Journal of Electrical and Computer Engineering Innovations
(JECEI)
Journal homepage: http://www.jecei.sru.ac.ir

Research paper

Hybrid Fine-Tuning of Large Language Models Using LoRA: Enhancing Multi-Task Text Classification through Knowledge Sharing

A. Beiranvand, M. Sarhadi, J. Salimi Sartakhti *

Department of Computer Engineering, University of Kashan, Kashan, Iran.

Article Info

Article History:
Received 02 November 2024
Reviewed 13 January 2025
Revised 12 February 2025
Accepted 02 March 2025

Keywords:
Large language model
Fine-Tuning
PEFT
LoRA
Knowledge sharing
Attention mechanism

*Corresponding Author's Email Address: Salimi@kashanu.ac.ir

Abstract

Background and Objectives: Large Language Models have demonstrated exceptional performance across various NLP tasks, especially when fine-tuned for specific applications. Full fine-tuning of large language models requires extensive computational resources, which are often unavailable in real-world settings. While Low-Rank Adaptation (LoRA) has emerged as a promising solution to mitigate these challenges, its potential remains largely untapped in multi-task scenarios. This study addresses this gap by introducing a novel hybrid approach that combines LoRA with an attention-based mechanism, enabling fine-tuning across tasks while facilitating knowledge sharing to improve generalization and efficiency. This study aims to address this gap by introducing a novel hybrid fine-tuning approach using LoRA for multi-task text classification, with a focus on inter-task knowledge sharing to enhance overall model performance.

Methods: We proposed a hybrid fine-tuning method that utilizes LoRA to fine-tune LLMs across multiple tasks simultaneously. By employing an attention mechanism, this approach integrates outputs from various task-specific models, facilitating cross-task knowledge sharing. The attention layer dynamically prioritizes relevant information from different tasks, enabling the model to benefit from complementary insights.

Results: The hybrid fine-tuning approach demonstrated significant improvements in **accuracy** across multiple text classification tasks. On different NLP tasks, the model showed superior generalization and precision compared to conventional single-task LoRA fine-tuning. Additionally, the model exhibited better scalability and computational efficiency, as it required fewer resources to achieve comparable or better performance. Cross-task knowledge sharing through the attention mechanism was found to be a critical factor in achieving these performance gains.

Conclusion: The proposed hybrid fine-tuning method enhances the accuracy and efficiency of LLMs in multi-task settings by enabling effective knowledge sharing between tasks. This approach offers a scalable and resource-efficient solution for real-world applications requiring multi-task learning, paving the way for more robust and generalized NLP models.

Introduction

Large language models (LLMs) have become essential in artificial intelligence, especially for natural language processing (NLP) and various other applications. These models, characterized by their sophisticated architectures and deep neural networks, have fundamentally transformed NLP by demonstrating unparalleled capabilities in both generating and comprehending human language. The impact of LLMs extends beyond NLP [1], influencing fields such as machine translation, sentiment analysis, and even creative writing. Despite their transformative potential,

fully fine-tuning these models presents significant challenges. The primary obstacle lies in the sheer number of parameters, often reaching billions, which necessitates substantial computational resources and advanced hardware. This complexity not only increases the cost and time required for fine-tuning but also raises concerns about energy consumption and environmental impact. Consequently, researchers are exploring alternative approaches such as transfer learning, parameter-efficient tuning, and the development of more efficient model architectures to mitigate these challenges.

To utilize an LLM for various tasks, a common approach is to fine-tune a pre-trained model on the specific task data [2], [3]. Full fine-tuning of a language model can be computationally intensive, typically requiring the update of all parameters in the pre-trained model, and the fine-tuned model may end up with as many parameters as the original model [4]. To overcome this issue, parameter-efficient fine-tuning methods like Low-Rank Adaptation (LoRA) [5] enable fine-tuning a pre-trained model by introducing small LoRA modules for different tasks. In these methods, the main parameters of the pre-trained model remain fixed, and only the weights of the two low-rank matrices in LoRA are updated, which are significantly fewer in number compared to the main parameters of the pre-trained model.

LoRA significantly reduces the computational resources required and enables the fine-tuning process across various tasks. For example, thousands of LLaMA models [6], fine-tuned using LoRA, are available on Hugging Face Hub [7]. These practical applications demonstrate that LoRA is not only widely used for fine-tuning tasks in LLMs but also achieves model accuracy comparable to other full-weight fine-tuning methods. The lightweight nature of LoRA-based fine-tuning allows for training multiple LoRA modules on a single GPU. LoRA-based fine-tuning systems, such as Alpaca-LoRA [8], primarily focus on optimizing single-task fine-tuning and have not fully explored efficient strategies for multi-task fine-tuning.

Despite the successes achieved, the majority of existing research and systems have focused predominantly on single-task fine-tuning, with limited exploration of efficient strategies for multi-task fine-tuning. In this paper, we introduce a hybrid model that fine-tunes large language models using the LoRA method, enhancing model accuracy by enabling simultaneous learning across multiple tasks. This hybrid approach employs an attention mechanism to integrate the outputs of various tasks, yielding superior performance in diverse text classification tasks.

Existing parameter-efficient fine-tuning techniques, such as Low-Rank Adaptation (LoRA), have shown promise in reducing computational requirements. However, their applications have been largely limited to single-task learning, leaving multi-task scenarios underexplored. Multi-task learning, with its potential for inter-task knowledge sharing, offers significant advantages in terms of generalization and resource efficiency, yet it poses unique challenges in balancing task-specific requirements. To address these challenges, we propose a hybrid fine-tuning approach that enhances multi-task text classification by leveraging LoRA alongside an attention mechanism for effective knowledge sharing.

The main contributions of this paper are as follows:

- **Hybrid Fine-Tuning Approach**: This paper introduces a hybrid fine-tuning approach that fine-tunes large language models (LLMs) using Low-Rank Adaptation (LoRA). Unlike traditional fine-tuning methods that focus on single tasks, this hybrid approach enables simultaneous fine-tuning across multiple tasks. The central innovation lies in leveraging knowledge sharing between tasks, allowing the model to learn from multiple tasks concurrently and enhance its overall performance. By sharing task-specific knowledge, the model improves generalization and accuracy across diverse text classification challenges.
- **Advanced Attention Mechanism**: The model incorporates an attention mechanism that facilitates cross-task knowledge integration. This attention layer intelligently combines outputs from different tasks, allowing the model to dynamically focus on the most relevant information from each task. As a result, the model benefits from a broader understanding of the data, as insights gained from one task can enhance the performance on others. This inter-task knowledge sharing is a key driver of the model's superior accuracy and efficiency.
- **Improved Computational Efficiency**: While enhancing accuracy through multi-task knowledge sharing, the proposed method also significantly optimizes computational resources. By using LoRA, the model reduces the number of trainable parameters, allowing it to operate efficiently on a single GPU without compromising on performance. This combination of enhanced learning and reduced computational demands makes the approach highly suitable for practical, large-scale deployments.
- **Comprehensive Experimental Validation:** To evaluate the proposed approach, we conduct extensive experiments on multiple benchmark datasets for diverse text classification tasks. Our results demonstrate that the hybrid method achieves improved accuracy and efficiency compared to both single-task fine-tuning and the base model.

Preliminaries

This section delves into the preliminary concepts,

including the fine-tuning of large language models, parameter-efficient fine-tuning methods, and the attention mechanism.

