries to improve LoRA by redefining some of its components. In this case, LoKr utilizes the standard Kronecker product of matrices $A \in \mathbb{R}^{s \times t}$ and $B \in \mathbb{R}^{p \times q}$:

$$B \otimes A = \begin{pmatrix} a_{11}B & \cdots & a_{1t}B \\ \vdots & \ddots & \vdots \\ a_{s1}B & \cdots & a_{st}B \end{pmatrix} \tag{4.3}$$

Where $B \otimes A \in \mathbb{R}^{sp \times tq}, sp = m, tq = n$. One unique advantage of using Kronecker products lies in the multiplicative nature of their ranks, allowing to move beyond the limitations of low-rank assumptions.

4.5 DoRA

Finally, Weight-Decomposed Low-Rank Adaptation (DoRA) [30] method is similar to the ones mentioned before: they compute a bias ΔMx and then add it to the original matrix. However, the main difference that makes DoRA better is its normalization strategy. They try to separate magnitude m and direction V, allowing both parts to be focused only on one type of transformation:

$$M' = m\frac{M + \Delta M}{\|M + \Delta M\|} = m\frac{M + BA}{\|M + BA\|} = m\frac{V}{\|V\|} \tag{4.4}$$

where $V \in \mathbb{R}^{m \times n}$ is the direction and $m \in \mathbb{R}^m$ is the magnitude. At the beginning, $m = \|M\|$ and B, A are initialized in the same way as LoRA for maintaining the original matrix at the start of the training process.

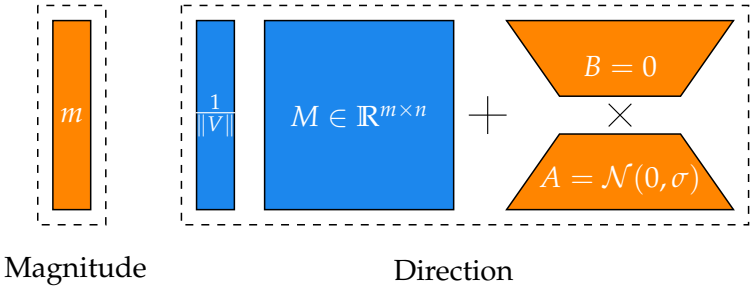

Figure 4.3: DoRA reparametrization.

Although DoRA tries to separate direction from magnitude, it does not still obtain its objective due to matrix normalization. Normalization does not always produces unitary matrices. This can be easily observed in this example:

$$M = \begin{pmatrix} 2 & 0 \\ 1 & 3 \end{pmatrix} \Longrightarrow \|M\| = \left(\frac{1}{\sqrt{5}}, \frac{1}{3}\right)^t \Longrightarrow \det\left(\frac{M}{\|M\|}\right) = \frac{2}{\sqrt{5}} \approx 0.89 \neq 1 \tag{4.5}$$

Therefore, normalizing a matrix is not an optimal technique for obtaining unitary matrices that uniquely alter the vector directions. Despite normalization only applies a slight scaling transformation, it might be interesting to see whether full separation of magnitude and scale gives better or worse results.

4.6 Experiments

In this section, comparative results among the PEFT techniques are presented. However, for the sake of simplicity, results with the LoKr technique have been discarded for discussion since it does not improve those achieved by LoRA and DoRA.

For comparing LoRA and DoRA, we will study its effects varying the intrinsic rank $r \in \{4, 8, 16, 32, 64\}$. One evaluation per dataset per method per rank has been performed due to time and computational power limitations. All datasets described in Chapter 3 were employed. More information about training hyper-parameters in Appendix A.

Figure 4.4 shows average accuracy across datasets over increasing matrix ranks in order to compare the performance of LoRA and DoRA techniques.

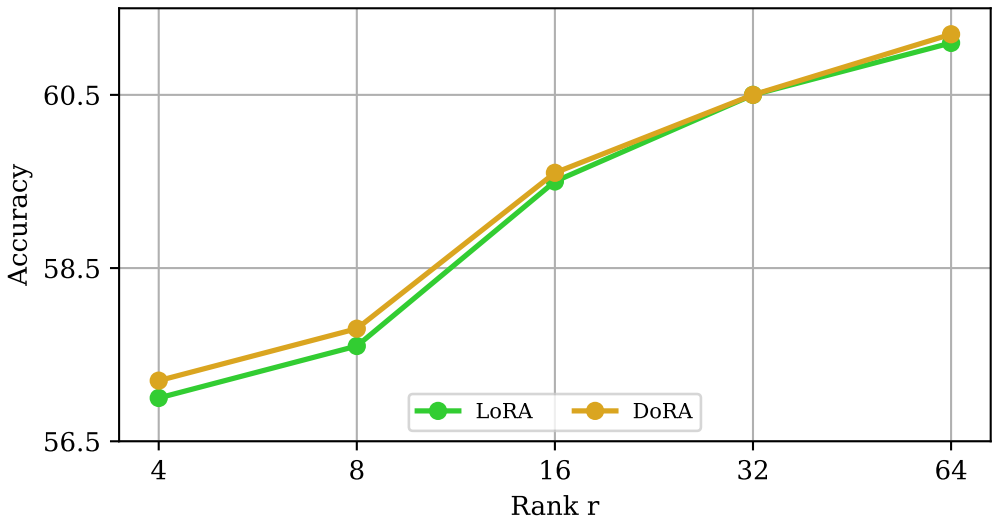

Figure 4.4: Comparative average accuracy across commonsense reasoning tasks over an increasing matrix rank for LoRA and DoRA.

At first sight, Figure 4.4 shows how DoRA performs slightly better than LoRA in all cases. This difference is bigger when the intrinsic rank is lower, which correlates with the results from the results reported in [30]. The difference obtained between both methods could be bigger, since only Query, Key and Value matrices from the Transformers were fine-tuned. In addition, the model employed, Phi-1.5, has less parameters than common large language models. Thus, the main factors that make the results reported different from those in [30] are the size of the LLMs employed and the amount of targeted matrices for fine-tuning.

4.7 Conclusions

During this chapter we have discussed how LoRA's family can be understood as additive fine-tuning, learning the correct bias to be added to the original result. This implication help us to understand its advantages, making these methods the most valuable from the PEFT methods due to its adaptability, efficiency and good performance.

From the experimental perspective, the first conclusion obtained is the results obtained by using LoKr, which are not as good as those reported using LoRA or DoRA. LoKr distribution of trainable parameters doesn't seem a good choice for fine-tuning LLMs, since both accuracy and time cost are worse than LoRA itself. We can also conclude that DoRA performs slightly yet consistently better than LoRA. From the results obtained, the maximum average accuracy obtained is when $r = 64$ and using DoRA, achieving 61.2 of accuracy. In this case, increasing the rank more than 64 could have given better results.

CHAPTER 5
OFT: Orthogonal fine-tuning

5.1 Introduction

This chapter introduces a new PEFT technique that is focused on orthogonality. As explained in Section 1.2, one of the goals of this work is to compare some fine-tuning techniques in terms of performance after adapting Phi-1.5 model. Section 5.2 is dedicated to introduce the main orthogonal PEFT technique. Section 5.3 shows a variant for making fine-tuning with orthogonal matrices. Section 5.4 performs experiments on the tasks commented in Chapter 3 and compares the results with those reported in Section 4.6 with other PEFT techniques. Finally, Section 5.5 is devoted to give detailed conclusions of this chapter.

5.2 OFT

The PEFT methods discussed in Sections 4.3, 4.4 and 4.5 have certain drawbacks that need consideration. While LoRA, DoRA, and LoKr concentrate on learning matrices that, when multiplied with the input, yield the necessary bias to augment the original result, Orthogonal Fine-Tuning (OFT) [31] takes a different approach. OFT is centered around learning an optimal distance-preserving transformation to modify the output, disregarding scaling transformations. In other words, OFT aims to learn the matrix $R \in SO(m)$ such that:

$$M' = RM \tag{5.1}$$

where $M, M' \in \mathbb{R}^{m \times n}$. To understand the effect of distance-preserving transformations, it is needed to understand hyperspherical energy. Hyperspherical energy is defined as the sum of hyperspherical similarity (e.g, cosine similarity) of M'. By performing OFT, hyperspherical energy remains the same during the fine-tuning stage. This means that the distance between subspaces remains the same. In the case of LoRA's family, hyperspherical energy could vary. OFT authors claim that it is preferable to preserve the same hyperspherical energy during fine-tuning [31].

Standard orthogonalization methods such as Gram-Schmidt method, despite differentiable, are often too expensive to compute in practice. For better efficiency, OFT adopt Cayley parametrization to generate the orthogonal matrix R:

$$R = (I + Q)(I - Q)^{-1} \tag{5.2}$$

where Q is a skew-symmetric matrix satisfying $Q = -Q^t$. Even using Cayley transform to parameterize the unitary matrix, making R can still be very parameter-inefficient with

a large dimension. To address this, R is represented as a block-diagonal matrix with r blocks, leading to the following form:

$$R = \begin{pmatrix} R_1 & & & \\ & R_2 & & \\ & & \ddots & \\ & & & R_r \end{pmatrix} \tag{5.3}$$

where $R_1, R_2, \ldots, R_r \in SO(\frac{n}{r})$ are obtained using Equation 5.2. OFT authors claim that the most effective update to modify the semantics is to change neuron directions, which is exactly what OFT is designed for.

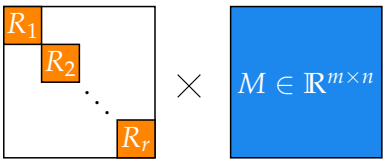

Figure 5.1: OFT reparametrization.

If orthogonalization is desired, there are two possible ways to achieve it. The first one is to produce an orthogonal differentiable matrix. The other option is to restrict the matrix, by adding a term in the loss function, to ensure that the matrix is close to be orthogonal.

The second option is simpler to execute but doesn't guarantee orthogonality in the resulting matrix. The first option yields the intended outcome, yet constructing fast differentiable orthogonal matrices suppose a challenge. As illustrated in Figure 5.1, employing the Cayley transformation requires to construct skew-symmetric matrices and subsequently transform them to each block, involving numerous inverse computations. This process significantly elevates time expenses due to the operations required to derive the resultant matrix R.

In addition to computing the unitary matrix R, the authors of OFT also introduce a straightforward extension to the original method. This extension involves learning a magnitude scaling for each neuron. This addition is motivated by the observation that scaling transformations do not alter the hyperspherical energy, given that vectors are normalized for computing it. Consequently, the resulting matrix M' is as follows:

$$M' = RMD \tag{5.4}$$

where R is computed using Equation 5.2 and $D = \text{diag}(s_1, \ldots, s_n)$ is the learnable scaling transformation. Whether to put D at the beginning or at the end is a choice made by the authors, though it is easy to see that $D_1 RM = RMD_2$ only if $D_1 = D_2 = I$. Therefore, despite author experiments show that using this method gives better results than usual OFT, the scaling transformation may not be correctly placed in all possible cases.

