

https://dspace.org/

ParameterEfficient FineTuning of Pretrained Large Language Models for Financial Text Analysis

Kelly Langa; Hairong Bau; Olaperi Okuboyejo

https://hdl.handle.net/10539/47589
SCOPUS

Downloaded from a DSpace repository. https://dspace.org/



Parameter-Efficient Fine-Tuning of Pre-trained Large Language Models for Financial Text Analysis

Kelly Langa(✉), Hairong Wang, and Olaperi Okuboyejo

University of the Witwatersrand, Johannesburg 2017, South Africa
kellylanga2@gmail.com

Abstract. Recent advancements in natural language processing (NLP) have been driven by large language models (LLMs) that excel in understanding the complexities of natural language. These models have transformed NLP tasks through transfer learning, where pre-trained LLMs are fine-tuned on domain-specific datasets. Financial sentiment analysis is particularly challenging due to the complexity of financial language, requiring more advanced methods than traditional sentiment analysis approaches. Fine-tuning LLMs can enhance performance in the financial domain, but the high computational cost of standard full fine-tuning is a barrier. This study explores the effectiveness of four (4) parameter-efficient fine-tuning (PEFT) methods, namely, Low-Rank Adaptation (LoRA), prompt tuning, prefix tuning, and adapters, for financial sentiment analysis. The findings show that PEFT methods can match or exceed the performance of full fine-tuning while significantly reducing computational requirements. Specifically, adapting the Open Pre-trained Transformers (OPT) model with LoRA achieved the highest accuracy of 89% using only 0.19% of the model's parameters. PEFT methods also resulted in substantial graphics processing unit (GPU) memory savings of up to 80%. Small-scale fine-tuned LLMs outperformed cutting-edge large-scale general-purpose models like ChatGPT, highlighting the value of domain-specific fine-tuning. LLMs demonstrated superiority over conventional long short-term memory (LSTM) models by achieving a 18% increase in accuracy, thereby validating their higher implementation costs.

Keywords: Natural language processing · Large language models · Parameter efficient fine tuning · Transfer learning · Financial sentiment analysis

1 Introduction

The field of natural language processing (NLP) has experienced remarkable advancements in recent years, largely attributed to the emergence of large language models (LLMs) that have attained cutting-edge performance on a wide range of language-related tasks. LLMs possess the ability to capture the intricate semantic and syntactic complexities of natural language, making them highly suitable

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2025
A. Gerber et al. (Eds.): SACAIR 2024, CCIS 2326, pp. 3–20, 2025.
https://doi.org/10.1007/978-3-031-78255-8_1

for tasks that require language understanding and text generation [4]. Given their centrality in language tasks, there has been a fundamental shift in NLP methodologies, moving away from specialized custom models trained from scratch towards building on top of general-purpose base models. With the increasing availability of open-source pre-trained general-purpose base LLMs, transfer learning has emerged as a dominant paradigm to specialize (fine-tune) pre-trained models for specific down-stream tasks using a domain specific dataset.

The current state-of-the-art LLMs such as GPT-4 and Gemini Ultra are generalists in that they have been pre-trained on a diverse large corpus of text. These LLMs perform poorly out-of-the-box (zero-shot inference) on knowledge-intensive tasks as these tasks require domain specific expertise and go beyond syntax analysis or simple pattern recognition [17]. One such task is financial sentiment analysis, which involves predicting the sentiment polarity of finance textual data, such as earnings calls and social media posts. Financial sentiment analysis poses unique challenges due to the complex and domain-specific nature of financial data, characterized by technical jargon and acronyms [21]. Traditional approaches to sentiment analysis using rule-based or machine learning techniques have limitations in accurately capturing the nuances of financial language, resulting in suboptimal performance [6].

Fine-tuning base LLMs for financial sentiment analysis holds promise for improving modelling accuracy in this domain. Fine-tuned LLMs have shown promising results in financial sentiment analysis due to their ability to learn comprehensive language representations that capture the semantic and syntactic relationships within words and phrases [28]. Full parameter fine-tuning, which encompasses fine-tuning all the layers of a pre-trained LLM, remains the most effective method for adapting a model to a new domain and task in terms of modelling performance [14]. However, full fine-tuning is computationally expensive and time-consuming for training large-scale language models, making it almost infeasible in academia and industry. In recent years, researchers have been actively exploring light-weight adaptation methods that allow fine-tuning using a small subset of parameters while maintaining high modelling performance [8]. Parameter-efficient fine-tuning (PEFT) techniques are lightweight adaptation methods which involve adding a small set of weights to a pre-trained LLM and only fine-tuning the newly added weights, while keeping the pre-trained LLM parameters fixed.

In this paper, we explore the potential of adapting LLMs, considering both modelling performance and training efficiency, for sentiment analysis of financial data using PEFT methods. By harnessing the capabilities of LLMs and optimizing the fine-tuning process, the study seeks to improve the efficiency and accuracy of sentiment analysis in the financial domain, enabling the democratization of artificial intelligence and better data-driven decision-making. As such, the main contributions of this paper can be summarised as follows:

1. We developed resource-efficient models leveraging LLMs and PEFT methods achieving state-of-the-art performance in financial sentiment analysis.

2. We extend on existing work by conducting a comprehensive comparative study into the fine-tuning of different open-source LLMs employing different PEFT methods and their various configurations.
3. We show that PEFT methods lead to significant savings in peak GPU memory consumption, conserving up to 80% of GPU memory.
4. We show that PEFT methods demonstrate high modularity by utilizing merely a fraction of the model's total parameters, allowing for the creation of compact checkpoints compared to full fine-tuning.