A. Large Language Models

Large Language Models represent some of the most cutting-edge advancements in artificial intelligence, characterized by their ability to generate text with high precision and quality. These models leverage complex architectures and deep neural networks, which enable them to produce text that is both coherent and contextually appropriate across a diverse array of topics. The foundational architecture of these models is built upon transformers, a breakthrough introduced in 2017 that rapidly became a cornerstone in natural language processing due to its unparalleled efficacy [9]. Key applications of these models include automated content generation, machine translation, natural language processing, and question-answering systems. LLMs can analyse vast amounts of data and learn linguistic patterns, allowing them to generate sentences that are logical and meaningful, often indistinguishable from text written by humans [2].

Prominent examples of LLMs include GPT [10] by OpenAI, BERT [3], and T5 [11] by Google. These models, through the analysis of vast amounts of data, have learned intricate linguistic patterns, enabling them to generate sentences that are both logical and contextually rich, making it challenging to differentiate their output from text authored by humans. The primary advantage of these models lies in their exceptional ability to understand and generate high-quality text across multiple languages, as well as to provide accurate responses to questions of varying complexity. As a result, LLMs have found extensive applications in areas such as machine translation, chatbots, automated content creation, and even recommendation systems. Given these capabilities, LLMs are not only powerful tools for natural language processing but have also become foundational pillars in the development of AI-driven technologies. However, to fully harness their potential, these models often require fine-tuning for specific tasks. This fine-tuning is essential for optimizing their performance in particular applications.

In this research, we adopt a hybrid approach using Low-Rank Adaptation (LoRA) to fine-tune large language models (LLMs) for multi-task text classification. The hybrid approach is designed to optimize computational efficiency while enhancing model accuracy, primarily by facilitating inter-task knowledge sharing. This approach enables the model to benefit from the learning outcomes of different tasks simultaneously, allowing it to leverage relevant information gained across tasks to improve overall performance. In this context, our hybrid approach ensures that the advantages of multi-task learning—particularly the ability to transfer knowledge across tasks—are fully realized. The dynamic sharing of task-specific knowledge allows the model to become more robust, mitigating issues such as overfitting to a particular dataset while also enhancing its ability to adapt to new, unseen tasks. By fine-tuning different tasks in parallel and allowing for cross-task information flow, our model achieves higher performance metrics than traditional fine-tuning approaches that isolate task learning.

B. Fine-Tuning Language Models

Training a large language model (LLM) from scratch requires significant time and financial resources. The use of thousands of GPUs can take several days [12] and demands substantial financial investment [13]. Fine-tuning pre-trained models has emerged as an efficient way to leverage the benefits of LLMs: fine-tuning is the process of adapting a pre-trained model to a specific task by training it further on task-specific data, thereby improving its performance on that task. This approach has gained widespread acceptance, as it allows researchers to utilize general-purpose pre-trained models and tailor them to meet specific needs. Many organizations, such as Meta with their LLaMA models [6], make their pre-trained models publicly available. These publicly available pre-trained models can be fine-tuned for various downstream tasks, making fine-tuning the most practical way to capitalize on the benefits of LLMs. However, fully fine-tuning large language models is computationally expensive, as it requires updating all the parameters of the model.

C. Parameter-Efficient Fine-Tuning Methods

Full fine-tuning of large pre-trained models generally requires updating all model parameters, often resulting in substantial computational costs [4]. In contrast, parameter-efficient fine-tuning (PEFT) methods [14] selectively adjust only a small number of additional parameters, leading to a significant reduction in computational and memory resources. One of the most advanced PEFT techniques is the Low-Rank Adaptation (LoRA) method [5], which achieves efficient fine-tuning by keeping the pre-trained model entirely frozen and applying weight updates through a trainable low-rank decomposition matrix, as illustrated in Fig. 1.

$$h = W_0 x + \Delta W x = W_0 x + BAx \tag{1}$$

where x represents the input data from the target task, $W_0 \in R^{d\times k}$ are the weights of the pre-trained model that remain fixed, $B \in R^{d\times r}$ and $A \in R^{r\times k}$ and are the trainable low-rank decomposition matrices, where $r \ll \min(d, k)$ is the rank of the decomposition. Fig. 1 illustrates the LoRA method, which we employed as the primary tool for fine-tuning large language models in this research.

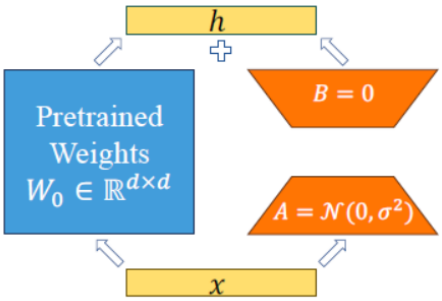

Fig. 1: Reparameterization in LoRA Method [5]. This method only trains A and B. The Pretrained Weight of model were frozen.

By utilizing LoRA, our research has been able to significantly reduce computational costs while simultaneously improving the accuracy and efficiency of the models across various text classification tasks.

D. Attention Mechanism

The attention mechanism is one of the key innovations in modern AI architectures, particularly in transformer-based models. This mechanism allows models to focus more on the relevant and important information within the data. Essentially, the attention mechanism assigns different weights to different inputs, helping the model to select the most important parts of the data for processing, thereby improving the overall performance of the model.

In transformer-based models, the attention mechanism is implemented as self-attention, where each word in a sentence attend to all other words in the same sentence. This mechanism consists of three matrices: query, key, and value. For each token, the model assigns different weights to each of these matrices based on their similarity with other tokens. The basic formula for the attention mechanism is given as (2).

$$Attention(Q, K, V) = softmax\left(\frac{QK}{\sqrt{d_k}}\right)V \quad (2)$$

where Q is the matrix of queries, K is the matrix of keys, and V is the matrix of values. The term d_k is a normalization factor that prevents the resulting values from becoming too large due to the dot product of the queries and keys. This process enables the model to identify and focus on the most relevant and important parts of the data [9].

To further enhance the model's ability to capture complex relationships within the data, the Multi-Head Attention mechanism is introduced. In Multi-Head Attention, instead of computing a single attention function, the model computes multiple attention functions (heads) in parallel. Each head independently performs the attention operation, capturing different aspects of the relationships between words. The outputs of these attention heads are then concatenated and linearly transformed to form the final output. The formula for Multi-Head Attention is as follows:

$$MultiHead(Q, K, V) = Concat(head_1, head_2, \dots, head_h)W^0 \quad (3)$$

where each $head_i$ is calculated as:

$$head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right) \quad (4)$$

here, W_i^Q, W_i^K and W_i^V are the weight matrices for the queries, keys, and values for the i-th attention head, and W^0 is the output weight matrix. The Multi-Head Attention mechanism allows the model to jointly attend to information from different representation subspaces at different positions, providing a more comprehensive understanding of the input data [9].

The attention mechanism is not only used as a tool to enhance the performance of language models, but it can also be leveraged to aggregate and combine the outputs of multiple models. This application is particularly useful in complex systems that require the integration of information from various sources. In such scenarios, the attention mechanism can assist the model in intelligently determining which parts of the different outputs should be given more focus, ultimately producing a higher-quality combined output. In this approach, the outputs of several models are fed as inputs to an attention layer. This allows the models to flexibly utilize the knowledge and information from multiple sources, thereby achieving higher accuracy and efficiency in solving complex tasks [15].