5.3 BOFT

Another approach to orthogonalization is Orthogonal Fine-tuning via Butterfly Factorization (BOFT) [32]. The butterfly structure, originally employed in the Cooley-Tukey algorithm for fast Fourier transform, has also been adopted as a network topology for efficient data exchange in computing. An example of this factorization is provided in Figure 5.2.

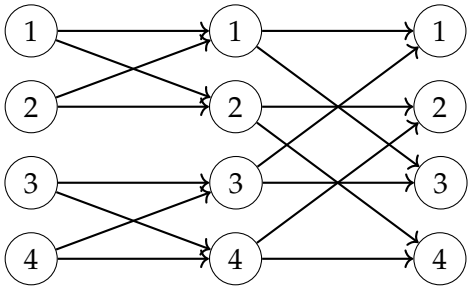

Figure 5.2: Butterfly structure for $d = 4$

In Fourier transform, a local modification in the frequency domain can induce a global change in the spatial domain, similar to our information transmission problem where each node at the initial level communicates with every node at the final level.

Because the final orthogonal matrix in BOFT results from multiplying several orthogonal matrices, the training runtime overhead is greater compared to OFT and other PEFT methods mentioned earlier. Additionally, it's uncertain whether the butterfly network is the most efficient method for transmitting information.

5.4 Experiments

In this section, comparative results among the PEFT techniques are presented. However, for the sake of simplicity, results with the BOFT technique have been discarded for discussion since it does not improve those achieved by OFT, LoRA and DoRA.

For comparing LoRA, DoRA and OFT, we will study its effects varying the intrinsic rank $r \in \{4, 8, 16, 32, 64\}$. One evaluation per dataset per method per rank has been performed due to time and computational power limitations. For making a 'fair evaluation' between all methods, OFT rank needs to be computed as a function of LoRA's rank in order to have approximately the same number of trainable parameters. This is due to the different meaning of the rank in both methods [26, 31]. No scaling transformations have been employed for OFT. All datasets described in Chapter 3 were employed. More information about training hyper-parameters in Appendix A.

Figure 5.3 shows average accuracy across datasets over increasing matrix ranks in order to compare the performance of LoRA, DoRA and OFT techniques.

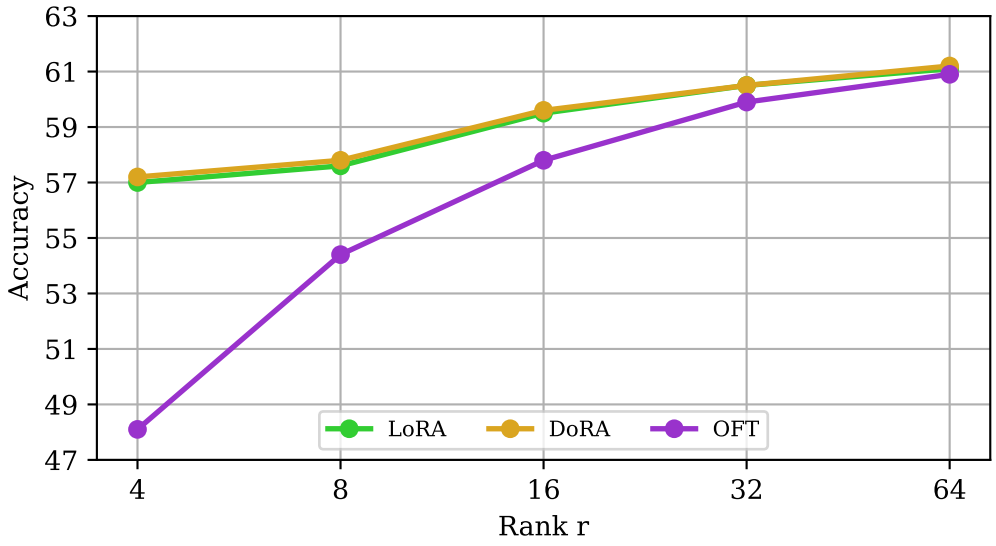

Figure 5.3: Comparative average accuracy across commonsense reasoning tasks over an increasing matrix rank for LoRA, DoRA and OFT.

It can be easily in Figure 5.3 seen that OFT performs significantly worse when the rank is lower. This can be influenced by the fact of computing similar amount of trainable parameters. For low ranks in OFT, there are many tiny blocks in R matrix, which do not perform empirically well. For higher ranks, the blocks are grouped into bigger ones in R, leading to better results.

Since OFT does not perform random initialization [31], only one evaluation per dataset per rank is enough to show its weakness against LoRA and DoRA.

5.5 Conclusions

This chapter is focused on showing another approach to do fine-tuning. Instead of adopting additive fine-tuning, a multiplicative fine-tuning approach is presented as a possible alternative.

From the experiments perform on the previous section, the main conclusion is the worse results obtained from BOFT. Although they were not as bad as the ones obtained with LoKr, both time and computational resources were excessive comparing to other PEFT techniques. BOFT tries to perform orthogonality in another way than OFT, but it does not obtain competitive results when used for fine-tuning LLMs. This is also supported by the authors [32], which mention its weaknesses.

Additionally, as observed in Figure 5.3, OFT performs similarly than LoRA and DoRA for $r \geq 32$, although being slightly worse in those cases. OFT is not a good choice when having low resources for increasing r, since low ranks perform worse. In addition, OFT time cost is $\times 3$ higher than LoRA. Thus, although OFT tries to introduce orthogonality as a better technique, it does not appear to give better results in LLMs. Possibly, scaling transformations are needed together with distance-preserving transformations in order to correctly fine-tune LLMs.

CHAPTER 6
SHOFT: SVD Householder orthonormal fine-tuning

6.1 Introduction

This chapter proposes a designed PEFT technique, SHOFT. As commented in section 1.2, one of the goals of this work is to develop a fine-tuning technique in order to improve performance after adapting Phi-1.5 model. Sections 6.2 and 6.3 explain some mathematical concepts needed to understand the proposed PEFT method. Section 6.4 is dedicated to introduce the insights of the method proposed and how it works. Section 6.5 shows the initialization of the proposed method, comparing it to other PEFT initializations. Section 6.6 performs experiments on the tasks commented in Chapter 3 and compares the results with those reported in Sections 4.6 and 5.4 with other PEFT techniques. Finally, Section 6.7 is devoted to give detailed conclusions of this chapter.

6.2 Singular value decomposition

The practical and theoretical importance of the singular value decomposition (SVD) [4] is hard to overestimate. If A is a real $m \times n$ matrix, then there exist unitary matrices:

$$U = [u_1 \mid \cdots \mid u_m] \in SO(m) \qquad V = [v_1 \mid \cdots \mid v_n] \in SO(n) \tag{6.1}$$

such that

$$U^t A V = \Sigma = \text{diag}(\sigma_1, \ldots, \sigma_p) \in \mathbb{R}^{m \times n}, \quad p = \min\{m, n\} \tag{6.2}$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$.

One important aspect of SVD is its geometric interpretation: any matrix A can be decomposed into two distance-preserving transformations U, V and a scaling transformation Σ.

6.3 Householder transformation

Let $u \in \mathbb{R}^n$ be a vector of unit length ($\|u\|_2 = 1$). Then

$$H = I - 2uu^t \tag{6.3}$$

is said to be a Householder transformation or reflector [4]. A Householder transformation is a linear transformation that describes a reflection about a hyperplane containing the origin.

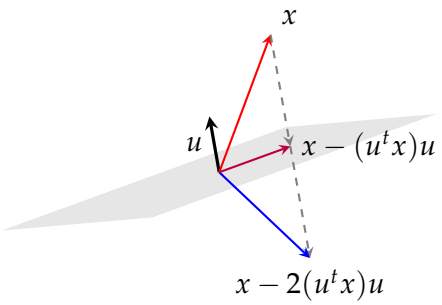

Figure 6.1: Householder transformation on $\mathbb{R}^3$

Figure 6.1 show how the x vector is projected into the hyperplane made by the normalized u vector and then reflected. The general formulation of Householder transformation for a given nonzero vector $u \in \mathbb{R}^n$ is the following

$$H = I - \frac{uu^t}{\tau} \tag{6.4}$$

where $\tau = \frac{\|u\|_2^2}{2}$. It can be demonstrated that for any matrix $U \in SO(n)$, it can be decomposed as a composition of Householder transformations [6] as follows

$$U = H_1 H_2 \cdots H_n \tag{6.5}$$

6.3.1. CWY transform

Computing the product of multiple matrices is costly. Thus, numeric methods are needed to facilitate this computation. In our case, the action of multiple Householder transformations and cast it in terms of high-performance matrix-matrix products was proposed in the late 1980s. In particular, the WY transform and the CWY transform were proposed. From [33] we refer to a theorem that ensures the properties mentioned before:

Theorem 1 *Let the matrix $U \in \mathbb{R}^{m\times k}$ have linearly independent columns. Partition U by columns as*

$$U = (u_0 \mid u_1 \mid \ldots \mid u_{k-1}) \tag{6.6}$$

and consider the vector $t = (\tau_0, \tau_1, \ldots, \tau_{k-1})^t$ with $\tau_i \neq 0, 0 \leq i < k$. Then, there exists a unique nonsingular upper triangular matrix $S \in \mathbb{R}^{k\times k}$ such that

$$\left(I - \frac{u_0 u_0^t}{\tau_0}\right)\left(I - \frac{u_1 u_1^t}{\tau_1}\right)\cdots\left(I - \frac{u_{k-1} u_{k-1}^t}{\tau_{k-1}}\right) = I - USU^t \tag{6.7}$$

S can be computed by the following three steps:

1. $S :=$ the upper triangular part of U^tU.
2. Divide the diagonal elements of S by two.
3. $S := S^{-1}$.

However, step 3 can be omitted by using the UT transform, considering $T^{-1} = S$. The basic idea is to compute $M := T^{-1}U^t$ as a triangular solve instead of computing the inverse and the profuct of both matrices, reducing the number of floating point operations performed.

6.4 Insights and method

The purpose of SHOFT is to divide the strategy of fine-tuning the pretrained matrix into its two atomic parts: fine-tuning distance-preserving transformations and scaling transformations separately. Given a matrix $M \in \mathbb{R}^{m\times n}$, we can decompose it using singular value decomposition:

$$M = U\Sigma V^t \tag{6.8}$$

where $U \in SO(m)$ and $V^t \in SO(n)$ are distance-preserving transformations and $\Sigma \in \mathbb{R}^{m\times n}$ is a scaling transformation. For fine-tuning U and V, we will employ accumulated Householder transformations. Additionally, in order to compute faster this transformations, UT transform described in Section 6.3 is used.

