Journal of Artificial Intelligence and Data Mining (JAIDM), Vol. 13, No. 1, 2025, 107-117.

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Research paper

Multilingual Language Models in Persian Natural Language Processing Tasks: A Performance Comparison of Fine-Tuning Techniques

Ali Reza Ghasemi and Javad Salimi Sartakhti*

Artificial Intelligence Group, Faculty of Electrical and Computer Engineering, University of Kashan, Kashan, Iran.

Article Info

Article History:
Received 09 December 2024
Revised 17 January 2025
Accepted 12 February 2025

DOI:10.22044/jadm.2025.15167.2625

Keywords:
Fine-Tuning Techniques, Large Language Models in Low-Resource Languages, Multilingual Language Models, multilingual BERT.

**Corresponding author: salimi@kashanu.ac.ir (J. Salimi Sartakhti).*

Abstract

This paper evaluates the performance of various fine-tuning methods for multilingual language models in Persian natural language processing tasks. In low-resource languages like Persian, which suffer from a lack of rich and sufficient data for training large models, it is crucial to select appropriate fine-tuning techniques that mitigate overfitting and prevent the model from learning weak or surface-level patterns. The main goal of this research is to compare the effectiveness of fine-tuning approaches such as Full-Finetune, LoRA, AdaLoRA, and DoRA on model learning and task performance. We apply these techniques to three different Persian NLP tasks: sentiment analysis, named entity recognition (NER), and span question answering (QA). For this purpose, we conduct experiments on three Transformer-based multilingual models with different architectures and parameter scales: BERT-base multilingual (~168M parameters) with Encoder only structure, mT5-small (~300M parameters) with Encoder-Decoder structure, and mGPT (~1.4B parameters) with Decoder only structure. Each of these models supports the Persian language but varies in structure and computational requirements, influencing the effectiveness of different fine-tuning approaches. Results indicate that fully fine-tuned BERT-base multilingual consistently outperforms other models across all tasks in basic metrics, particularly given the unique challenges of these embedding-based tasks. Additionally, lightweight fine-tuning methods like LoRA and DoRA offer very competitive performance while significantly reducing computational overhead and outperform other models in Performance-Efficiency Score introduced in the paper. This study contributes to a better understanding of fine-tuning methods, especially for Persian NLP, and offers practical guidance for applying Large Language Models to downstream tasks in low-resource languages.

1. Introduction

Since the introduction of large language models (LLMs), these models have become a fundamental tool in natural language processing (NLP), particularly in text processing, and have significantly surpassed traditional methods like recurrent neural networks (RNNs) [1] and their enhanced versions Like Long short-term memory (LSTM) [2]. These new language models, based on the attention mechanism [3] and the Transformer architecture, enable processing of text while preserving long-range dependencies and provide better control over NLP challenges such as ambiguity.

Despite the impressive advancements of LLMs, several challenges remain in adapting these models to various tasks and datasets. One of the main issues is the enormous number of parameters in these models, making the training process highly

resource-intensive. Full training of these large models requires extensive computational power, long training times, and vast amounts of data, which are often beyond the reach of small or medium-sized enterprises and individual researchers. To mitigate this, language models are typically pre-trained on large datasets for general NLP tasks such as masked language modeling [4] and then made available for further fine-tuning on specific tasks.

Fine-tuning, an essential approach in this context, involves adapting pre-trained models to specific tasks by adjusting their parameters to optimize performance on the given task. Initially, full fine-tuning of all model parameters was the most common approach. However, the high computational cost of updating all parameters during fine-tuning remained a significant challenge. To address this, more efficient fine-tuning methods, such as LoRA (Low-Rank Adaptation), were introduced [5]. These techniques aim to reduce hardware and computational costs while striving to improve or maintain model accuracy, achieving performance comparable to fully fine-tuned models. Particularly in resource-constrained environments, these methods can significantly accelerate the fine-tuning process of large language models, enabling them to perform well across various NLP tasks with limited data. In comparison, fully fine-tuning a model requires substantially more data to adjust all of its parameters, which can be resource-intensive but effective in well-resourced settings.