The rest of the paper is organized into five sections. Section 2 provides a background of the relevant topics, including large language models, the transformer model architecture, transfer learning and model adaptation. Section 3 outlines the experimental procedure. Section 4 presents and discusses the study's findings. Section 5 concludes the paper and suggests future research directions.

2 Background

Transfer learning is among the most impactful concepts in deep learning and refers to the process of leveraging knowledge acquired from pre-training on a generic large-scale dataset to initialise a model for a target task [28]. In the realm of NLP, pre-trained LLMs capture general language patterns and semantic representation that possess transferability across diverse downstream tasks [19]. Leveraging a pre-trained model weights as a starting point to building a customized model has proven to be superior in terms of modelling performance and modelling efficiency as compared to building a model from scratch [4,19].

Fine-tuning is often employed to facilitate efficient transfer learning from pre-trained models to specific tasks. It involves taking a pre-trained model and adapting it to a specific task by further training it on a task-specific dataset. Full fine-tuning of a pre-trained LLM is the gold standard approach for adapting a model to a new target task [13,14]. With full parameter fine-tuning, all layers of the model are updated. However, it is computationally expensive and time consuming to update all weights of LLMs.

2.1 Parameter-Efficient Fine-Tuning

In recent times, substantial advancements have been made by researchers in the development of techniques for fine-tuning LLMs in a computationally efficient manner without sacrificing on modelling performance. These techniques are known as PEFT. PEFT methods enable efficient adaption of pre-trained LLMs to downstream tasks by adding a small set of weights to the pretrained LLM and only fine-tune the newly added weights while the weights of the pre-trained LLM remain frozen [8,13]. This approach results in major savings in computational and storage costs, while being able to maintain performance levels that are comparable to those achieved through full fine-tuning [3,8].

PEFT techniques were developed based on the observation that earlier layers in pre-trained LLMs learn generic linguistic patterns that are applicable across

various tasks while later layers tend to capture more task-specific patterns [18]. The later layers therefore undergo significant changes while earlier layers remain relatively stable during full fine-tuning [25]. The most notable PEFT methods include prompt tuning, prefix tuning, adapter tuning and low-rank adaptation (LoRA). Further elaboration on these techniques is provided in the subsequent subsections.

Prompt Tuning. Prompt tuning represents an innovative approach for fine-tuning LLMs for specific tasks, leveraging the concept of soft prompts which are learnable tensors that are concatenated with the input embeddings. Specifically, prompt tuning inserts trainable vectors at the beginning of the input sequence only at the embedding layer, without altering the model's deeper layers. Unlike traditional prompts that involve prepending text with specific instructions or examples, soft prompts are not directly interpretable by humans since they do not correspond to real-world tokens but rather serve as virtual tokens optimized during training [15]. This method preserves the original model parameters, focusing updates exclusively on the soft prompt embeddings through backpropagation, which makes it a more parameter-efficient alternative to full model fine-tuning.

Despite its efficiency prompt tuning is not without challenges. It tends to converge slower than other fine-tuning methods, which can make optimization more difficult in practice [15]. However, its ability to improve language model performance on natural language understanding tasks, combined with the computational and memory efficiencies it offers, makes it a promising tool in the evolving landscape of model adaptation.

Prefix Tuning. Introduced by Li and Liang in 2021 [16], this technique innovates by appending task-specific trainable vectors, referred to as prefixes, not only to the input layer but across all layers of the transformer model [16]. These prefixes are integrated into the key and value matrices of the model's self-attention mechanism, providing a task-relevant context that guides the model towards generating desired outputs [16]. Distinctively, prefix tuning diverges from direct manipulation of soft prompts. It employs a separate bottleneck feed-forward neural network (FFN) to optimize these prefix parameters, thereby circumventing the training instabilities and performance degradation associated with direct training through the reparameterizion of the prefix vectors [8]. After the optimization process, this FFN can be discarded, leaving the optimized prefixes to influence the model's behavior.

The length of the prefix plays a critical role in the tuning process, with empirical evidence suggesting that there exists an optimal prefix length beyond which performance may plateau or slightly decline [16]. This balance underscores the nuanced relationship between the amount of introduced task-specific information and model performance. A drawback of prefix tuning, however, is that its optimization process can be complex, necessitating experimentation with various parameters to achieve the best results [23]. Moreover, the introduction of

prefixes reduces the available sequence length for actual input data, potentially limiting the model's capacity to handle longer inputs.

Adapter Tuning. Adapter tuning has emerged as an innovative and lightweight approach to customizing pre-trained networks for specific downstream tasks without the need for comprehensive retraining of the entire model. By integrating small, fully connected networks known as adapters between the sub-layers of transformer models as shown in Fig. 1, this method achieves notable performance enhancements while tuning a small percentage of the total model weights [12]. The essence of adapter tuning lies in its ability to introduce architectural modifications that repurpose a pre-trained network, focusing solely on the new parameters introduced by the adapters while keeping the original LLM parameters frozen [10]. This strategy not only preserves the model's acquired knowledge but also ensures a minimal addition of parameters, making it a highly memory-efficient solution.