For instance, in a multilingual machine translation system, the outputs of different models trained for various languages can be combined using the attention mechanism to create more accurate and reliable translations. Similarly, in recommendation systems or data analysis applications, the outputs of multiple models can be aggregated using attention to achieve better results [16]. The effectiveness of using the attention mechanism for output aggregation lies in its ability to automatically learn which outputs and their components are more significant under different circumstances, thereby improving the final output and enhancing decision-making quality.

In this study, we employed an attention mechanism to intelligently combine the outputs of fine-tuned multi-task models, enabling more effective cross-task knowledge sharing. By integrating outputs from multiple tasks, this approach allows language models fine-tuned using the Low-Rank Adaptation method to draw on the relevant insights learned from different tasks concurrently. Instead of treating each task as isolated, the attention mechanism dynamically identifies and emphasizes critical features from each task, thereby enhancing the model's ability to generalize across diverse datasets.

Related Work

This section reviews related works in the field, starting with a number of studies that have full fine-tuned language models, adjusting all model parameters. Following this, we introduce several studies that have utilized parameter-efficient fine-tuning (PEFT) methods. Finally, we discuss works that have employed hybrid fine-tuning approaches.

A. Full Fine-Tuning of Language Models

Fully fine-tuning language models is a widely used approach for task-specific optimization. For instance, In [17], the authors explore the use of the BERT model for sentiment analysis, employing a full fine-tuning approach. In this method, the pre-trained BERT model, initially trained on a large text corpus, is further fine-tuned on a specific sentiment analysis dataset. This involves updating all the parameters of the BERT model to adapt it to the task at hand. The full fine-tuning process takes advantage of BERT's capability to understand the intricate dependencies between words in a sentence and to leverage various contextual cues. As a result, the fine-tuned model demonstrates significantly improved performance in sentiment analysis tasks.

Similarly, In another study [18], a new model called LUKE is introduced, which is specifically designed to enhance the representation of entities within text. LUKE incorporates an entity-aware self-attention mechanism, allowing the model to pay special attention to entities mentioned in the text. By fully fine-tuning the BERT model on several tasks related to entity recognition, LUKE is able to accurately identify and represent the semantic relationships between entities and the surrounding text, resulting in better overall performance.

In the work of Nogueira et al. [19], BERT is utilized to enhance the performance of passage re-ranking tasks. The approach involves a two-step process: first, an initial retrieval system provides a list of candidate passages for a given query. Then, the BERT model, which has been fully fine-tuned, is used to re-rank these passages. The BERT model processes the input queries and candidate passages, encoding the sentences and analysing the semantic relationships between the words. This allows the model to more accurately determine the relevance of each passage to the query, improving the final ranking.

Khashabi et al. [20] introduce the UnifiedQA model, which is built on transformer architectures and trained using a transfer learning approach. This model is capable of answering questions across various formats. The key innovation is the unification of all question-answering tasks into a single text format, which is then used to fully fine-tune the language model. This unification allows the model to leverage a larger and more diverse dataset during training, leading to increased accuracy and flexibility in handling different types of question-answering tasks.

Full fine-tuning allows models to adapt comprehensively to specific tasks, maximizing their performance. However, this approach requires updating all model parameters, leading to high computational costs and memory requirements, making it less feasible for resource-constrained environments or scenarios requiring frequent model updates.

B. Parameter-Efficient Fine-Tuning (PEFT)

To mitigate the resource demands of full fine-tuning, parameter-efficient fine-tuning (PEFT) methods have been proposed. Pfeiffer et al. [21] introduced a novel approach for adapting transformer models using adapter modules. Rather than fully fine-tuning all the parameters of the model, this method involves adding a small set of new parameters, known as adapters, for each specific task. These adapter modules are connected to different layers of the transformer model, allowing the model to be optimized for various tasks without altering the primary parameters of the model. This approach significantly reduces the time and computational resources required for fine-tuning, while also facilitating the sharing of base models across multiple tasks. The results demonstrate that using adapter modules can achieve performance comparable to full fine-tuning methods, and in some cases, even surpass them.

Li and Liang [22] introduced Prefix-Tuning, another PEFT method, where instead of fine-tuning all model parameters, a continuous prefix of text is optimized. This prefix is added to the input of a transformer model, guiding it during text generation. This method offers a cost-effective and flexible alternative to traditional full fine-tuning approaches. The prefix is directly appended to the input sequence, and only a small number of parameters related to this prefix are optimized, leaving the rest of the model unchanged. The models used in their experiments, primarily GPT-2 and other generative transformer models, demonstrated that Prefix-Tuning could achieve performance similar to, or even better than, full fine-tuning while only optimizing a small portion of the model's parameters.

Stickland and Murray [23] proposed the PALs (Projected Attention Layers) method, another PEFT approach that adapts the BERT model to various tasks by replacing standard attention layers with projected attention layers. This method allows the model to be effectively adapted to various tasks by adding additional parameters to the attention layers, which are fine-tuned for each specific task. This approach enables efficient adaptation without the need to fine-tune all of the model's parameters, significantly reducing the computational load while maintaining high performance across multiple tasks.

Zhang et al. [24] introduced LoRA-FA, a novel method designed to improve the efficiency of fine-tuning large language models. The primary goal of LoRA-FA is to reduce the memory overhead associated with fine-tuning LLMs by minimizing the need for activation memory without sacrificing performance or incurring heavy recomputation costs. LoRA-FA works by keeping the low-rank weight matrix A fixed while only updating the higher-rank weight matrix B. This effectively reduces the activation memory required during the fine-tuning process. The method has been extensively tested across various models and scales, with empirical results showing that LoRA-FA offers comparable accuracy to full parameter fine-tuning and LoRA, while significantly reducing memory costs.

PEFT methods significantly reduce computational and memory requirements, enabling fine-tuning on resource-limited devices. However, these methods may not achieve the same level of performance improvement as full fine-tuning, especially for tasks requiring deep model adaptations.

C. Hybrid Fine-Tuning Approaches

Hybrid approaches aim to balance the comprehensive adaptation of full fine-tuning with the efficiency of PEFT. These approaches combine the strengths of both full fine-tuning and parameter-efficient fine-tuning methods, aiming to optimize model performance while minimizing computational resources. These approaches leverage the comprehensive capabilities of full fine-tuning by updating a significant portion of the model's parameters while simultaneously employing parameter-efficient techniques to reduce the overall computational cost.

Sar et al. [25] introduced the USE model, designed to generate high-quality representations of sentences and textual phrases. This model is particularly geared towards improving performance across a range of natural language processing (NLP) tasks and machine learning applications. The USE model employs two main architectures for generating vector representations of sentences:

Transformer-based Encoder: This version of the USE model is built upon the transformer architecture and utilizes the multi-head attention mechanism to understand semantic dependencies within the text. Due to its ability to model complex relationships between words, this architecture is particularly well-suited for applications requiring high precision and detail.

Deep Averaging Network (DAN): This version is lighter and faster, focusing on the simplicity of averaging word embeddings rather than employing the full transformer architecture.

In [26], the authors propose an approach that integrates Low-Rank Adaptation (LoRA) with traditional fine-tuning techniques. This hybrid method allows for the efficient fine-tuning of large language models by updating both a small, low-rank set of parameters and a larger set of fully fine-tuned parameters. This combination enables the model to achieve high performance with reduced computational costs compared to traditional full fine-tuning methods.