Considering that $W_U \in \mathbb{R}^{m\times r_U}, W_V \in \mathbb{R}^{n\times r_V}$ are trainable matrices containing r_U and r_V vectors that describe the Householder transformations, we compute their respective accumulated Householder matrices $H_U \in SO(m), H_V \in SO(n)$. For fine-tuning the scaling transformation Σ, we fine-tune all its elements, since Σ only contains the singular values of M. Thus, SHOFT can be formulated as follows:

$$M' = H_U \cdot U \cdot \Sigma' \cdot V^t \cdot H_V = U'\Sigma'V' \tag{6.9}$$

As noted in [9], any unitary matrix can be decomposed into a product of Householder transformations. This suggests that using a greater number of Householder vectors yields better approximations of any unitary matrix. Indeed, SHOFT can be seen as a method for fine-tuning the SVD of the original matrix, learning both the singular vectors and singular values. Consequently, from a theoretical standpoint, SHOFT can approximate any matrix $M' \in \mathbb{R}^{m\times n}$ by employing a sufficient number of Householder vectors, as it can approximate any singular value decomposition of any matrix.

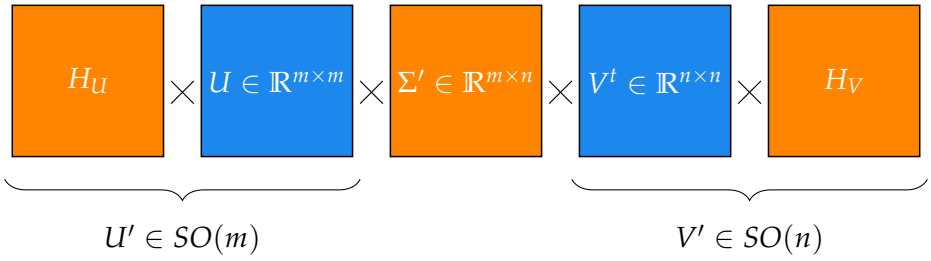

Figure 6.2: SHOFT reparametrization.

SHOFT might initially be perceived solely as an orthogonal PEFT method. Nevertheless, it is important to recognize that SHOFT cleverly incorporates the use of a scaling transformation. As discussed earlier in Section 5.4, OFT can utilize a scaling transformation,

though its correct position was uncertain. In contrast, SHOFT effectively leverages the algebraic structure provided by SVD to achieve this goal.

Some advantages of SHOFT include:

- It employs orthogonal transformations, which, as discussed in section 5.2, are preferable to additive weight updates.
- It wisely uses scaling and distance-preserving transformations to correctly separate magnitude and direction.
- It is more interpretable than typical PEFT techniques. SHOFT fine-tunes both singular vectors and singular values, providing insight into their variations during training.
- It is faster than standard orthogonal PEFT techniques due to the use of CWY and UT transforms.
- There is no additional inference overhead because, during deployment, the learned matrix M' can be explicitly computed.

In terms of memory, SHOFT needs to compute the SVD and store the U and V matrices as non-trainable parameters at the beginning of the training process. This approach increases both time and memory costs. However, since the SVD is only computed at the start, its cost can be amortized over the duration of the training.

On the other hand, removing the original matrix helps free up some memory, though it does not eliminate the additional memory usage entirely. To completely resolve this issue, the current method must be reformulated. This problem will be addressed in the next chapter by introducing a generalization of the designed method.

6.5 Initialization

Initialization plays a crucial role in the training process, impacting both the speed of convergence and the quality of the final result. Random initialization have shown better results than constant initialization by breaking symmetry and improving generalization. One constrain imposed in PEFT techniques is that the initialization needs to ensure that, at the beginning, the matrix remains untouched ($M' = M$).

While LoRA, DoRA, LoKr can be randomly initialized, OFT cannot, due to also needing to preserve orthogonality ($Q = \mathbf{0}$ to ensure $R = I$). In general, orthogonal PEFT methods cannot be randomly initialized. By contrast, SHOFT can be randomly initialized by considering consecutive pairs of equal vectors. Since H_U, H_V are accumulated reflections, we can obtain the identity matrix by putting together two identical vectors that express the same reflection:

$$\underbrace{\left(I - \frac{u_0 u_0^t}{\tau_0}\right)\left(I - \frac{u_0 u_0^t}{\tau_0}\right)}_{I} \cdots \underbrace{\left(I - \frac{u_r u_r^t}{\tau_r}\right)\left(I - \frac{u_r u_r^t}{\tau_r}\right)}_{I} = I \tag{6.10}$$

Thus, if k is even, we can generate $r = \frac{k}{2}$ pairs of random vectors. If k is odd, we can generate $\lfloor \frac{k}{2} \rfloor$ pairs of random vectors and a zero vector. Vectors in W_U, W_V will be picked from a random high-dimensional uniform distribution of range $[-1, 1]$. This distribution

is considered the best option in order to correctly sample the unit hipersphere. In the case of Σ', it will be initialized to the original singular values, Σ.

6.6 Experiments

In this section, SHOFT is compared against the best performing PEFT technique from Chapters 4 and 5. As in previous chapters, this comparison is performed in terms of accuracy as a function of the intrinsic rank $r \in \{4, 8, 16, 32, 64\}$. Five evaluations per dataset per rank have been performed in this case for obtaining more precise results. All datasets described in Chapter 3 were employed. For all comparisons, $r_U = r_V = r$ in order to perform 'fair evaluations'.

DoRA results produced for making Figures 4.4 and 5.3 will be used to alongside with the new evaluations performed. More information about training hyper-parameters in Appendix A.

6.6.1. Experiments on WinoGrande

For WinoGrande dataset, described in section 3.6, DoRA will be employed to make the comparison, presented in Figure 6.3.

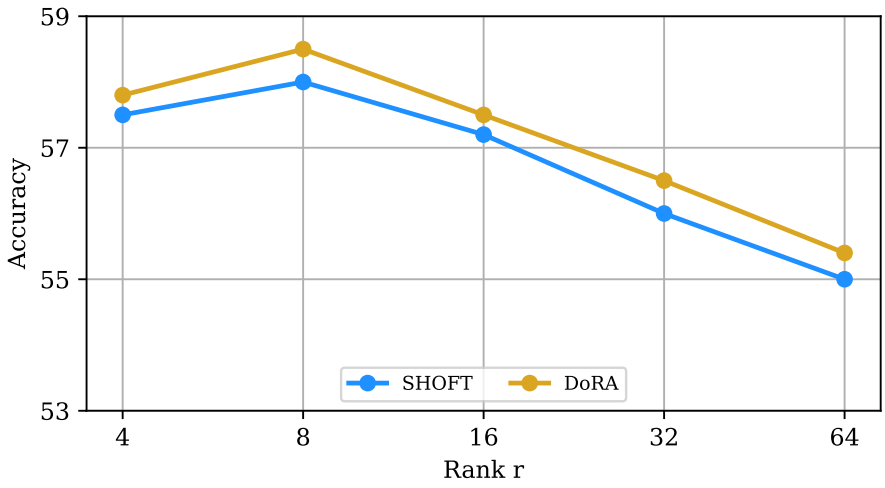

Figure 6.3: Comparative average accuracy on WinoGrande over an increasing matrix rank for DoRA and SHOFT.

As observed in Figure 6.3, there is a clear decrease tendency after $r = 8$. DoRA performs slightly better, although the maximum difference between both methods is less than 0.5 of accuracy. Therefore, we can't conclude there is a significant difference between both methods in this task.

6.6.2. Experiments on BoolQ

For BoolQ dataset, described in section 3.3, DoRA will be employed to make the comparison, presented in Figure 6.4.

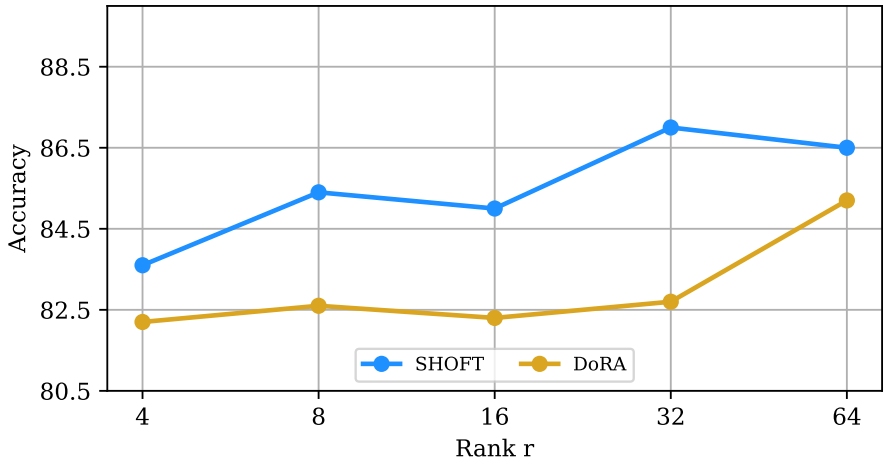

Figure 6.4: Comparative average accuracy on BoolQ over an increasing matrix rank for DoRA and SHOFT.

As observed in Figure 6.4, there's a slightly increase tendency. In this case, we can see how SHOFT outperforms DoRA, giving higher results. Therefore, in BoolQ SHOFT gives better performance.

6.6.3. Experiments on PIQA

For PIQA dataset, described in section 3.4, DoRA will be employed to make the comparison, presented in Figure 6.5.

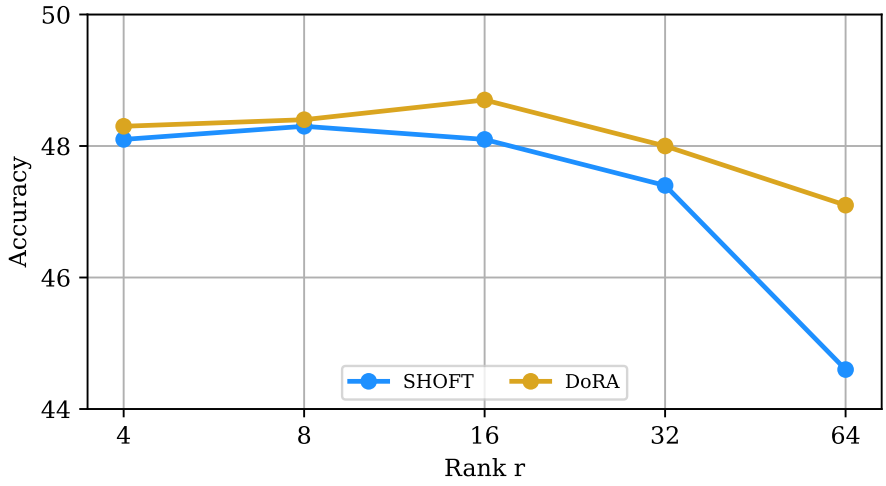

Figure 6.5: Comparative average accuracy on PIQA over an increasing matrix rank for DoRA and SHOFT.