The core architecture of these adapters incorporates a bottleneck design, which significantly reduces the dimensionality of the input features before applying a non-linearity and projecting them back to their original dimension. This process, coupled with an internal skip-connection within each adapter module, allows for an effective balance between performance and parameter efficiency [23]. The skip-connection plays a crucial role, ensuring that the adapters can function as an approximate identity when their parameters are near-zero at initialization, thus preserving the integrity of the original input while introducing task-specific adaptation [12]. Originating from the pioneering work of Houlsby et al. [10], the adapter tuning method has been refined to include not just the bottleneck structure but also specific layer normalization parameters trained for each task. The Houlsby adapter configuration places two adapters in the sublayers of the transformer as shown in Fig. 1.

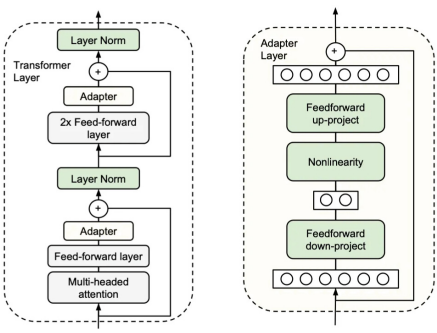


Fig. 1. Adapters incorporated in the transformer architecture using the Houlsby adapters configuration [10]

Low-Rank Adaptation. Developed by Hu et al. [11], LoRA fundamentally changes the approach to model adaptation by injecting trainable low-rank decomposition matrices into the layers of a pre-trained model [1]. Figure 2 illustrates a schematic representation of LoRA, showcasing the self-attention layer LoRA configuration which adds LoRA matrices to the self-attention sub-layer. This method is predicated on the insight that pre-trained models, despite their size, operate within a low intrinsic dimension, meaning that their complexity can be effectively captured using matrices of much lower rank than initially presumed [1]. At the heart of LoRA is the concept of matrix rank, which signifies the maximum number of linearly independent rows or columns in a matrix. This concept is crucial because it informs us about the amount of unique information or the dimensionality of the space spanned by the matrix [30]. LoRA leverages this by approximating the weight adjustments needed during fine-tuning as the product of two smaller matrices (W_A and W_B), significantly reducing the number of parameters required for effective fine-tuning. The rank r becomes a critical hyperparameter in this setup, dictating the degree of approximation and the balance between model complexity and training efficiency [7].

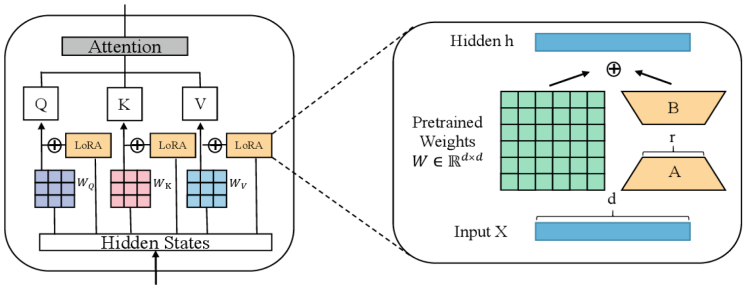


Fig. 2. Low-rank adaptation incorporated in the transformer architecture using the self-attention layer LoRA configuration [27]

3 The Proposed Method

This section outlines the methodologies employed in our study to fine-tune LLMs using PEFT techniques. We detail the steps from dataset preprocessing and splitting, through the integration of PEFT methods in model adaptation, to the benchmarking against established adaptation methods. The section also describes the evaluation metrics used to assess the efficiency and performance of the fine-tuned models, and the iterative process of analysis and hyperparameter tuning to optimize results.

3.1 Data and Data Preprocessing

Data Description. Large language models were fine-tuned using the financial phrasebank dataset [22] for a sentiment analysis task, aiming to thoroughly assess the encoded knowledge within the LLMs. This dataset consists of sentences from financial news categorised by sentiment. The dataset serves as a benchmark to assess the effectiveness of sentiment models specifically designed for economic texts. The dataset consists of 4846 sentences labelled as either negative, positive or neutral sentiment. The class distribution of the neutral, negative and positive labels are 59%, 28% and 13% respectively.

Tokenization and Encoding. The text in the dataset was tokenized and encoded into numerical representations that can be understood by the model. Pre-trained LLMs have their own built-in-tokenizers specifically designed to work with the model's vocabulary and architecture [9]. The tokenizers of the employed LLMs are based on the byte-pair encoding (BPE) and SentencePiece tokenizers. These tokens were then encoded into unique integers based on the model's vocabulary.

Batch Processing and Padding. The sequence length of the sentences in the dataset varies, necessitating uniform input sequences to enable parallel processing. To optimize memory usage and computational efficiency, batch processing and padding were employed. A batch size of 8 was utilized along with dynamic padding. With dynamic padding the sequence length of a batch is determined by the longest sequence in the batch. Sequences shorter than the designated length will be padded with special tokens.

Attention Mask. Attention masks were added to the padded sequences to guide the model's attention mechanism, helping to identify the actual content of each sequence while distinguishing it from the padding tokens used to equalize the sequence lengths.

Sentiment Label Encoding. The financial phrasebank dataset has categorical sentiment labels. The categorical labels were converted into unique integers in line with the vocabulary of the model.

3.2 Methods

Training, Validation and Test Split. The dataset was split into 70% training set, 15% validation set, and 15% testing set.