Similarly, the work by Kim et al. [27] explores a hybrid fine-tuning approach that combines Prefix-Tuning with full parameter fine-tuning. By optimizing a prefix of continuous text in conjunction with full fine-tuning, the model benefits from the flexibility and efficiency of parameter-efficient methods while retaining the comprehensive improvements provided by full fine-tuning. The results demonstrate that this hybrid approach can achieve performance levels comparable to full fine-tuning while requiring fewer computational resources.

Hybrid approaches effectively combine the strengths of full and parameter-efficient fine-tuning, achieving a good trade-off between performance and resource efficiency. However, their complexity can increase implementation overhead and may require careful tuning to balance the contributions of different components.

By critically analysing these methods, this paper identifies the potential of hybrid approaches to leverage the advantages of knowledge sharing and efficiency, forming the foundation for the proposed hybrid fine-tuning method.

Methodology

In this section, we introduce a novel hybrid model designed to enhance the fine-tuning process of large language models (LLMs) using the Low-Rank Adaptation (LoRA) technique. Our proposed method is structured to combine the strengths of multiple fine-tuned models to improve overall accuracy and efficiency in various text classification tasks. Each component model in our hybrid architecture shares the same pre-trained weights W^*, but has its own learnable parameters ΔW_m, which correspond to the matrices A and B in the LoRA method. After fine-tuning for each task, the weights W_m are defined as:

$$W_m = W^* + \Delta W_m \tag{5}$$

During the fine-tuning process on the data of task m, the objective is to optimize ΔW_m. For this optimization is formulated as follows:

$$\mathcal{L}(\Delta W_m) = \min_{\Delta W} \sum_{n=1}^{N} -\log p(y_n \,|X_n;\, W^* + \Delta W_m) \tag{6}$$

where N denotes the number of training samples for task m, X_n represents the input samples, and y_n are their corresponding labels.

Fig. 2 illustrates the proposed framework and the integration of these fine-tuned models.

Each of these fine-tuned models functions as an independent module within the proposed framework, as depicted on the left side of Fig. 2. These modules are

specifically tailored to handle individual tasks by employing the LoRA (Low-Rank Adaptation) fine-tuning technique. This approach allows for efficient adaptation of large language models to specific tasks without requiring complete retraining of the model, thus optimizing computational resources. For example, as shown in the left section of Fig. 2, the i-th module is designed for the task of sentiment analysis. The input to this module is a sentence, which could represent any textual content, such as a review, a social media post, or a customer feedback comment. The module processes the input and classifies it into one of the predefined categories like here: positive or negative sentiment.

To ensure accurate classification, a task-specific head is integrated into the module. This head maps the model's internal representations to the class space relevant to the sentiment analysis task. This mapping step is crucial for transforming the abstract feature representations learned by the model into interpretable outputs aligned with the specific requirements of the task.

The training process is guided by the computation of an error signal, which quantifies the difference between the predicted class and the true class label. This error is calculated using the cross-entropy loss function (7), a widely used metric in classification problems that effectively handles the probabilistic nature of model outputs. Once the error is computed, it is backpropagated through the network, enabling the model to adjust its parameters and improve its performance on the sentiment analysis task during subsequent iterations.

$$Loss_{cls} = \mathcal{L}(Head(LM(X)), y) \quad (7)$$

This modular design not only enhances the flexibility of the overall framework but also ensures that each module is highly specialized and optimized for its respective task, ultimately contributing to the robustness and adaptability of the proposed system.

In this way, it is possible to fine-tune k distinct modules on k different tasks. Each module is trained to learn task-specific knowledge, ensuring optimal performance for its assigned task. This approach ensures that the expertise gained by each module is highly specialized, tailored to the unique requirements of the respective task.

The main proposed model, depicted on the right side of Fig. 2, is essentially a combination of these individual modules. By integrating the knowledge acquired by each module, the overall model leverages the specialized expertise of all modules to perform complex or multifaceted tasks effectively.

This design allows the system to benefit from the modularized training of diverse tasks, enabling a flexible and scalable architecture. As a result, the proposed framework can tackle multiple tasks simultaneously or sequentially by drawing on the task-specific knowledge embedded within its constituent modules. This modular combination enhances the adaptability and efficiency of the model, particularly in multi-task learning scenarios.

Based on the model depicted in Fig. 2, the outputs of the fine-tuned models are initially aggregated using the aggregation function Agg1 and converted into a vector. It is important to note that these model outputs represent the values computed by the language model before passing through the softmax layer and being transformed into probability values. Subsequently, these aggregated values pass through an attention layer. The attention layer computes a new vector for each of these input vectors, determining how much attention each task's output should receive and adjusting the values accordingly. All outputs from the attention layer are then passed to a second aggregation function, Agg2, and finally combined with the output of the neural network layer using the Agg3 function, forming the final output of the model.

Aggregation functions are versatile tools designed to combine multiple inputs into a singular, cohesive output. The most widely used aggregation functions include:

Mean Function: Computes the average of the input values, providing a balanced representation of the data.

Max/Min Function: Identifies the highest or lowest value among the inputs, highlighting the most extreme values.

Sum Function: Adds up all input values, offering a cumulative measure of the inputs.

These are just a few examples, and a variety of other functions can also be utilized depending on the specific needs of the task at hand.

Attention Layer: This layer utilizes the self-attention mechanism, where the outputs from the Agg1 function serve simultaneously as queries (Q), keys (K), and values (V). After applying Agg1, each output from the fine-tuned models functions as its own query, key, and value. This allows the model to not only focus on its own output but also to dynamically attend to the outputs of other models fine-tuned on different tasks. This interaction enables the model to weigh the relevance of each output in the context of the others, leading to a more informed and refined final result.

Normalization: To enhance the model's performance, the aggregated data is normalized. Normalizing the data ensures that the model inputs fall within a specified range, which aids in accelerating the learning process and improving prediction accuracy. Moreover, this process prevents the model from disproportionately focusing on features with larger scales, which could lead to imbalances in learning. Overall, data normalization not only reduces the model's training time but also helps in improving its generalization ability.