As observed in Figure 6.5, there's a logarithmic decrease tendency. In this case, we can see how DoRA outperforms SHOFT. In this task, increasing r supposed an accuracy loss in both methods, which was commented in Section 4.3. Therefore, in PIQA DoRA gives better performance.

6.6.4. Experiments on SIQA

For BoolQ dataset, described in section 3.5, DoRA will be employed to make the comparison, presented in Figure 6.6.

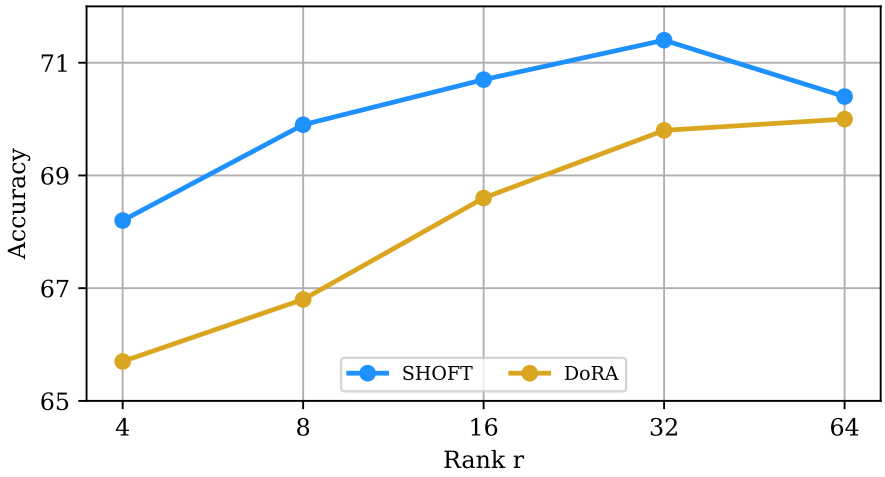

Figure 6.6: Comparative average accuracy on SIQA over an increasing matrix rank for DoRA and SHOFT.

As observed in Figure 6.6, there is a clear increase tendency with a peak at $r = 32$. In this case, SHOFT clearly outperforms DoRA. This fact is more noticeable in lower ranks. Thus, in SIQA SHOFT gives better performance.

6.6.5. Experiments on ARC-e

For ARC-e dataset, described in section 3.7, DoRA will be employed to make the comparison, presented in Figure 6.7.

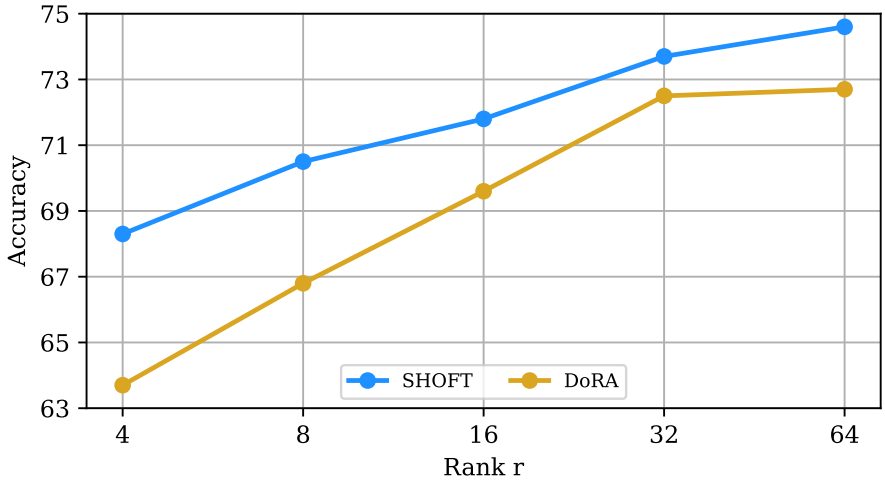

Figure 6.7: Comparative average accuracy on ARC-e over an increasing matrix rank for DoRA and SHOFT.

As observed in Figure 6.7, there is a clear increase tendency. In this case, SHOFT outperforms DoRA along all ranks explored. Thus, in ARC-e SHOFT gives better performance.

6.6.6. Experiments on ARC-c

For ARC-c dataset, described in section 3.7, LoRA will be employed to make the comparison, presented in Figure 6.8.

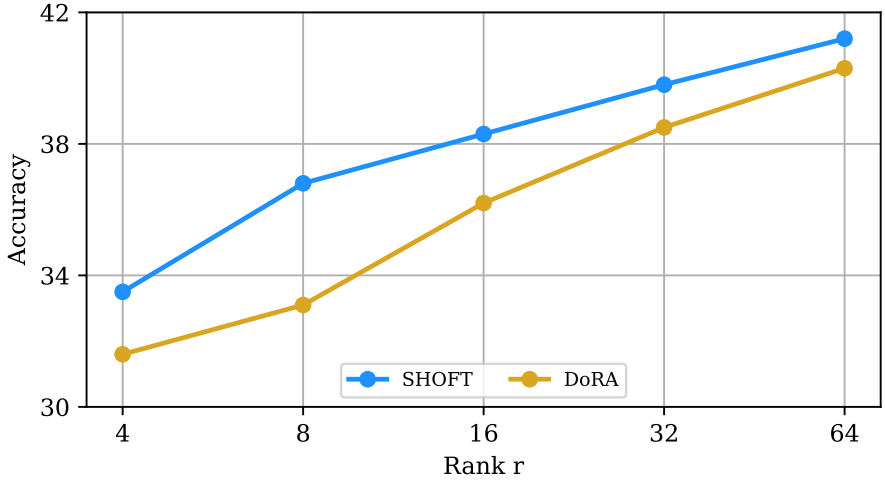

Figure 6.8: Comparative average accuracy on ARC-c over an increasing matrix rank for DoRA and SHOFT.

As observed in Figure 6.8, there is a clear increase tendency. This make sense since in the previous subsection similar results were obtained with ARC-e subset. In this case, SHOFT outperforms DoRA along all ranks explored. Thus, in ARC-c SHOFT gives better performance.

6.6.7. Experiments on OBQA

For OBQA dataset, described in section 3.8, DoRA will be employed to make the comparison, presented in Figure 6.9.

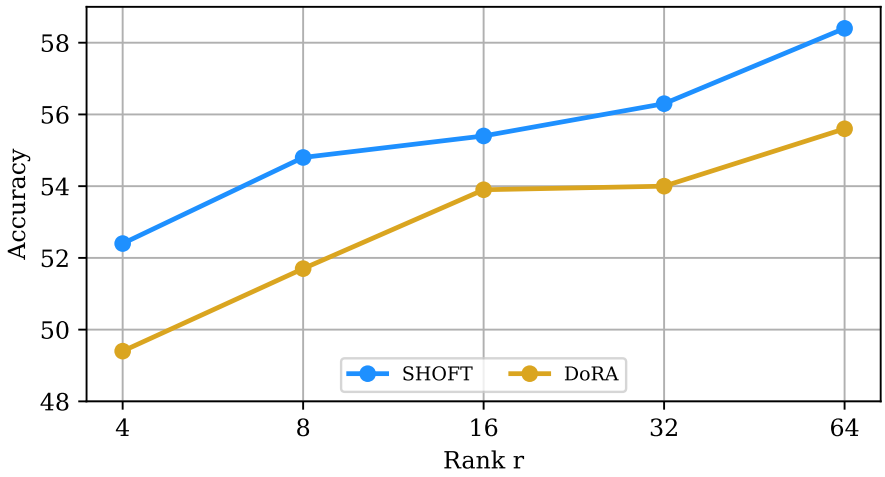

Figure 6.9: Comparative average accuracy on OBQA over an increasing matrix rank for DoRA and SHOFT.

As observed in Figure 6.9, there is a clear increase tendency. In this case, SHOFT outperforms DoRA along all ranks explored. Thus, in OBQA SHOFT gives better performance.

6.6.8. Average results

As done in sections 4.6 and 5.4, we will also study the comparative average accuracy across commonsense reasoning tasks over an increasing matrix rank for DoRA and SHOFT. This results are presented in Figure 6.10.

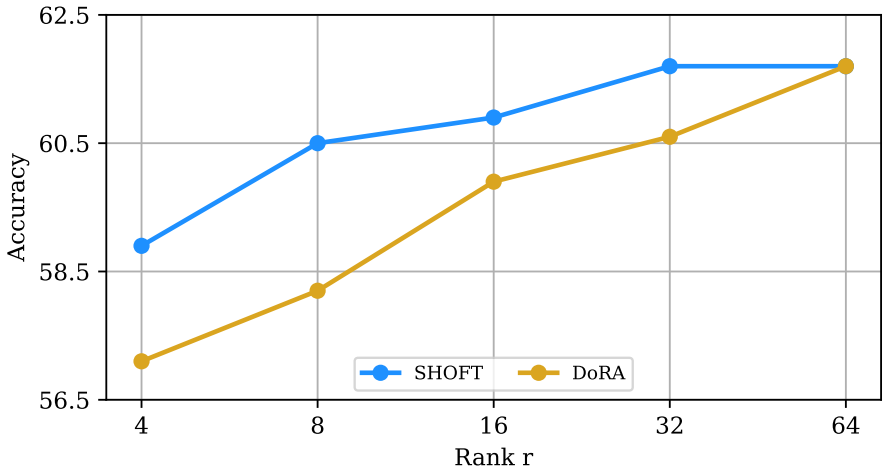

Figure 6.10: Comparative average accuracy across commonsense reasoning tasks over an increasing matrix rank for DoRA and SHOFT.

As observed in Figure 6.10, there's a clear increase tendency. In this case, SHOFT clearly performs better, although at $r = 64$ both methods provide similar results. Even in lower ranks, SHOFT obtains high performance compared with DoRA. Thus, SHOFT gives better results in average than DoRA.

6.7 Conclusions

The purpose of this chapter was to design a PEFT technique that improves the weaknesses presented on previous chapters. The role of Householder reflections and SVD played a crucial role for achieving this objective.

A random initialization for the multiplicative fine-tuning method was proposed, which was not previously tried by other orthogonal PEFT techniques. This was suggested to improve SHOFT capabilities while preserving orthogonality.

Experimental results have shown that SHOFT consistently outperforms DoRA on most tasks. This can also be observed in Figure 6.10, in which clearly SHOFT outperforms DoRA.