Pre-trained Models. We conducted a comparative analysis of LLMs built on three distinct transformer architectures: encoder-only, encoder-decoder, and decoder-only, due to the limited existing literature on their parameter-efficient adaptation strategies. The selection of LLMs was based on several criteria, including performance on benchmarks such as the Massive Multitask Language Understanding (MMLU) and General Language Understanding Evaluation (GLUE), open-source licensing allowing commercial use, fine-tunability, model size, and out-of-the-box quality. The LLMs employed are Robustly Optimized BERT approach (RoBERTa) [20], Fine-tuned LAnguage Net Text-To-Text Transfer Transformer (FLAN-T5) [5] and Open Pre-trained Transformers (OPT) [29]. We used smaller versions of the LLMs taking into account our limited computational resources. Table 1 gives a summary of the LLMs that were employed, showing their model size, architecture, training objective, vocabulary size, context length and dataset size used for pre-training. The capacity of an LLM is a dynamic interplay between these factors. The models fall within a similar scale range, and their pre-training utilised a diverse generic dataset consisting of billions of tokens making them general-purpose models. The pre-training strategies used are the Masked Language Modelling (MLM) and Autoregressive Language Modelling (ALM).

Table 1. Comparative overview of hyperparameters utilized in the pre-training of RoBERTa-large, FLAN-T5-base and OPT-350 models

	RoBERTa-large	FLAN-T5-base	OPT-350
Model size	355M	248M	350M
Model Architecture	encoder-only	encoder-decoder	decoder-only
Pre-training strategy	MLM	MLM	ALM
Pre-training data size	160B tokens	34B tokens	180B tokens
Vocabulary size	50K	32K	50K
Context length	512	512	2048

Incorporating PEFT Techniques in Fine-Tuning LLMs. To date, there still exists a void in comprehensive investigation regarding the impact of various efficient tuning methods on LLMs pre-trained under diverse settings. We employed four PEFT methods, each introducing distinct modifications to the transformer architecture. The PEFT methods are prompt tuning, prefix tuning, adapter tuning and low-rank adaptation (LoRA). We further explored various configurations of these methods as summarized in Table 2.

Model Training. The aforementioned parameter-efficient fine-tuning techniques were employed to fine-tune the LLMs. The models were fine-tuned by adjusting the extra added PEFT parameters using backpropagation and gradient decent while the base model was kept frozen. Table 3 summarises the

Table 2. PEFT methods and their configurational variants

PEFT Method	Configuration
Prompt Tuning	Initializing prompt with own text
	Random prompt initialization
	Initializing prompt with LLM tokens
Prefix Tuning	No reparameterization
	Reparameterization
Adapter	Pfeiffer adapter [26]
	Houlsby adapter [10]
	Parallel adapter [12]
LoRA	Self-attention layer LoRA
	Feed-forward layer intermediate weights LoRA
	Feed-forward layer output weights LoRA

hyperparameters that were utilized to fine-tune the OPT model with LoRA. Prompt tuning and prefix tuning utilize the number of virtual tokens hyperparameter, adapters employ the bottleneck dimension hyperparameter, and LoRA utilizes the LoRA rank, LoRA alpha, and LoRA dropout hyperparameters. The LoRA alpha and dropout were set at 32 and 0.1, respectively, in line with the recommendations from the original LoRA paper [11]. To identify the optimal hyperparameter configurations for training the LLMs, we utilized the HyperOpt Tree of Parzen Estimators (TPE) search algorithm in conjunction with an Asynchronous Successive Halving (ASHA) trial scheduler. Fractional GPU utilization was employed to concurrently execute multiple trials on a single GPU, thereby conserving resources.

Table 3. Hyperparameters used for fine-tuning OPT model with LoRA

Hyperparameter	OPT model		
	Low-rank adaptation		
	Attention weights	FFN intermediate weights	FFN output weights
LoRA rank	18	5	41
LoRA dropout	0.1	0.1	0.1
LoRA alpha	32	32	32
Batch size	8	8	8
Epochs	3	3	4
Learning rate (LR)	0.0001227	0.00008069	0.0001569
LR Scheduler	warmup LR	warmup LR	warmup LR
Warmup ratio	0.03560	0.03778	0.08630
Optimizer	AdamW	AdamW	AdamW
Loss	cross-entropy	cross-entropy	cross-entropy
Dropout	0.1	0.1	0.1

Hardware, Libraries and Software. Model training and analysis were conducted in Python 3 on Google Colab, utilizing an Intel Xeon CPU (13 GB RAM) and NVIDIA Tesla K80 GPU (12 GB VRAM). Open-source libraries, transformers and PyTorch were used for model implementation, with scikit-learn for performance evaluation and pandas for data manipulation. Visualizations were created with seaborn and matplotlib. PEFT methods were implemented using Peft, opendelta, and adapter-transformers libraries. Hyperparameter tuning was performed with Ray Tune, and artifacts were logged using the weights and biases library.

3.3 Analysis

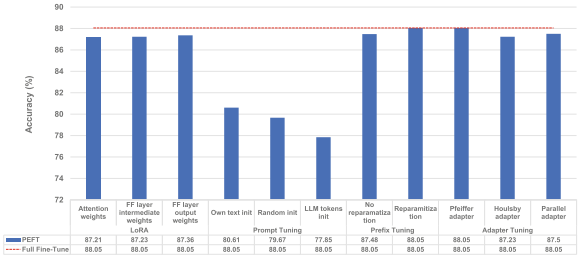
Benchmark. To evaluate the effectiveness of fine-tuning LLMs using PEFT techniques, we established a benchmark and conducted tests using established fine-tuning methods and traditional NLP algorithms, as well as presenting outcomes obtained from previous studies. The methods included full fine-tuning (baseline), model head fine-tuning (adapting only the last layer), and zero-shot inference without fine-tuning. Additionally, we referenced results from prior studies employing traditional NLP algorithms (LSTM) and state-of-the-art general-purpose LLMs (GPT-4 and ChatGPT) to justify the use of LLMs and the associated fine-tuning costs.