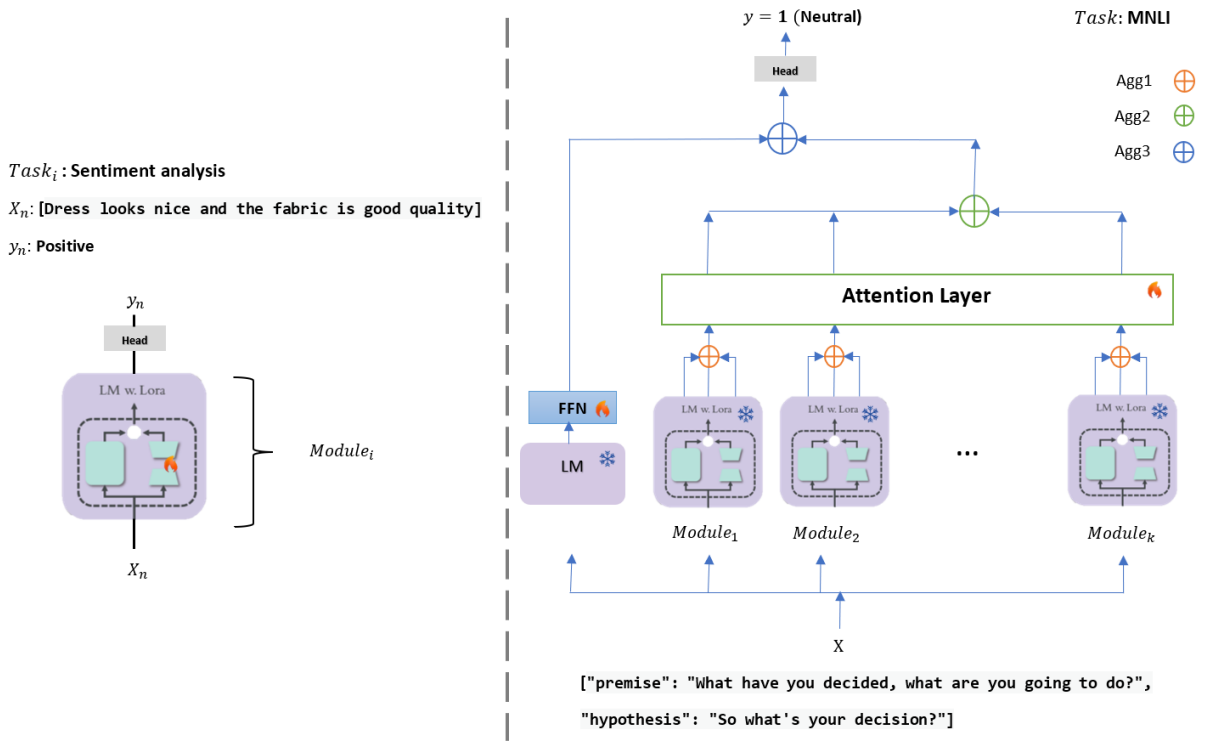

Fig. 2: The left image depicts a single module undergoing fine-tuning of a language model on a specific task (sentiment analysis) using the LoRA method. The right image illustrates the proposed framework, which combines these fine-tuned modules for the target task (here, MNLI).

For normalization, after each aggregation function, a normalization layer is added. In this layer, after calculating the mean μ and variance σ^2, the data is normalized according to (8):

$$\hat{X}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{8}$$

where ϵ is a small value added to prevent division by zero. Following normalization, by adding two learnable parameters, the output of the normalization layer is given by

$$y_i = \gamma \hat{x}_i + \beta \tag{8}$$

During the training phase of the proposed model, only the parameters of the Feed-Forward Network (FFN) and the attention layers are updated. The language model and all k modules remain frozen, with only their learned knowledge being shared across tasks. This design allows the model to focus on learning the new task (e.g., the MNLI task shown in Fig. 2) without altering the parameters of the pre-trained language model or the task-specific modules.

By leveraging this knowledge-sharing property, the framework can achieve high performance and accuracy on previously unseen tasks without the need to fine-tune the entire language model for each new task. This approach ensures that the expertise gained from prior tasks is effectively utilized to generalize to new scenarios, significantly reducing the need for extensive retraining.

Additionally, since the proposed framework employs parameter-efficient methods during the module training phase, such as LoRA (Low-Rank Adaptation), the computational cost of training is kept minimal. Unlike full fine-tuning approaches, which require updating the entire language model, this method modifies only a small subset of parameters. Consequently, the training and inference processes are computationally efficient, allowing the model to run effectively on a single GPU.

This efficiency makes the framework practical and scalable, particularly in resource-constrained settings, while maintaining high performance across both seen and unseen tasks. It demonstrates the capability of leveraging modular and efficient design principles to achieve robust task adaptation with minimal computational overhead.

Experiments and Results

To evaluate the effectiveness of the proposed method, several experiments were conducted. This section details the experiments and the results obtained from them.

A. Datasets

To fine-tune and evaluate the model, the GLUE dataset [28] was used, which encompasses multiple tasks. The details of the dataset are provided in Table 1.

To assess the generalization ability of the model, four additional datasets—STS-B, IMDB, AG News, and TREC—were employed, none of which were used in the training process of the hybrid method.

Table 1: Details of the datasets

	Task	Type	#Class	#Train	#Test	Labels	Metric
Single-sentence	SST-2	Sentiment	2	6920	872	positive, negative	Accuracy
	CoLA	acceptability	2	8551	1042	grammatical, not_grammatical	Matthews_Correlation
Sentence-pair	MNLI	NLI	3	392702	9815	entailment, neutral, contradiction	Accuracy
	QNLI	NLI	2	104743	5463	entailment, not_entailment	Accuracy
	RTE	NLI	2	2490	277	entailment, not_entailment	Accuracy
	WNLI	NLI	2	635	72	entailment, contradiction	Accuracy
	MRPC	Paraphrase	2	3668	408	equivalent, not_equivalent	Accuracy
	QQP	Paraphrase	2	363846	40431	equivalent, not_equivalent	Accuracy
Single-sentence	IMDB	Sentiment	2	25000	25000	positive, negative	Accuracy
	AG_NEWS	News Categorization	4	120000	7600	World, Sports, Business, Science/Technology	Accuracy
	TREC	Question Classification	6	5452	500	Abbreviation, Entity, Description, Human, Location, Numeric	Accuracy
Sentence-pair	STS-B	Sentiment. similarity	Regression	5749	1500	-	Pearson

The STS-B dataset, part of the GLUE benchmark [28], is designed for evaluating semantic similarity between sentence pairs. Each pair is annotated with a similarity score ranging from 0 to 5, where higher scores indicate greater semantic similarity.

The IMDB dataset [29], widely used for sentiment analysis, contains 50,000 movie reviews evenly split into 25,000 training samples and 25,000 test samples. Each review is labelled as either positive or negative, and the dataset is balanced, ensuring equal representation of both sentiment classes.

The AG News dataset [30] is a benchmark dataset used for news categorization. It consists of 120,000 training samples and 7,600 test samples, categorized into four classes: World, Sports, Business, and Science/Technology. Each sample includes a title and a brief description of the news article, making it ideal for evaluating text classification methods.

The TREC dataset [31], widely utilized for question classification, contains 5,452 training questions and 500 test questions categorized into six main types: Abbreviation, Entity, Description, Human, Location, and Numeric. These classes are further divided into finer subcategories, offering a hierarchical structure that is useful for question classification and intent detection tasks.

B. Evaluation Metrics

In this subsection, we describe the evaluation metrics used to assess model performance across different datasets, as summarized in Table 1.

Each metric is selected based on the specific nature and objectives of the corresponding task.

Accuracy is a standard evaluation metric for classification tasks. It measures the proportion of correctly predicted samples relative to the total number of samples. This metric is widely used in datasets such as SST-2, IMDB (sentiment analysis), AG_NEWS, TREC, MNLI, QNLI, RTE, WNLI (natural language inference), and MRPC, QQP (paraphrase detection) [28]. The formula for accuracy is given as:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (9)$$

where:

TP (True Positives): The number of correctly predicted positive samples.

TN (True Negatives): The number of correctly predicted negative samples.

FP (False Positives): The number of incorrectly predicted positive samples.

FN (False Negatives): The number of incorrectly predicted negative samples.

Matthews Correlation Coefficient (MCC) [28] is a robust evaluation metric specifically suited for binary classification tasks, particularly in scenarios with imbalanced datasets. It considers all four categories of the confusion matrix (true positives, true negatives, false

positives, and false negatives) to provide a balanced assessment of model performance. In this study, MCC is employed for the CoLA dataset, which focuses on grammatical acceptability. The formula for MCC is:

$$MCC = \frac{(FP*FN)-(TP*TN)}{\sqrt{(TN+FN)(TN+FP)(TP+FN)(TP+FP)}} \quad (10)$$

MCC ranges from −1 to +1, where +1 indicates perfect prediction, 0 indicates no better than random prediction, and −1 indicates total disagreement between prediction and observation.