CHAPTER 7
RHOFT: Randomized SHOFT

7.1 Introduction

This chapter proposes a generalization of the designed PEFT technique commented in the previous chapter, RHOFT. As explained in Section 1.2, one of the goals of this work is to develop a fine-tuning technique in order to improve performance after adapting Phi-1.5 model. Section 7.2 explain some mathematical concepts needed for understanding the variant PEFT method. Section 7.3 makes an analysis of the distribution of the singular values. Section 7.4 is dedicated to introduce the insights of the method proposed and how it works. Section 7.5 performs experiments on the tasks commented on Chapter 3 and compares the results with the ones obtained in Sections 4.3, 5.4 and 6.6 with other PEFT techniques. Finally, Section 7.6 is devoted to give detailed conclusions of this chapter.

7.2 Randomized singular value decomposition

Randomized singular value decomposition (randomized SVD) is a technique used to approximate a given matrix with less singular values than usual SVD. If A is a real $m \times n$ matrix and $k \in \mathbb{N}, k \leq \min\{m, n\}$, then we can compute:

$$U = [u_1 \mid \cdots \mid u_k] \in SO(m) \qquad V = [v_1 \mid \cdots \mid v_k] \in SO(n) \tag{7.1}$$

such that

$$A \approx U\Sigma V^t \tag{7.2}$$

where $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_k)$ are the highest k approximated eigenvalues. Halko, Martinsson, and Tropp [34], demonstrated that a modular framework could be used for good randomized matrix approximations.

The geometric interpretation of SVD is crucial because it reveals the intrinsic structure and dimensionality of the matrix learned. By isolating the directions of maximum variance (through the singular values), SVD helps identify the most important features and reduces the dimensionality without significant loss of information.

Randomized SVD is faster to compute that usual SVD and approximates only the desired number of singular vectors and singular values.

7.3 SVD curve analysis

In order to understand RHOFT, it would be interesting to see how the magnitude of the largest singular values of the targeted matrices are distributed. That is, the curve made by sorting them for each target.

For this purpose, Phi-1.5 Key, Query and Value matrices will be employed. Figures 7.1, 7.2 and 7.3 show the average curves of the Query, Key and Value matrices respectively. The X axis represent the position that occupies the singular value, and the Y axis represents its associated magnitude. Since there are 24 matrices for each target, the representation of the curve will be an average of the respective matrices. All quartiles are represented in the form of dots, allowing to show the cumulative sum of the singular values at 4 stages.

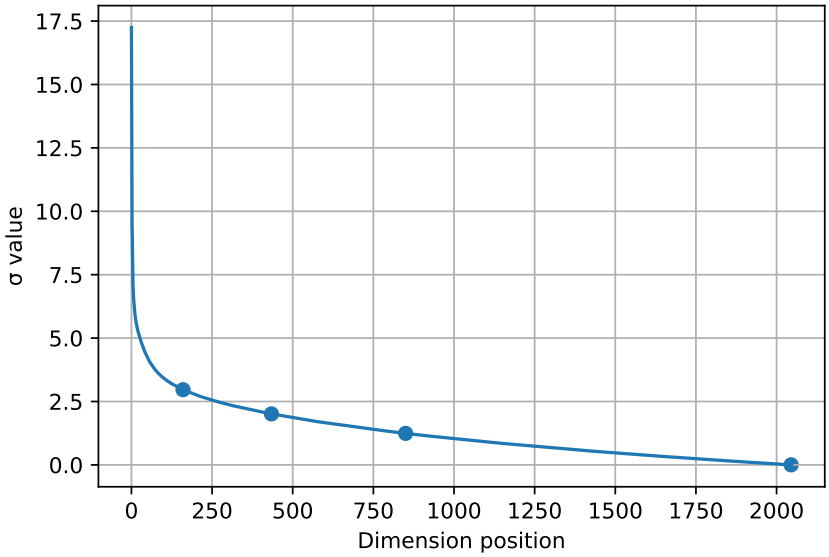

Figure 7.1: SVD curve of Query matrix

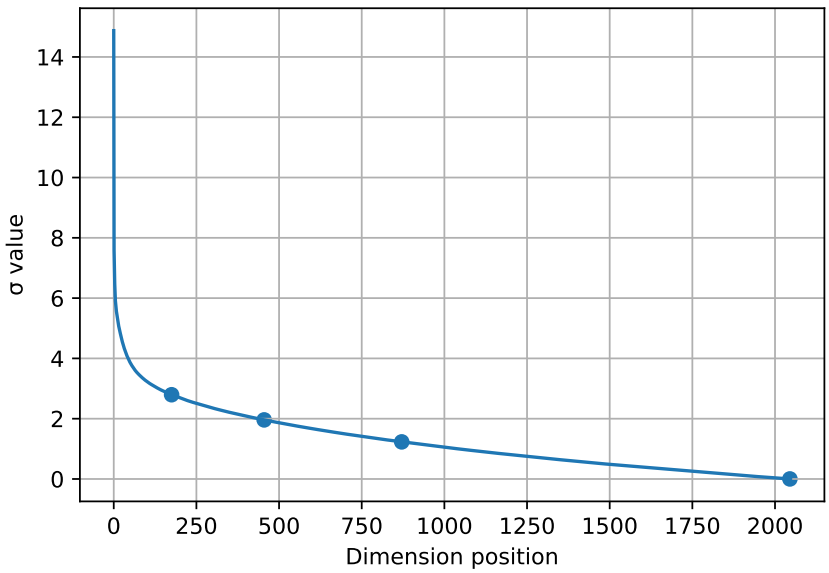

Figure 7.2: SVD curve of Key matrix

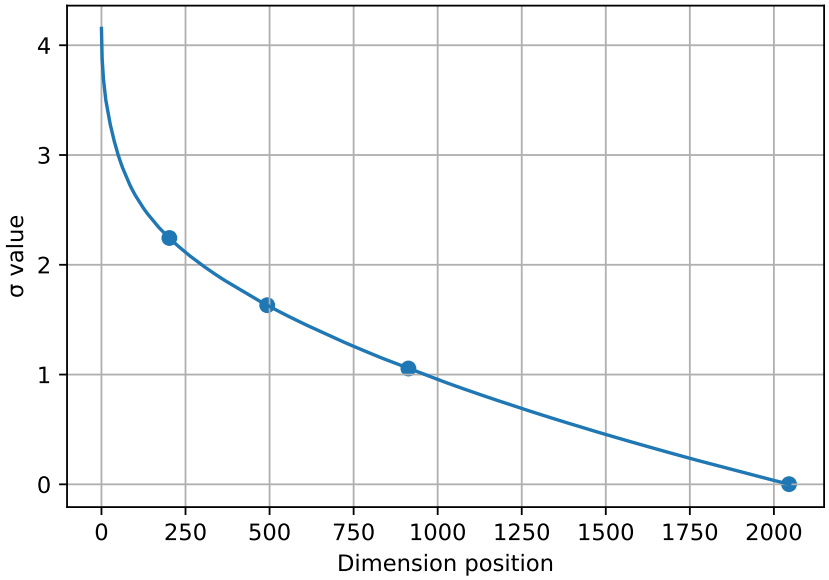

Figure 7.3: SVD curve of Value matrix

Both Query and Key curves show high top singular values, while Value curve largest singular value is less than 5. However, although their different shapes, all curves share a common property: first, second and third quartiles are situated before 250, 500 and 1000 respectively.

The main implication of the curves represented is the importance of each dimension in order to fine-tune. For instance, a dimension with a singular value of less than 1 is less important than a dimension with a singular value of 14. Lower singular values mean that the dimension learned does not contain many useful information.

7.4 Insights and methods

As mentioned in Section 6.4, one issue concerning SHOFT is that it increases the size of the model while training because of the need of storing matrices U, V from SVD. This can be solved by computing the randomized SVD instead of the full SVD. Given a number k, the randomized SVD reconstructs the top k singular values:

$$M \approx U\Sigma V^t \tag{7.3}$$

where $U \in \mathbb{R}^{m\times k}$ and $V^t \in \mathbb{R}^{k\times n}$ are distance-preserving transformations combined with projection transformations, and $\Sigma \in \mathbb{R}^{k\times k}$ is a scaling transformation. For initialize and fine-tune U, V and Σ, we will do the same approach as in SHOFT.

By choosing $k \ll \min\{m, n\}$, we can reduce the number of non-trainable parameters we must save by loosing some precision in the decomposition. In order to maintain the same number of non-trainable parameters as the pretrained matrix, then $k = \frac{mn}{m+n}$. For square matrices, that limit is $k = \frac{d}{2}$.

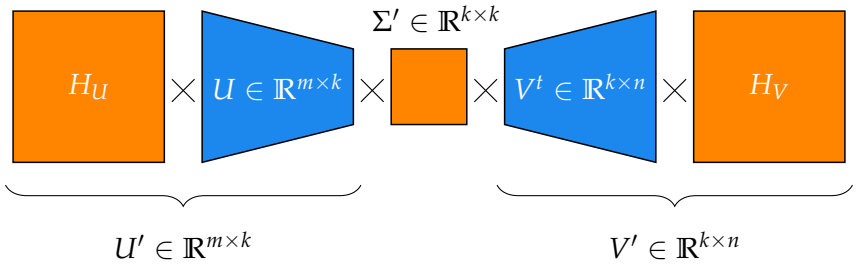

Figure 7.4: RHOFT reparametrization.

If k is lower than this limit, then RHOFT employs less non-trainable parameters than the pretrained matrix. This fact opens new opportunities not previously contemplated due to memory overhead, such as:

- Fine-tuning more layers while training.
- Use more trainable parameters during training.
- Train large models by reducing the total amount of parameters in training.

Thus, for a low k, RHOFT can also be used as a reduction technique for training the model, in addition to being a fine-tuning technique.

In order to give a better understanding on how removing lower singular values can give good results, a geometric approach can be contemplated. Considering both singular values and vectors form a hypersphere, we can deduce that small singular values can be removed in order to flatten the hypersphere in one dimension. If the singular value associated was small enough, the resulted hypersphere will be mostly equivalent to the original.

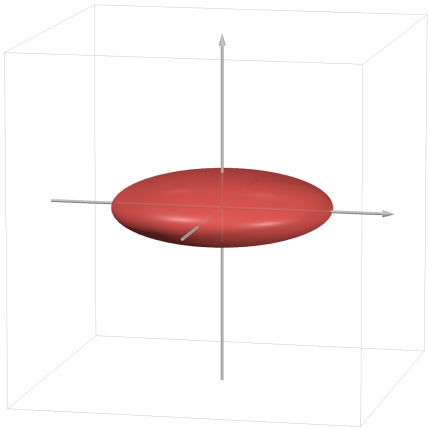

Figure 7.5: Hypersphere on $\mathbb{R}^3$

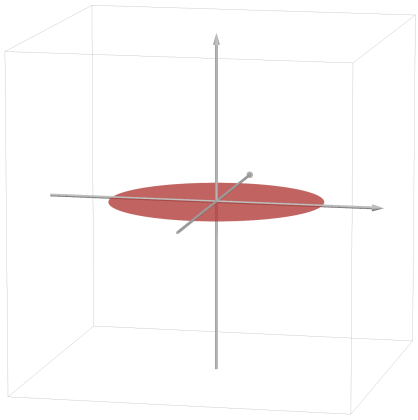

Figure 7.6: Hypersphere on $\mathbb{R}^2$

For instance, as shown in Figure 7.5, we can consider a hypersphere on $\mathbb{R}^3$ in which on one of its axis is mostly flatten. Thus, Figures 7.5 and 7.6 show shapes that are similar when stretched enough. As observed in Section 7.3, this is the case to the targeted matrices, since the curves seem to follow nonlinear decreases.