Evaluation Metric. In evaluating the modelling performance of the models, accuracy scores were computed. In evaluating the computational efficiency of training the models, the following evaluation metrics were used: number of trainable parameters, storage required to save the adaptation parameters, training time, inference latency and peak GPU memory usage.

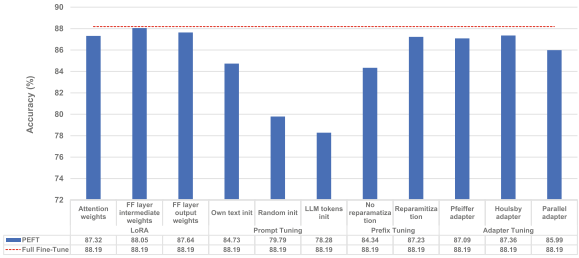
4 Results and Discussion

4.1 Knowledge Transfer Efficacy of LLMs Through PEFT Techniques

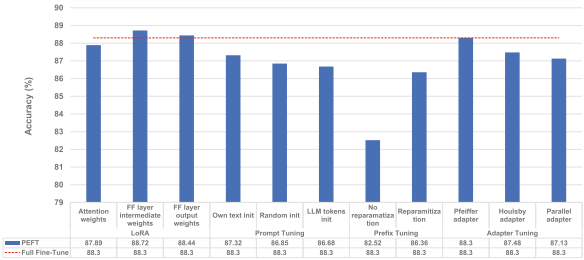
Figure 3 illustrates the modeling performance of the employed PEFT methods and their various configurations in terms of accuracy across three different models, with full fine-tuning serving as the baseline for comparison. Evaluation of each PEFT method's performance on RoBERTa, FLAN-T5 and OPT models revealed consistent performance, ranging from a minimum accuracy of 78% to a maximum of 89%. This suggests the effectiveness of PEFT methods in transferring knowledge during the fine-tuning process of LLMs pre-trained using different strategies. Overall, the OPT model showed the highest average accuracy when fine-tuned using the different PEFT methods, followed by RoBERTa and then FLAN-T5. The observed trend suggests that the LLM's capacity and its ability to transfer knowledge are primarily determined by its model size and the scale of the pre-training data, with the pre-training strategy having a less significant impact in terms of architecture.



(a) RoBERTa model



(b) FLAN-T5 model

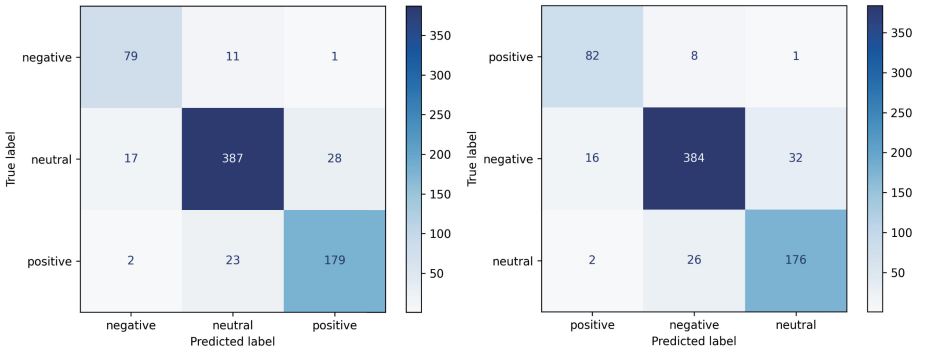


(c) OPT model

Fig. 3. Modelling performance of PEFT methods and their various configurations with full fine-tuning as the baseline

4.2 Comparing the Modeling Performance of PEFT Methods

LoRA emerged as the top-performing PEFT method in terms of modeling accuracy. It surpassed full fine-tuning when applied to the OPT model, specifically in configurations using feed-forward layer intermediate weights and feed-forward layer output weights. The feed-forward layer intermediate weights LoRA refers to the application of the LoRA technique to the intermediate weights within the feed-forward layer, while the feed-forward layer output weights LoRA applies LoRA to the output weights of the feed-forward layer. Drawing from our review of existing literature, LoRA is almost always applied using the self-attention weights LoRA configuration which was proposed in the original LoRA paper [11]. However, our findings strongly advocate for its utilization using the feed-forward layer intermediate weights. This deviation from the conventional approach aligns



(a) LoRA (b) Adapter tuning

Fig. 4. Confusion matrix generated for LoRA and adapter tuning when applied to the OPT model

with the common understanding in deep learning that early layers capture generic linguistic patterns, while later layers specialize in task-specific features. Indeed, both feed-forward layer intermediate weights and feed-forward layer output weights LoRA configurations outperform the self-attention weights configuration. Both these methods inject LoRA modules in the later layers closer to the model head.

Adapter tuning emerged as the second-best performing method, matching the performance of full fine-tuning when utilizing the Pfeiffer adapter configuration on both the RoBERTa and OPT models. Notably, the Pfeiffer adapter configuration outperforms the pioneering Houlsby adapter configuration, which originally introduced the concept of adapters. Unlike the Houlsby adapter, which inserts an adapter module after both the self-attention layer and the feed-forward layer, the Pfeiffer adapter inserts the module solely after the feed-forward layer. Injecting adapter modules solely at the higher layers yields superior results compared to distributing them across both early and top layers, which can potentially lead to diminished performance. The results further emphasize the intuition that lower layers capture lower general features that are shared among tasks, while the later layers develop task-specific features. The parallel adapter configuration falls short of both the Houlsby and Pfeiffer adapter configurations in terms of modelling performance, highlighting the advantage of applying adapters in a serial manner.