Pearson Correlation Coefficient [28] is used to measure the strength and direction of the linear relationship between predicted and actual values in regression tasks. It is particularly relevant for the STS-B dataset, where the objective is to evaluate semantic similarity scores between sentence pairs. A higher correlation indicates a stronger agreement between predicted and true scores. The formula for the Pearson correlation is:

$$Pearson\ Correlation = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{n}(x_i-\bar{x})^2 \sum_{i=1}^{n}(y_i-\bar{y})^2} \quad (11)$$

where:

x_i and y_i: Predicted and actual values, respectively.

$\bar{x}$ and $\bar{y}$: Mean of predicted and actual values, respectively.

n: Number of samples.

A higher correlation value (closer to +1) indicates a stronger agreement between predicted and true scores.

C. Experimental Settings

For the experiments, the language models DistilBERT [32] with approximately 66 million parameters, BERT-base [3] with approximately 110 million parameters, and the ELECTRA [33] model in both small and base versions with approximately 14 million and 110 million parameters, respectively, were used. Each of the tasks was fine-tuned individually on these models using the LoRA method. The implementations were carried out in Python using the Transformers [34] and Huggingface PEFT [35] libraries.

In the experiments, the number of fine-tuned models was set to k=9, except for the experiment examining the number of tasks, where this variable is adjusted. During the LoRA fine-tuning process, the ΔW_m matrices were applied to all three matrices—key, value, and query—in the language model's attention module with a rank of r=4, chosen based on preliminary experiments where this value provided a balance between computational efficiency and model accuracy.

Lower values of r were found to reduce the model's expressive power, while higher values increased computational costs without significant gains in performance. The AdamW optimizer was used in all experiments, and the models were trained for 20 epochs. The initial learning rate was set to $2e-5$ with weight decay=0.01, and the batch size was 16 for all datasets. We employed the NVIDIA GeForce RTX 3090 24GB. The mean function was used for both aggregation functions Agg1 and Agg2, while the sum function was employed for Agg3. The number of heads in the model is 4. The code related to this paper is available at this link[1].

D. Evaluation of the Proposed Method's Performance

In this section, we present the results of training the proposed method on all datasets after 20 epochs. The results, as shown in Table 2 for the two language models utilized, indicate the validation accuracy achieved.

Table 2: The effectiveness of the proposed method (AttEns) and other methods

LLM	Method	WNLI	QQP	MRPC	RTE	MNLI-mm	MNLI	QNLI	CoLA	SST2	Average
DistilBERT	Model	0.5493	0.3771	0.3161	0.4729	0.3267	0.3178	0.4794	0.0000	0.4908	0.3700
	FFN	0.4366	0.7648	0.7107	0.5523	0.5288	0.5238	0.6908	0.1458	0.8337	0.5763
	FT_LoRA	0.4366	0.8432	0.6838	0.4440	0.7389	0.7426	0.7426	0.0000	**0.9162**	0.6164
	AttEns	**0.6447**	**0.8544**	**0.8014**	**0.6642**	**0.7610**	**0.7523**	**0.8473**	**0.3952**	0.8933	**0.7348**
BERT-base	Model	0.5633	0.6066	0.6789	0.5090	0.3302	0.3226	0.4880	0.0950	0.5149	0.4565
	FFN	0.3802	0.7600	0.6985	0.5956	0.5372	0.5294	0.6847	0.3084	0.8589	0.5947
	FT_LoRA	**0.5774**	0.8615	0.7009	0.5848	0.8087	0.7940	**0.8813**	**0.4993**	0.9082	0.7351
	AttEns	0.5633	**0.8654**	**0.8186**	**0.6714**	**0.8102**	**0.8007**	0.8795	0.4988	**0.9105**	**0.7576**
ELECTRA-Small	Model	0.5915	0.4742	0.3137	0.5270	0.3303	0.3216	0.5249	0.0104-	0.4988	0.4477
	FFN	0.5774	0.7542	0.7132	0.5523	0.4869	0.4787	0.7100	0.3268	0.6938	0.5881
	FT_LoRA	0.5633	0.8362	0.7377	0.5631	0.7823	0.7671	0.8504	0.3463	0.8772	0.7026
	AttEns	**0.5774**	**0.8396**	**0.8235**	**0.6787**	**0.7870**	**0.7708**	**0.8533**	**0.5312**	**0.8853**	**0.7496**
ELECTRA-Base	Model	0.3943	0.3604	0.6838	0.5018	0.3585	0.3532	0.4946	0.020-	0.5080	0.4568
	FFN	0.4788	0.8000	0.7436	0.5956	0.6472	0.6320	0.7810	0.5208	0.8337	0.6703
	FT_LoRA	0.4084	0.8822	0.8431	0.7003	0.8647	0.8607	0.9203	0.6012	0.9288	0.7788
	AttEns	**0.5774**	**0.8825**	**0.8627**	**0.7689**	**0.8651**	**0.8632**	**0.9211**	**0.6362**	**0.9369**	**0.8126**

[1] https://github.com/Azadeh297/Attention-hybrid-method

The models evaluated are as follows:

Model: The base language model without any additional training or fine-tuning was used to infer the data, and its accuracy was calculated.

FFN: A single neural network layer was added on top of the base language model and fine-tuned. In this configuration, the parameters of the base model remain fixed, and only the parameters of the added neural network layer are updated.

FT_LoRA: The language model was fine-tuned separately on the data using the LoRA method.

AttEns: The proposed method.

The values listed in Table 2 for all datasets represent accuracy, except for the CoLA dataset, where the evaluation metric is the Matthews correlation coefficient. Based on the results in Table 2, the proposed method has achieved the highest accuracy in most tasks. This improvement is attributed to the use of the attention mechanism and the integration of information from multiple tasks, allowing the model to identify more complex patterns and thereby achieve higher accuracy.

To better explain this improvement, consider the QNLI task, which is related to natural language inference. In this task, the AttEns method achieved an accuracy of over 92%. One reason for this improvement could be the method's ability to recognize complex relationships between sentences. For instance, imagine that the model needs to compare two sentences to determine whether the second sentence is a logical conclusion of the first. In traditional methods like FFN or FT_LoRA, this process is carried out directly without utilizing information from other tasks. However, in the AttEns method, the model also leverages information from other tasks, such as recognizing contradictions in MNLI and semantic sentence matching in QQP. This combination of information allows the model to perform better in identifying complex relationships, particularly in tasks related to natural language inference.

In tasks like RTE, MRPC, and WNLI, there is a significant difference in the accuracy obtained from the proposed method compared to the single fine-tuned model. One possible reason for this is the smaller number of samples in these datasets compared to others. The proposed hybrid model has demonstrated higher accuracy in situations where limited data is available, potentially because the single fine-tuned model might have overfitted due to the small dataset size, resulting in poorer performance. This finding suggests that the proposed method can offer better generalizability even in data-limited scenarios.

By comparing the results obtained from different language models, in many cases, the larger language model has yielded better results, indicating that using a larger language model with more parameters can be effective in improving model performance.