7.5 Experiments

In this section, RHOFT is compared against SHOFT and the best performing PEFT technique from Chapters 4 and 5. As in previous chapters, this comparison is performed in terms of accuracy as a function of the intrinsic rank $r \in \{4, 8, 16, 32, 64\}$. Five evaluations per dataset per rank have been performed in this case for obtaining more precise results. All datasets described in Chapter 3 were employed. For all comparisons, $r_U = r_V = r$ in order to perform 'fair evaluations'. In the case of RHOFT, $k = 1024$, which means that less than 25% of the lowest singular values are removed for making RHOFT as memory efficient as DoRA.

DoRA results produced for making Figures 4.4 and 5.3 will be used to alongside with the new evaluations performed. More information about training hyper-parameters in Appendix A.

7.5.1. Experiments on WinoGrande

For WinoGrande dataset, described in section 3.6, DoRA will be employed to make the comparison, presented in Figure 7.7.

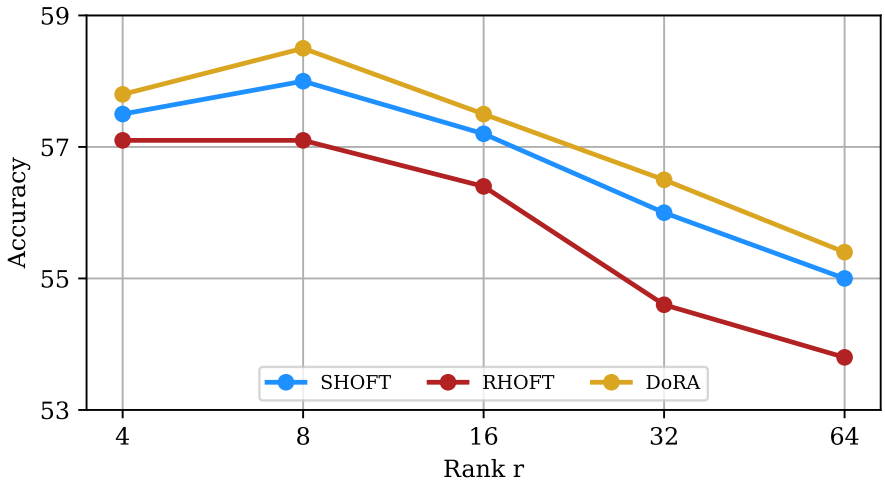

Figure 7.7: Comparative average accuracy on WinoGrande over an increasing matrix rank for DoRA and RHOFT.

As observed in Figure 7.7, RHOFT follows the same tendency as SHOFT. It gives slightly worse results than SHOFT, as expected.

7.5.2. Experiments on BoolQ

For BoolQ dataset, described in section 3.3, DoRA will be employed to make the comparison, presented in Figure 7.8.

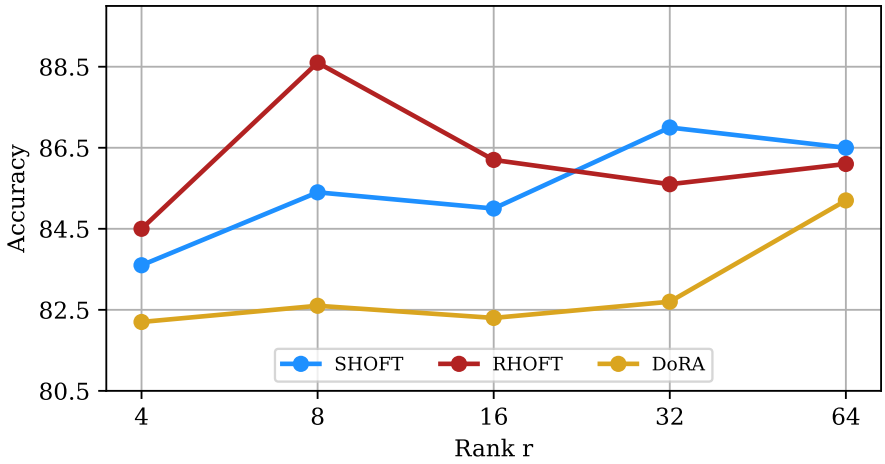

Figure 7.8: Comparative average accuracy on BoolQ over an increasing matrix rank for DoRA and RHOFT.

As observed in Figure 7.8, RHOFT does follow a similar tendency as SHOFT, although there is a peak at $r = 8$. Except to that peak, RHOFT performs similarly to SHOFT in average in BoolQ.

7.5.3. Experiments on PIQA

For PIQA dataset, described in section 3.4, DoRA will be employed to make the comparison, presented in Figure 7.9.

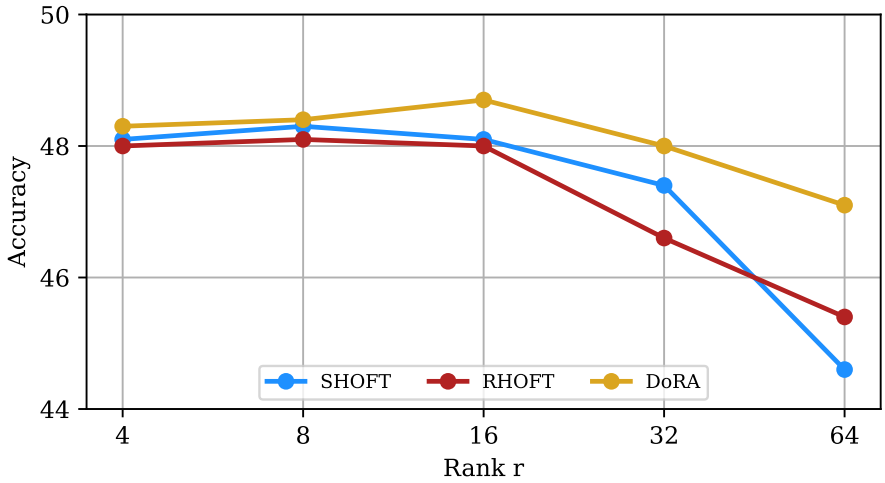

Figure 7.9: Comparative average accuracy on PIQA over an increasing matrix rank for DoRA and RHOFT.

As observed in Figure 7.9, RHOFT follow a similar tendency as SHOFT, although it performs slightly better at $r = 64$. In this case, RHOFT is more robust than SHOFT in PIQA.

7.5.4. Experiments on SIQA

For SIQA dataset, described in section 3.5, DoRA will be employed to make the comparison, presented in Figure 7.10.

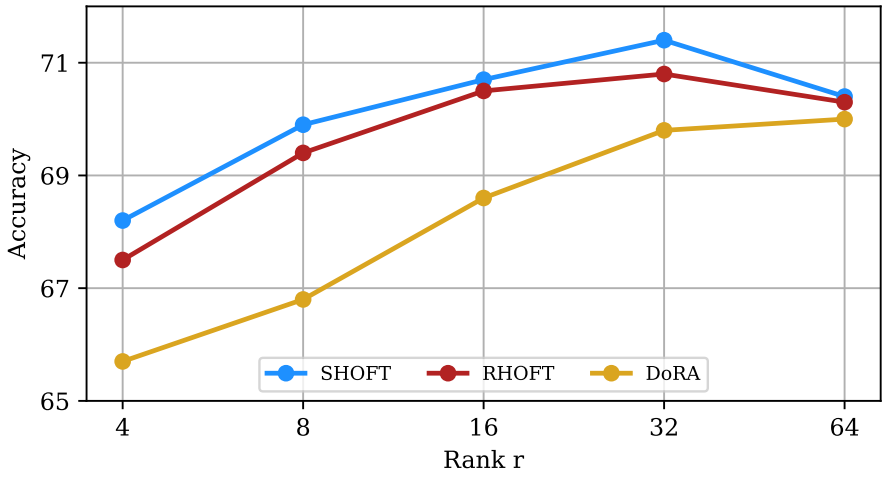

Figure 7.10: Comparative average accuracy on SIQA over an increasing matrix rank for DoRA and RHOFT.

As observed in Figure 7.10, RHOFT follows the same tendency as SHOFT. It gives slightly worse results than SHOFT, as expected.

7.5.5. Experiments on ARC-e

For ARC-e dataset, described in section 3.7, DoRA will be employed to make the comparison, presented in Figure 7.11.

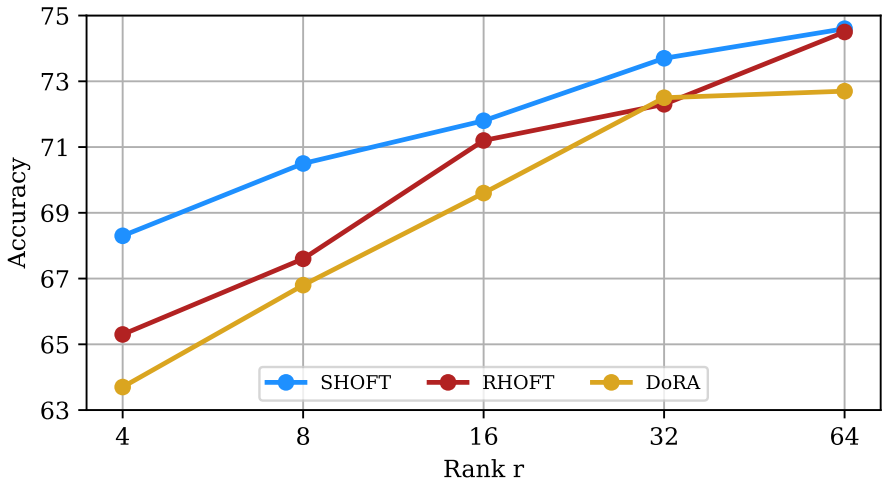

Figure 7.11: Comparative average accuracy on ARC-e over an increasing matrix rank for DoRA and RHOFT.

As observed in Figure 7.11, RHOFT follows the same tendency as SHOFT. It gives slightly worse results than SHOFT, as expected.

7.5.6. Experiments on ARC-c

For ARC-C dataset, described in section 3.7, DoRA will be employed to make the comparison, presented in Figure 7.12.