Prefix tuning secured the third position, achieving the same performance as full fine-tuning when employing the reparameterization prefix tuning configuration on the RoBERTa model. Prefix-tuning prepends task-specific trainable prefix vectors to the key matrix and value matrix in the self-attention layer. Considering that the self-attention layer is the foundational element of the transformer architecture, which provides it with the capability to capture intricate relationships and nuances within data, the incorporation of prefix modules in the self-attention layer's matrix weights proves to be an efficient strategy for

enhancing LLMs. It is more effective to carry out the optimization of prefix parameters through an independent feed-forward network (reparameterization), rather than directly adjusting the soft prompts, to avoid instability and performance degradation.

The performance of prompt tuning fell behind that of the other PEFT methods. Notably, initializing the prompt with a custom-designed prompt yielded better results compared to other prompt tuning configurations. The subpar performance of prompt tuning was expected, as this method introduces no alterations to the internal structure of the transformer but merely appends tokens to the input sequence. These additional tokens solely guide the model in generating the output. Consequently, prompt tuning with random prompt initialization and initialization using tokens from the model's vocabulary proved ineffective in steering the model towards generating the desired output. Both approaches seemed to cause confusion within the model, resulting in poorer performance.

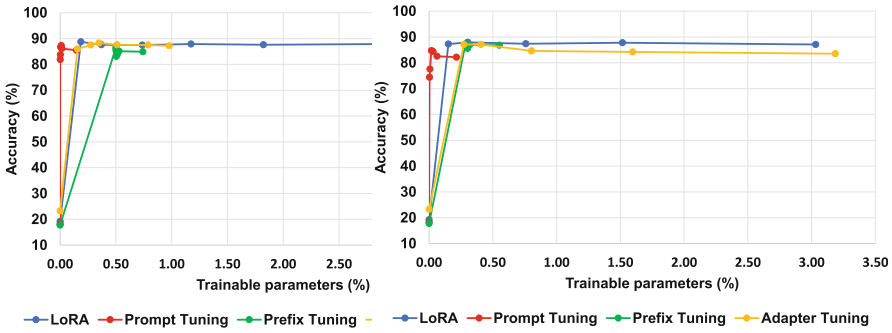
Notably, in handling imbalance inherent in our dataset, our models demonstrate remarkable capability in classifying each class effectively without exhibiting bias towards any specific class. This equitable performance is further elucidated through the detailed examination provided by the confusion matrix showcased in Fig. 4. In the realm of finance, where precision and recall play pivotal roles in decision making processes concerning investments, striking a balance between these two metrics is paramount.

4.3 Number of Trainable Parameters

Table 4 illustrates the characteristics of each PEFT method, including trainable parameters, storage size, and training time, focusing on the best performing configuration for each PEFT method, with full fine-tuning serving as the baseline. Across the board, the trainable parameters introduced by our PEFT methods represent a fraction ranging from 0.01% to 1.06% of the model's total parameters. Notably, LoRA applied to the OPT model excels, outperforming full fine-tuning while utilizing 0.19% of the model's total parameters. Interestingly, the relationship between the performance of PEFT methods and the number of trainable parameters is not linear. Figure 5 elucidates how performance tends to improve with an increase in trainable parameters up to a certain threshold, after which a slight decline is observed. This phenomenon underscores the critical role of the PEFT method's architecture and the placement of PEFT modules in determining its efficacy. Notably, some PEFT methods outperform full fine-tuning due to their ability to mitigate catastrophic forgetting, a phenomenon observed during full fine-tuning of LLMs, where the model inadvertently forgets previously acquired knowledge from pre-training, leading to overfitting on small fine-tuning datasets. Given the minimal number of trainable parameters, PEFT methods offer notable benefits in terms of portability and modularity. They enable the creation of compact checkpoints, typically just a few megabytes in size, compared to the bulky checkpoints generated through full fine-tuning.

Table 4. Comparison of trainable parameters and training time for different fine-tuning methods

Model	Method	Trainable parameters	Trainable parameters size (MB)	Training time (s)
RoBERTa	Full fine-tune	355.36M (100%)	1421.45 (100%)	458
	LoRA (FFN intermediate weights)	3.33M (0.93%)	13.34 (0.94%)	303
	Prompt tuning (Own text init)	2.11M (0.59%)	8.44 (0.59%)	779
	Prefix tuning (Reparamitization)	3.78M (1.06%)	15.12 (1.06%)	363
	Adapter tuning (Pfeiffer)	3.59M (1.00%)	14.34 (1.01%)	229
Flan-T5	Full fine-tune	296.93M (100%)	990.31 (100%)	289
	LoRA (FFN intermediate weights)	67.58M (0.30%)	101.4 (10.24%)	470
	Prompt tuning (Own text init)	46.08K (0.01%)	0.18 (0.02%)	207
	Prefix tuning (Reparamitization)	0.64M (0.002%)	101.24 (10.22%)	340
	Adapter tuning (Pfeiffer)	0.90M (0.41%)	102.31 (10.33%)	389
OPT	Full fine-tune	331.20M (100%)	1324.79 (100%)	477
	LoRA (FFN intermediate weights)	0.62K (0.19%)	2.46 (0.19%)	206
	Prompt tuning (Own text init)	32.26K (0.01%)	0.13 (0.01%)	1886
	Prefix tuning (Reparamitization)	1.66M (0.50%)	6.65 (0.50%)	600
	Adapter tuning (Pfeiffer)	1.16M (0.35%)	4.63 (0.35%)	291



(a) OPT model (b) FLAN-T5 model

Fig. 5. Performance of PEFT methods with number of trainable parameters

4.4 Computational Efficiency

Figure 6 depicts the peak GPU memory consumption during the fine-tuning of LLMs using both PEFT methods and full fine-tuning. Notably, PEFT methods demonstrate significant savings, conserving up to 80% of GPU memory when fine-tuning the RoBERTa model. These savings are primarily achieved by reducing the need for gradient computation for most parameters, as they are held fixed during fine-tuning.