To further evaluate the proposed method, we applied it to four additional datasets, IMDB, STS-B, AG_NEWS and TREC where the fine-tuned models for these datasets were not used in the combination of the proposed method. The results are shown in Table 3. According to the results, the AttEns method outperformed the FFN and FT_LoRA methods on all four datasets. For example, on the IMDB dataset, the AttEns method achieved an accuracy of 0.9397, which is clearly better than the FFN (0.8405) and FT_LoRA (0.8789) methods. This improvement in accuracy demonstrates the proposed method's high generalization ability to unseen data and indicates better performance in real-world scenarios. Imagine that the model needs to detect the sentiment of a movie review in the IMDB dataset. In traditional methods like FFN or FT_LoRA, the model only uses the training data from the same dataset, limiting its ability to generalize to new data. However, in the AttEns method, the model uses the attention mechanism to also leverage information from other related tasks. For example, if the model learned how to identify similar sentences in the QQP dataset, it could enhance this knowledge and apply it to better understand the sentiment in movie reviews.

Table 3: Results of the proposed method on two datasets not used in the combination of the proposed method (BERT-base language model)

Method	STS-B	IMDB	AG_NEWS	TREC
Model	-0.0608	0.4956	0.2505	0.0180
FFN	0.2247	0.8405	0.8942	0.7666
AttEns	**0.4158**	**0.9397**	**0.9107**	**0.8220**

On the STS-B dataset, the AttEns method also achieved an accuracy of 0.4158, which is an improvement compared to other methods. This improved performance indicates that the proposed model has high generalization ability even in scenarios where data is limited or heterogeneous.

The results show that using the attention mechanism and integrating information from various tasks enables the model to better identify and analyse complex features in new data, which is particularly important in real-world scenarios and unseen data.

E. Evaluating the Impact of k

In this section, the impact of the number of components, k, in the proposed hybrid model is examined. The results of this experiment are presented in Fig. 3. This chart demonstrates that increasing the number of components generally improves the accuracy of the hybrid model, although the extent of this improvement varies at different points. As observed in Fig. 3, the accuracy of the model significantly improves when the number of components increases from 3 to 5.

This enhancement is due to the increased capacity of the model to learn and integrate diverse information from various tasks.

When the number of components reaches 7, accuracy continues to improve in some datasets, but the improvement is not as pronounced. As the number of components increases, the model can examine each part of the data with greater detail, resulting in better performance. However, while the model continues to improve in accuracy, this improvement becomes slower compared to earlier stages. This slowdown occurs because, with more components, the model needs to process more information, requiring more resources for complete and optimized processing. This observation highlights the importance of selecting an appropriate number of components to optimize the model's performance.

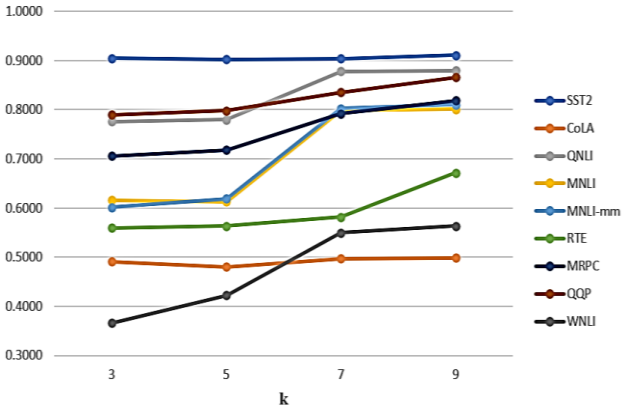

Fig. 3: Results of the proposed method for different values of k (BERT-base language model).

Limitations and Future Work

While the proposed hybrid fine-tuning method demonstrates notable improvements in accuracy and computational efficiency, several limitations must be acknowledged. Firstly, the method's effectiveness has been primarily validated on text classification tasks. Its applicability to other NLP domains, such as sequence generation or machine translation, remains unexplored and requires further investigation. Secondly, the reliance on specific datasets like GLUE may constrain the generalizability of the findings to other domains or languages. Future research should aim to extend this approach to a broader range of datasets and languages while examining its compatibility with other parameter-efficient fine-tuning techniques. Additionally, exploring the trade-offs between various aggregation functions and attention mechanisms could provide valuable insights for further optimizing model performance.

Results and Discussion

The results obtained from various datasets indicate that the proposed AttEns method consistently outperforms traditional fine-tuning approaches such as FFN and FT_LoRA across multiple NLP tasks. As shown in Table 2, AttEns achieves the highest accuracy in most datasets, particularly excelling in QNLI (92.11%), RTE (86.51%), and IMDB (93.97%). This improvement is primarily due to the model's ability to integrate task-specific information through an attention-based ensemble mechanism, which enhances its ability to identify complex patterns and generalize across tasks. Additionally, the method exhibits superior performance on datasets with limited samples, such as RTE and MRPC, where single-task fine-tuning often leads to overfitting.

Furthermore, the generalization ability of AttEns is evident in its strong performance on IMDB, STS-B, AG_NEWS, and TREC, despite these datasets not being explicitly incorporated into the model's training. The improvement in STS-B (41.58%) suggests that the attention mechanism enables the model to leverage knowledge from related tasks, leading to better sentence similarity evaluation. Moreover, the analysis of k, the number of task components, reveals that increasing k enhances performance up to a certain point, after which the improvement plateaus due to computational constraints. Overall, these results highlight the effectiveness of AttEns in improving language model accuracy, particularly in multi-task learning and low-data scenarios.

Conclusion

This paper introduced a hybrid approach for fine-tuning large language models using the LoRA method, which is capable of improving model accuracy by learning multiple tasks simultaneously. The results from the experiments showed that this method outperformed traditional fine-tuning methods, especially on the GLUE dataset. The use of the attention mechanism to integrate and influence different tasks was one of the main factors contributing to the success of this method. Additionally, the method demonstrated good generalizability on unseen data. Ultimately, this research marks a significant step towards reducing computational costs and enhancing the efficiency of large language models in various natural language processing tasks.

Author Contributions

All authors contributed equally to the conception, design, methodology, data analysis, manuscript preparation, and revision of this research. All authors have reviewed and approved the final version of the manuscript.

Acknowledgment

We sincerely thank the authors of previous studies whose work has informed and inspired this research. We

also extend our heartfelt appreciation to the respected referees for their thorough review of this paper.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this manuscript.

Abbreviations

NLP	Natural Language Processing
NLI	Natural Language Inference
LLM	Large Language Model
LoRA	Low Rank Adaptation
PEFT	Parameter-Efficient Fine-Tuning
MCC	Matthews Correlation Coefficient
TP	True Positives
TN	True Negatives
FP	False Positives
FN	False Negatives
FT	Fine Tuning
FFN	Feed Forward Network
GLUE	General Language Understanding Evaluation
CoLA	Corpus of Linguistic Acceptability
SST-2	Stanford Sentiment Treebank
MRPC	Microsoft Research Paraphrase Corpus
QQP	Quora Question Pairs
STS-B	Semantic Textual Similarity Benchmark
MNLI	Multi-Genre NLI
QNLI	Question NLI
RTE	Recognizing Textual Entailment
WNLI	Winograd NLI
IMDB	Internet Movie Database
TREC	Text Retrieval Conference
AttEns	Attention Ensemble

References

[1] K. I. Roumeliotis, N. D. Tselikas, "Chatgpt and open-ai models: A preliminary review," Future Internet, 15(6): 192, 2023.