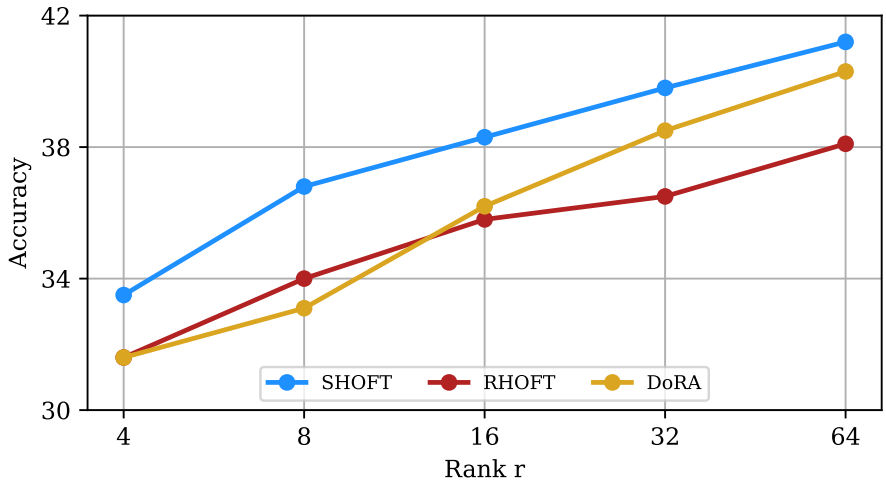

Figure 7.12: Comparative average accuracy on ARC-c over an increasing matrix rank for DoRA and RHOFT.

As observed in Figure 7.12, RHOFT follows the same tendency as SHOFT. It gives worse results than SHOFT, as expected. However, it is curious to see that in this task, which was named as ARC Challenge, RHOFT accuracy difference with respect to SHOFT is less than those reported on ARC Easy task.

7.5.7. Experiments on OBQA

For OBQA dataset, described in section 3.8, DoRA will be employed to make the comparison, presented in Figure 7.13.

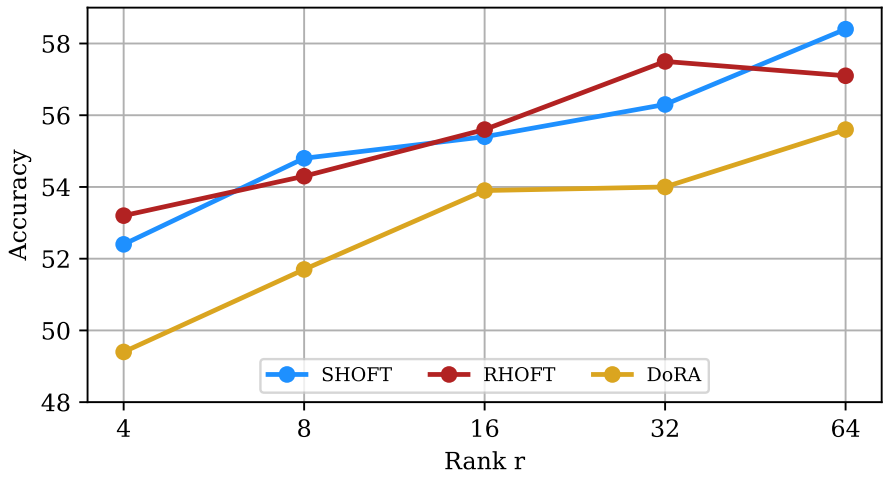

Figure 7.13: Comparative average accuracy on OBQA over an increasing matrix rank for DoRA and RHOFT.

As observed in Figure 7.13, RHOFT follows the same tendency as SHOFT. Nevertheless, RHOFT seems to perform similarly to SHOFT, concluding there is no significant accuracy difference between RHOFT and SHOFT.

7.5.8. Average results

As done in sections 4.6, 5.4 and 6.6, we will also study the comparative average accuracy across commonsense reasoning tasks over an increasing matrix rank for DoRA, SHOFT and RHOFT. This results are presented in Figure 7.14.

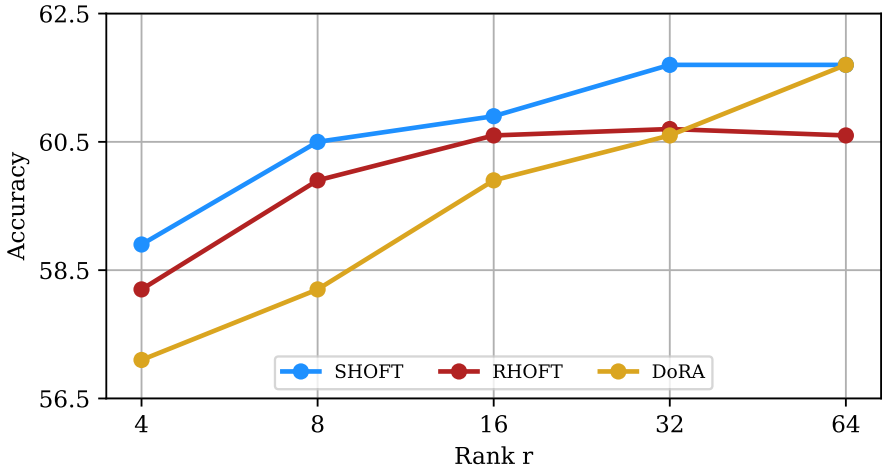

Figure 7.14: Comparative average accuracy across commonsense reasoning tasks over an increasing matrix rank for DoRA, SHOFT and RHOFT.

As observed in Figure 7.14, there's a clear increase tendency. In this case, RHOFT performance is slightly below SHOFT performance. However, RHOFT outperforms DoRA for $r \leq 32$. Thus, RHOFT gives better results than DoRA for lower ranks. Additionally, at $r = 64$, RHOFT accuracy decreases.

7.5.9. Effect of k

In addition to the analysis made in the previous subsections, some works [35] show that fine-tuning the top singular components gives better results in average than full fine-tuning. We will conduct an ablation study on how different k values affects the performance of the fine-tuned model. Considering all matrix targets are 2048×2048 matrices, we will adjust k within the set $k \in \{64, 128, 256, 512, 1024, 2048\}$ and fixing $r_U = r_V = 16$. Results are provided in Figure 7.15.

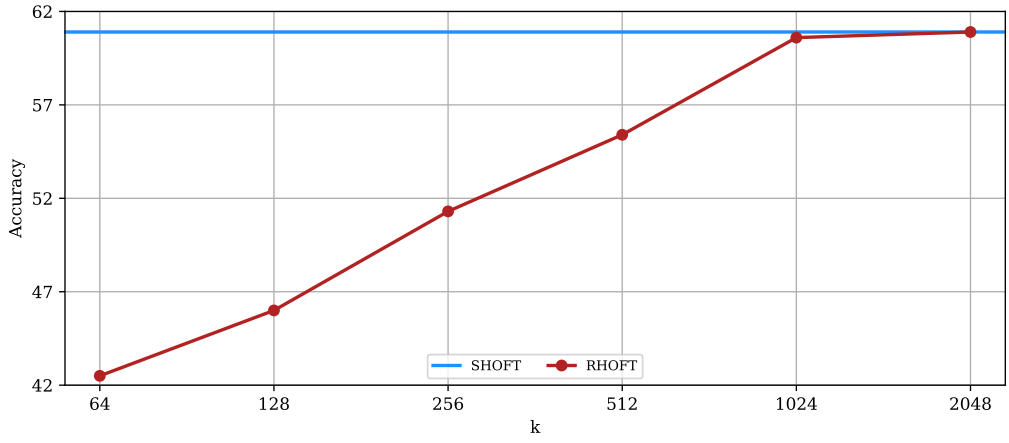

Figure 7.15: Average accuracy on all datasets varying k

As observed in Figure 7.15, lower values of k reduce accuracy consistently. This could be because, although lower dimensions can be removed, they are essential to obtain higher performance. In the case of $k = 1024$, this loss can be ignored.

7.5.10. Speed comparison

This subsection will be devoted to study time performance of all the PEFT methods commented in previous chapters and sections. For this experiment, NVIDIA GeForce RTX 3090 GPUs were employed for avoiding hardware inconvenients.

Table 7.1: Average time on the commonsense reasoning tasks

LoRA	LoKr	DoRA	OFT	BOFT	SHOFT	RHOFT
1.5	1.6	2.0	4.2	5.3	2.7	2.5

Table 7.1 show the different time performance obtained per method. Time performance is measured as the average of the seconds per training step along all datasets commented on Chapter 3. LoRA is the fastest method, followed by its variants. However, SHOFT and RHOFT are time efficient considering they perform multiplicative fine-tuning. Compared to OFT and BOFT, the designed methods give good results. If we sort the methods by speed and take LoRA as the reference we can obtain Table 7.2.

Table 7.2: Time performance on the commonsense reasoning tasks

LoRA	LoKr	DoRA	RHOFT	SHOFT	OFT	BOFT
1.0	1.1	1.3	1.7	1.8	2.8	3.5

From table 7.2 it is clear that both RHOFT and SHOFT need less that the double of time of LoRA, and gives better results than DoRA, as seen before.

7.6 Conclusions

The purpose of this chapter was to redesign SHOFT in order to make it memory efficient. Randomized SVD helps to reduce the amount of non-trainable parameters saved, while discarding the less valuable singular values.

Most experiments performed seem to give the same conclusions: RHOFT performs slightly worse than SHOFT on the tasks selected. This can also be observed in Figure 7.14. This makes sense, since half of the singular values have been discarded in RHOFT in order to make it efficient. This values represented less than 25% of the total sum of singular values, which not supposed a problem for fine-tuning correctly the model.

Additionally, there were other two key experiments. The first one, the effect of varying k, helped to understand how accuracy increases when augmenting k. The second one, the speed comparison along all PEFT methods, gave us as conclusions that SHOFT and RHOFT are time efficient compared to other PEFT methods.

CHAPTER 8
Conclusions

8.1 Summary of work done

In this work we have seen how PEFT methods bring opportunities to LLMs to be correctly fine-tuned without performing a full fine-tuning. All PEFT methods were discussed in detail, exposing important aspects that may be unnoticeable at first sight. It also were designed PEFT methods inspired on aspects of the state-of-the-art PEFT methods. The importance of orthogonal fine-tuning was remarked theoretically and then tested empirically, giving the desired results with SHOFT and RHOFT.

For this work, +800 experiments were performed. This entailed, on average, +1600 hours of computation (approximately 66 days). Additionally, experiments needed to be performed using NVIDIA GeForce RTX 3090 GPUs, since more than 12 GBs of GPU memory were required by each experiment.

After the experimental part, we discussed the performance of SHOFT and RHOFT against DoRA, which is the best state-of-the-art PEFT technique. The conclusions were clear, both methods significantly surpass DoRA in average.