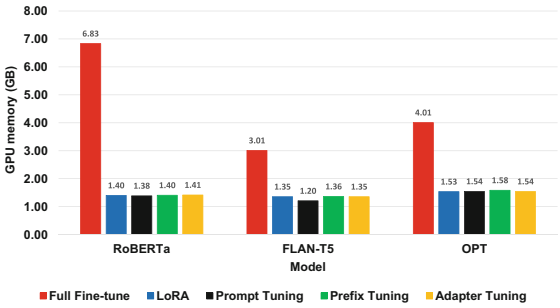


Fig. 6. Peak GPU memory consumed during fine-tuning

4.5 Inference Latency

Network latency is a crucial consideration when deploying deep networks in real-world scenarios, where rapid inference times are paramount, often ranging from milliseconds to seconds. In assessing network latency, the focus is primarily on measuring the feed-forward operation of the network. Figure 7 illustrates the inference latency of the PEFT methods for the utilized LLMs, with full fine-tuning serving as the baseline. PEFT methods introduce additional trainable parameters to the base model, potentially resulting in inference latency due to increased data flow paths. However, the results indicate that there is no statistically significant difference in the inference latency between the various techniques. Given their negligible impact on inference latency, PEFT methods present an appealing option for deployment in sectors like finance.

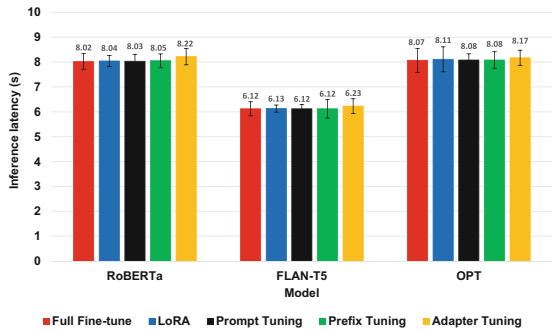


Fig. 7. Inference latency using various fine-tuning methods

4.6 Comparison with Traditional NLP Algorithms and Cutting-Edge Models

The extent to which LLMs, fine-tuned for financial text using lightweight adaptation methods, outperform traditional NLP models in finance-related tasks remains uncertain. Table 5 presents sentiment analysis results using LSTM models, a conventional choice for NLP tasks. By employing the OPT model fine-tuned with LoRA, we observed up to 18% increase in accuracy over the LSTM model. This superior performance suggests that self-supervised pre-training of LLMs, although costly, offers substantial benefits. Contrary to the common belief that fine-tuning LLMs requires a large dataset, our findings indicate that LSTMs need more data to achieve optimal performance compared to domain-adapted LLMs.

This study was limited to small-scale LLMs due to computing power constraints. Based on LLM scaling laws, we expect performance to improve with large-scale LLMs. Considering the emergent zero-shot learning abilities of large-scale LLMs, a pertinent question arises: are specialized models crafted through fine-tuning necessary, or can superior performance be achieved with general-purpose large-scale LLMs like ChatGPT? Comparing our fine-tuned models using PEFT methods to state-of-the-art large-scale LLMs, we observe that small-scale fine-tuned LLMs outperform GPT-4 and ChatGPT. Therefore, in specialized domains, it is preferable to fine-tune a small-scale domain-specific LLM rather than rely on a large-scale general-purpose LLM out-of-the-box.

Table 5. The modelling performance of LSTM models, state-of-the-art LLMs and OPT model fine-tuned using LoRA

Model	Accuracy (%)	F1-score
LSTM [2]	71	0.64
LSTM [24]	72	0.66
ChatGPT [17]	78	–
GPT-4 [17]	76	–
OPT+LoRA	89	0.89

5 Conclusion

This paper examines the effectiveness of parameter-efficient fine-tuning methods for adapting LLMs to sentiment analysis of financial data, focusing on both performance and training efficiency. Through a thorough analysis of methods such as LoRA, prompt tuning, prefix tuning, and adapter tuning, alongside traditional full fine-tuning, key insights emerged. Adapting the OPT model with LoRA achieved the highest performance with 89% accuracy, using less than 1.06% of the model's total parameters. The storage size for trainable parameters was minimal (0.1 MB to 7 MB), and PEFT methods added negligible inference

overhead while reducing peak GPU memory consumption by up to 80%. Additionally, small-scale fine-tuned LLMs outperformed state-of-the-art large-scale LLMs in specialized domains and demonstrated an 18% accuracy improvement over traditional LSTM models, justifying their higher implementation costs. The advancements from this paper can make AI more accessible, especially for those with limited computational resources, and contribute to environmental sustainability by reducing the energy and carbon footprint of training large-scale AI models.

Future Work. These findings emphasize the potential of PEFT methods for practical applications in various domains and highlight the need for further research to refine these techniques and explore their broader applicability in NLP. The following are possible future research avenues to build on this work. Firstly, exploring the composability of PEFT methods by stacking, fusing, or mixing different PEFT modules to harness their combined knowledge effectively. Additionally, leveraging lower-precision floating point representations (e.g., FP16) during training with PEFT methods has the potential to accelerate training speed.