[2] T. Brown et al., "Language models are few-shot learners," in Proc. Advances in Neural Information Processing Systems 33 (NeurIPS 2020), 33: 1877-1901, 2020.

[3] J. Devlin, M. W. Chang, K. Lee, K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.

[4] K. Lv, Y. Yang, T. Liu, Q. Gao, Q. Guo, X. Qiu, "Full parameter fine-tuning for large language models with limited resources," arXiv preprint arXiv:2306.09782, 2023.

[5] E. J. Hu et al., "Lora: Low-rank adaptation of large language models," arXiv preprint arXiv:2106.09685, 2021.

[6] H. Touvron et al., "Llama 2: Open foundation and fine-tuned chat models," arXiv preprint arXiv:2307.09288, 2023.

[7] Hugging Face. https://huggingface.co/, 2023.

[8] Eric Wang. Alpaca-lora. https://github.com/tloen/alpaca-lora, 2023.

[9] A. Vaswani et al., "Attention is all you need," in Proc. Advances in neural information processing systems 30 (NIPS 2017), 2017.

[10] J. Achiam et al., "Gpt-4 technical report," arXiv preprint arXiv:2303.08774, 2023.

[11] C. Raffel et al., "Exploring the limits of transfer learning with a unified text-to-text transformer," J. Mach. Learn. Res., 21(140): 1-67, 2020.

[12] D. Narayanan et al., "Efficient large-scale language model training on gpu clusters using megatron-lm," in Proc. the International Conference for High Performance Computing, Networking, Storage and Analysis: 1-15, 2021.

[13] O. Sharir, B. Peleg, Y. Shoham, "The cost of training nlp models: A concise overview," arXiv preprint arXiv:2004.08900, 2020.

[14] S. Mangrulkar, S. Gugger, L. Debut, Y. Belkada, S. Paul, B. Bossan, "Peft: State-of-the-art parameter-efficient fine-tuning methods," 2022.

[15] A. Hernández, J. M. Amigó, "Attention mechanisms and their applications to complex systems," Entropy, 23(3): 283, 2021.

[16] S. Dathathri et al., "Plug and play language models: A simple approach to controlled text generation," arXiv preprint arXiv:1912.02164, 2019.

[17] C. Sun, X. Qiu, Y. Xu, X. Huang, "How to fine-tune bert for text classification?," in Proc. Chinese computational linguistics: 18th China National Conference (CCL 2019): 194-206, 2019.

[18] I. Yamada, A. Asai, H. Shindo, H. Takeda, Y. Matsumoto, "LUKE: Deep contextualized entity representations with entity-aware self-attention," arXiv preprint arXiv:2010.01057, 2020.

[19] R. Nogueira, K. Cho, "Passage Re-ranking with BERT," arXiv preprint arXiv:1901.04085, 2019.

[20] D. Khashabi et al., "Unifiedqa: Crossing format boundaries with a single qa system," arXiv preprint arXiv:2005.00700, 2020.

[21] J. Pfeiffer et al., "Adapterhub: A framework for adapting transformers," arXiv preprint arXiv:2007.07779, 2020.

[22] X. L. Li, P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," arXiv preprint arXiv:2101.00190, 2021.

[23] A. C. Stickland, I. Murray, "Bert and pals: Projected attention layers for efficient adaptation in multi-task learning," in Proc. International Conference on Machine Learning, PMLR: 5986-5995, 2019.

[24] L. Zhang, L. Zhang, S. Shi, X. Chu, B. Li, "Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning," arXiv preprint arXiv:2308.03303, 2023.

[25] D. Cer et al., "Universal sentence encoder," arXiv preprint arXiv:1803.11175, 2018.

[26] N. Shazeer et al., "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," arXiv preprint arXiv:1701.06538, 2017.

[27] X. Wang, L. Aitchison, M. Rudolph, "LoRA ensembles for large language model fine-tuning," arXiv preprint arXiv:2310.00035, 2023.

[28] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," arXiv preprint arXiv:1804.07461, 2018.

[29] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, C. Potts, "Learning word vectors for sentiment analysis," in Proc. the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: 142-150, 2011.

[30] X. Zhang, J. Zhao, Y. LeCun, "Character-level convolutional networks for text classification," in Proc. Advances in neural information processing systems 28 (NIPS 2015), 2015.

[31] X. Li, D. Roth, "Learning question classifiers," in Proc. COLING 2002: The 19th International Conference on Computational Linguistics, 2002.

[32] V. Sanh, L. Debut, J. Chaumond, T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," arXiv preprint arXiv:1910.01108, 2019.

[33] K. Clark, M.-T. Luong, Q. V. Le, C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," arXiv preprint arXiv:2003.10555, 2020.

[34] T. Wolf et al., "Transformers: State-of-the-art natural language processing," in Proc. the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations: 38-45, 2020.

Biographies

Azadeh Beiranvand Borjele completed Bachelor's degree in Software Engineering in 2005 and Master's degree in Artificial Intelligence in 2012 at Shahid Chamran University, Ahvaz, Iran. Currently, She is currently a doctoral student in Artificial Intelligence at University of Kashan, Kashan, Iran. Her research interests include graph representation learning, graph neural networks, large language models and dynamic complex networks.

- Email: a.Beiranvand@grad.kashanu.ac.ir
- ORCID: 0009-0007-7077-8896
- Web of Science Researcher ID: NA
- Scopus Author ID: NA
- Homepage: https://scholar.google.com/citations?user=81IV9sEAAAAJ&hl=en

Mahdiye Sarhadi Dadiyan completed Bachelor's degree in Information Technology Engineering in 2010 at Zahedan PNU, Zahedan, Iran and Master's degree in Computer Engineering (Artificial Intelligence) in 2016 at Kharazmi University, Tehran, Iran. Right now, She is a doctoral student in Artificial Intelligence at University of Kashan, Kashan, Iran. Her research interests include reinforcement learning, deep learning, large language models.

- Email: mahdiye.sarhadi@grad.kashanu.ac.ir
- ORCID: 0009-0006-5351-1885
- Web of Science Researcher ID: NA
- Scopus Author ID: NA
- Homepage: NA

Javad Salimi Sartakhti is an Assistant Professor of Artificial Intelligence in the department of Computer Engineering at the University of Kashan, Iran. He obtained his B.Sc. degree in computer engineering from the University of Kashan and his M.Sc. degree in Software Engineering from the Tarbiat Modares University, Tehran, Iran, in 2008 and 2013, respectively. In January 2017, he obtained his Ph.D. degree in Artificial Intelligence at the Isfahan University of Technology. He ranked first among students of computer engineering in all three degrees. His main research interests are LLM, NLP, and Deep learning.

- Email: salimi@kashanu.ac.ir
- ORCID: 0000-0003-1183-1232
- Web of Science Researcher ID: HJY-2812-2023
- Scopus Author ID: 51864592100
- Homepage: https://faculty.kashanu.ac.ir/salimi/en

How to cite this paper:
A. Beiranvand, M. Sarhadi, J. Salimi Sartakhti, “Hybrid fine-tuning of large language models using lora: enhancing multi-task text classification through knowledge sharing,” J. Electr. Comput. Eng. Innovations, 13(2): 417-430, 2025.

DOI: 10.22061/jecei.2025.11314.794

URL: https://jecei.sru.ac.ir/article_2303.html