8.2 Objectives achieved

Four main objectives were proposed at the beginning of this work.

The state-of-the-art PEFT techniques where analyzed and described in deep, alongside with its disadvantages and advantages. Theoretical assertions were confirmed by performing the corresponding experiments.

SHOFT and RHOFT were designed considering the good and bad aspects of the PEFT techniques studied and the concepts learned during this degree. These methods gave better results than state-of-the-art PEFT techniques.

Thus, all objectives were accomplished successfully.

8.3 Future work

As future work, there are several aspects that were skipped and may be interesting to explore:

- Perform experiments on bigger LLMs to conclude SHOFT and RHOFT high performance other models.
- Study the effect of adding more trainable parameters by removing non-trainable parameters in RHOFT.
- Perform experiments on other machine learning models and tasks, such as computer vision or machine translation, using SHOFT and RHOFT.

Bibliography

[1] Lingling Xu et al. Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment. *Springer*, 2023.

[2] Christopher M. Bishop. Pattern Recognition and Machine Learning. *Springer*, 2006.

[3] Vaswani et al. Attention Is All You Need. *Advances in Neural Information Processing Systems*. Curran Associates, Inc. 30, 2017.

[4] Gene H. Golub and Charles F. Van Loan. Matrix Computations. *The Johns Hopkins University Press*, 1983.

[5] Kevin Patrick Murphy. Probabilistic Machine Learning: An Introduction *MIT Press*, 2022.

[6] Jesús Urías. Householder factorizations of unitary matrices. *Physics Institute, UASLP. San Luis Potosí, SLP, Mexico.*.

[7] Jindong Wang, Yiqiang Chen. Introduction to Transfer Learning. Algorithms and Practice. *Springer*, 2023.

[8] Tom M. Mitchell Machine Learning McGraw-Hill, New York, 1997

[9] Xiaobai Sun et al. A Basis-Kernel Representation of Orthogonal Matrices. *Argonne National Laboratory*, MCS-P431-0594, 1995.

[10] Humza Naveed et al. A Comprehensive Overview of Large Language Model. *arXiv*, cs.LG, 2024.

[11] Shervin Minaee et al. Large Language Models: A Survey. *arXiv*, cs.LG, 2024.

[12] Jacob Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *North American Chapter of the Association for Computational Linguistics*. 2019.

[13] Radford, Alec and Wu, Jeff and Child, Rewon and Luan, David and Amodei, Dario and Sutskever, Ilya. Language Models are Unsupervised Multitask Learners. 2019.

[14] Tom B. Brown et al. Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*. 2020.

[15] Mike Lewis et al. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *Annual Meeting of the Association for Computational Linguistics*. 2019.

[16] Yuanzhi Li et al. Textbooks Are All You Need II: phi-1.5 technical report. *arXiv*, cs.LG, 2023.

[17] Suriya Gunasekar et al. Textbooks Are All You Need. *arXiv*, cs.LG, 2023.

[18] Erik Nijkamp et al. CodeGen: An Open Large Language Model for Code with Multi-Turn Program Synthesis. *International Conference on Learning Representations*. 2023.

[19] Stephen H. Bach et al. PromptSource: An Integrated Development Environment and Repository for Natural Language Prompts. *Annual Meeting of the Association for Computational Linguistics*. 2022.

[20] Clark et al. BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions. *North American Chapter of the Association for Computational Linguistics* 2019.

[21] Yonatan Bisk et al. PIQA: Reasoning about Physical Commonsense in Natural Language. *AAAI Conference on Artificial Intelligence* 2019.

[22] Maarten Sap et al. SocialIQA: Commonsense Reasoning about Social Interactions. *Association for Computational Linguistics* 2019.

[23] Keisuke Sakaguchi et al. WINOGRANDE: An Adversarial Winograd Schema Challenge at Scale. *arXiv*, cs.LG, 2019.

[24] Peter Clark et al. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *arXiv*, cs.LG, 2018.

[25] Todor Mihaylov et al. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. *Conference on Empirical Methods in Natural Language Processing* 2018.

[26] Lingling Xu et al. LoRA: Low-Rank Adaptation of large language models. *International Conference on Learning Representations*. 2021.

[27] Vlad Fomenko et al. A Note on LoRA. *arXiv*, cs.LG, 2024.

[28] Shih-Ying Yeh et al. Navigating Text-To-Image Customization: From LyCORIS Fine-Tuning to Model Evaluation *arXiv*, cs.LG, 2024.

[29] Ali Edalati et al. KronA: Parameter Efficient Tuning with Kronecker Adapter *arXiv*, cs.LG, 2024.

[30] Shih-Yang Liu et al. DoRA: Weight-Decomposed Low-Rank Adaptation. *arXiv*, cs.LG, 2024.

[31] Zeju Qiu et al. Controlling Text-to-Image Diffusion by Orthogonal Finetuning. *Advances in Neural Information Processing Systems*. 2024.

[32] Weiyang Liu et al. Parameter-Efficient Orthogonal Finetuning via Butterfly Factorization *arXiv*, cs.LG, 2024.

[33] T. Joffrain et al. *Accumulating Householder Transformations, Revisited.*. ACM Transactions on Mathematical Software, Vol. 32, No. 2, 2006.

[34] Halko et al. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2), 217–288, 2011.

[35] Fanxu Meng et al. PiSSA: Principal Singular Values and Singular Vectors Adaptation of Large Language Models. *arXiv*, cs.LG, 2024.

APPENDIX A
Hyperparameters

For performing the experiments described in sections 4.3, 5.4, 6.5 and 7.5, some hyperparameters must be set in order to make fair comparison between the techniques. This appendix is devoted to gather all these hyperparameter configurations. Table A.1 gathers all hyperparameters configurations for training.

Table A.1: Hyperparameter configurations for Phi-1.5 on the commonsense reasoning tasks.

	LoRA	DoRA	OFT	SHOFT	RHOFT
Rank r	16	16	64	16, 16	16, 16
α	32	32	32	-	-
Dropout	0.05	0.05	0.05	0.05	0.05
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW
LR	$2e^{-4}$	$2e^{-4}$	$2e^{-4}$	$2e^{-4}$	$2e^{-4}$
LR Scheduler	Linear	Linear	Linear	Linear	Linear
Batch size	16	16	16	16	16
Warmup Steps	100	100	100	100	100
Epochs	3	3	3	3	3
Targets	Q, K, V	Q, K, V	Q, K, V	Q, K, V	Q, K, V

The hyperparameter α is not used in SHOFT and RHOFT. Additionally, SHOFT and RHOFT have two values in Rank row corresponding to r_U, r_V respectively.

ANEXO

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**				x
ODS 2. **Hambre cero.**				x
ODS 3. **Salud y bienestar.**				x
ODS 4. **Educación de calidad.**				x
ODS 5. **Igualdad de género.**				x
ODS 6. **Agua limpia y saneamiento.**		x		
ODS 7. **Energía asequible y no contaminante.**	x			
ODS 8. **Trabajo decente y crecimiento económico.**			x	
ODS 9. **Industria, innovación e infraestructuras.**	x			
ODS 10. **Reducción de las desigualdades**.	x			
ODS 11. **Ciudades y comunidades sostenibles**.		x		
ODS 12. **Producción y consumo responsables.**				x
ODS 13. **Acción por el clima.**				x
ODS 14. **Vida submarina.**				x
ODS 15. **Vida de ecosistemas terrestres.**				x
ODS 16. **Paz, justicia e instituciones sólidas.**				x
ODS 17. **Alianzas para lograr objetivos.**				x

Reflection on the Relationship of the Final Degree/Master's Project with the SDGs and the Most Related SDGs:

Pretrained models play a crucial role in modern AI, enabling the rapid deployment of sophisticated solutions across a wide array of fields. These models, already trained on vast amounts of data, can be fine-tuned for specific tasks with minimal additional training, making them highly efficient and versatile. This work aims to delve into the time and performance benefits of PEFT techniques. PEFT techniques are crafted to minimize the computational power and resources required for fine-tuning, which is essential for making AI more sustainable and accessible. This aligns directly with Sustainable Development Goals (SDGs) 6 and 7, which focus on clean water and sanitation, and affordable and clean energy, respectively. By reducing the electricity and water consumption needed for cooling large data centers, PEFT techniques offer an environmentally friendly solution that conserves energy and water.

Moreover, the advanced PEFT techniques such as SHOFT and RHOFT are linked to SDG 9, which emphasizes industry, innovation, and infrastructure. These techniques are built upon the latest PEFT methods, enhancing existing industry standards and driving forward innovation in the fine-tuning process. SHOFT and RHOFT improve efficiency by selectively tuning only specific parts of the model, which significantly reduces the computational load and speeds up the fine-tuning process. By enhancing these techniques, PEFT methods make AI development more efficient and sustainable, ensuring that industry practices keep pace with technological advancements and contribute to building robust and innovative infrastructures.

PEFT techniques are also instrumental in reducing inequalities, aligning with SDG 10, which aims to reduce inequality within and among countries. By making advanced AI more accessible and affordable, especially in resource-constrained environments like developing countries, PEFT significantly lowers costs of producing fine-tuned models. This accessibility allows organizations with limited budgets to deploy AI in critical sectors such as education, healthcare, and economic development. For instance, AI can be used to improve educational resources, provide better healthcare diagnostics, and support local businesses, thereby helping to bridge social and economic disparities. By democratizing access to advanced AI, PEFT techniques play a vital role in fostering inclusive growth and reducing inequality.

Furthermore, PEFT techniques contribute to the achievement of SDG 11, which focuses on sustainable cities and communities. By enabling the development of AI models for smart and sustainable cities without demanding extensive computational resources, PEFT techniques make it possible to implement advanced AI solutions in urban planning and management. This capability can significantly enhance urban planning by providing better data analysis and predictive insights, optimize resource management by improving the efficiency of utilities and services, and improve disaster response systems by enabling faster and more accurate emergency responses. These improvements lead to cities that are more resilient, inclusive, and sustainable, ensuring better quality of life for their inhabitants.

Lastly, the efficiency of PEFT techniques reduces the barrier to entry for small businesses and developing economies to utilize advanced AI, fostering innovation and productivity across various sectors. This is closely related to SDG 8, which promotes sustained, inclusive, and sustainable economic growth, full and productive employment, and decent work for all. As AI becomes more accessible and scalable, it stimulates economic growth by enhancing operational efficiencies, enabling the creation of new business models, and facilitating more informed decision-making processes. Small businesses can use AI to optimize their operations, improve customer service, and develop innovative products and services, driving economic expansion. This widespread access to AI can substantially accelerate economic growth and sustainable development on a global scale, ensuring that the benefits of AI technology are widely distributed and contribute to overall societal advancement. Through these multifaceted impacts, PEFT techniques not only advance the field of AI but also contribute to a more equitable, sustainable, and prosperous world.