Acknowledgement. The first author, K. Langa, wishes to acknowledge the DSI-CSIR NEPTTP for their generous financial support, which has made this work possible.

References

1. Aghajanyan, A., Zettlemoyer, L., Gupta, S.: Intrinsic dimensionality explains the effectiveness of language model fine-tuning. arXiv preprint arXiv:2012.13255 (2020)
2. Araci, D.: FinBERT: financial sentiment analysis with pre-trained language models (2019)
3. Chen, L., Zaharia, M., Zou, J.: FrugalGPT: how to use large language models while reducing cost and improving performance. arXiv preprint arXiv:2305.05176 (2023)
4. Chronopoulou, A., Baziotis, C., Potamianos, A.: An embarrassingly simple approach for transfer learning from pretrained language models. arXiv preprint arXiv:1902.10547 (2019)
5. Chung, H.W., et al.: Scaling instruction-finetuned language models. arXiv preprint arXiv:2210.11416 (2022)
6. Dalika, N.K., Seetharam, Y.: Sentiment and returns: analysis of investor sentiment in the South African market. Ph.D. thesis, University of the Witwatersrand, Faculty of Commerce, Law and Management (2014)
7. Dettmers, T., Pagnoni, A., Holtzman, A., Zettlemoyer, L.: QLORA: efficient finetuning of quantized LLMs. arXiv preprint arXiv:2305.14314 (2023)
8. Ding, N., et al.: Parameter-efficient fine-tuning of large-scale pre-trained language models. Nat. Mach. Intell. **5**(3), 220–235 (2023)
9. He, P., Liu, X., Gao, J., Chen, W.: DeBERTa: decoding-enhanced BERT with disentangled attention. In: International Conference on Learning Representations (2021). https://openreview.net/forum?id=XPZIaotutsD
10. Houlsby, N., et al.: Parameter-efficient transfer learning for NLP. In: International Conference on Machine Learning, pp. 2790–2799. PMLR (2019)

11. Hu, E.J., et al.: LoRA: low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021)
12. Hu, Z., et al.: LLM-adapters: an adapter family for parameter-efficient fine-tuning of large language models (2023)
13. Kumar, A., Raghunathan, A., Jones, R., Ma, T., Liang, P.: Fine-tuning can distort pretrained features and underperform out-of-distribution. arXiv preprint arXiv:2202.10054 (2022)
14. Lee, Y., et al.: Surgical fine-tuning improves adaptation to distribution shifts. arXiv preprint arXiv:2210.11466 (2022)
15. Lester, B., Al-Rfou, R., Constant, N.: The power of scale for parameter-efficient prompt tuning. arXiv preprint arXiv:2104.08691 (2021)
16. Li, X.L., Liang, P.: Prefix-tuning: optimizing continuous prompts for generation. arXiv preprint arXiv:2101.00190 (2021)
17. Li, X., Zhu, X., Ma, Z., Liu, X., Shah, S.: Are ChatGPT and GPT-4 general-purpose solvers for financial text analytics? An examination on several typical tasks. arXiv preprint arXiv:2305.05862 (2023)
18. Lialin, V., Deshpande, V., Rumshisky, A.: Scaling down to scale up: a guide to parameter-efficient fine-tuning. arXiv preprint arXiv:2303.15647 (2023)
19. Liu, R., Shi, Y., Ji, C., Jia, M.: A survey of sentiment analysis based on transfer learning. IEEE Access **7**, 85401–85412 (2019)
20. Liu, Y., et al.: RoBERTa: a robustly optimized BERT pretraining approach (2019)
21. Loughran, T., McDonald, B.: Textual analysis in finance. Annu. Rev. Financ. Econ. **12**, 357–375 (2020)
22. Malo, P., Sinha, A., Korhonen, P., Wallenius, J., Takala, P.: Good debt or bad debt: detecting semantic orientations in economic texts. J. Assoc. Inf. Sci. Technol. **65** (2014)
23. Mo, Y., Yoo, J., Kang, S.: Parameter-efficient fine-tuning method for task-oriented dialogue systems. Mathematics **11**(14), 3048 (2023)
24. Omarkhan, M., Kissymova, G., Akhmetov, I.: Handling data imbalance using CNN and LSTM in financial news sentiment analysis. In: 2021 16th International Conference on Electronics Computer and Computation (ICECCO), pp. 1–8. IEEE (2021)
25. Peters, M.E., Ruder, S., Smith, N.A.: To tune or not to tune? Adapting pretrained representations to diverse tasks. arXiv preprint arXiv:1903.05987 (2019)
26. Pfeiffer, J., et al.: AdapterHub: a framework for adapting transformers. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 46–54 (2020)
27. Wang, C., Yi, J., Zhang, X., Tao, J., Xu, L., Fu, R.: Low-rank adaptation method for Wav2Vec2-based fake audio detection (2023)
28. You, K., Liu, Y., Wang, J., Long, M.: LogME: practical assessment of pre-trained models for transfer learning. In: International Conference on Machine Learning, pp. 12133–12143. PMLR (2021)
29. Zhang, S., et al.: OPT: open pre-trained transformer language models (2022)
30. Zi, B., Qi, X., Wang, L., Wang, J., Wong, K.F., Zhang, L.: Delta-LoRA: fine-tuning high-rank parameters with the delta of low-rank matrices. arXiv preprint arXiv:2309.02411 (2023)