Persian, as a language with unique characteristics such as complex syntax and morphology, a diverse vocabulary, and cultural nuances in word meanings, poses particular challenges for NLP. Additionally, Persian has absorbed vocabulary and expressions from various regional dialects and underlanguages, and the widespread use of Finglish (Persian written in the Latin alphabet) in informal communication further complicates text processing. Another significant challenge is the absence of diacritics (Harekat), which are typically omitted in Persian text, unlike in Arabic or English. This makes it difficult for NLP models to disambiguate words, such as "خِرد" (wisdom) and "خُرد" (broken), which are written identically without diacritics. Moreover, Persian contains many homographs that share the same written form and vowel marks but have different meanings depending on context. For instance, "مهر" can refer to the "sun", "a month of Mehr in the Persian calendar" or "friendship/love". These features, coupled with inherent ambiguities, make the need for specialized models and fine-tuning methods even more pressing. LLMs, however, are not initially trained to handle all languages and often focus on languages like English during pre-training. With the rapid development of multilingual language models (Multilingual LLMs) [6], it has become possible to process multiple languages simultaneously, reducing the need to train separate models for each language.

This shift towards multilingual models comes with two primary challenges: first, ensuring that the model retains its linguistic capabilities across multiple languages, and second, optimizing the model to deliver strong performance in low-resource languages like Persian. By overcoming these challenges, multilingual LLMs can leverage shared linguistic features and semantic representations across languages, enabling better understanding and processing of various languages.

This paper explores the performance of these models and their fine-tuning methods on Persian NLP tasks, examining their effectiveness in handling underrepresented languages like Persian to achieve improved accuracy, robustness, and generalization in various applications, such as sentiment analysis, named entity recognition, and question answering. Additionally, it discusses the trade-offs in computational efficiency when employing Parameter-Efficient Fine-Tuning methods, highlighting how these techniques balance performance with resource constraints.

2. Related Work

In the comprehensive benchmarking study by Abaskohi et al. [7], the efficacy of large language models for Persian was explored, with a focus primarily on GPT-3.5-turbo, alongside evaluations of GPT-4 and OpenChat-3.5. This study assessed these models across a variety of Persian language tasks, including mathematical problems, entailment, sentiment analysis, multiple-choice question answering (MCQA) in common and literary domains, MCQA focusing on math and logic, named entity recognition (NER), reading comprehension, and elementary school benchmarks. To address the lack of available Persian datasets, two new benchmarks were introduced, specifically targeting reasoning tasks. The findings revealed that while LLMs like GPT-4 performed exceptionally well in tasks requiring reasoning and general knowledge, they often fell short when compared to smaller pre-trained models that were specifically fine-tuned for particular tasks. Additionally, performance improvements were observed when test sets were translated to English before inputting them into GPT-3.5,

further illustrating the challenges faced by LLMs in processing the Persian language [7].

However, this study primarily relied on closed-source models like GPT-3.5 and GPT-4, and used zero or few-shot learning [8] with prompt engineering, without a focus on direct fine-tuning of these models. Due to the closed-source nature of the models and their massive size, fine-tuning GPT-4 was impractical and unfeasible. Instead, the study placed greater emphasis on a broad range of tasks by generating and embedding relevant data. This limitation presents an opportunity in the current literature, highlighting the need for further research on fine-tuning techniques specifically tailored for Persian NLP tasks. Such techniques could complement this valuable study and push the boundaries of Persian language processing.

This paper aims to address these gaps by selecting fewer tasks but focusing on smaller, open-source models, which are more accessible and manageable compared to GPT-3.5 or GPT-4. Rather than relying on zero or few-shot learning, this study emphasizes directly fine-tuning the weights of these models using different parameter-efficient methods such as LoRA. These methods are particularly useful for enhancing model adaptability in low-resource settings. Consequently, this paper evaluates various fine-tuning methods, including Full-Finetune, LoRA, AdaLoRA, and DoRA, to optimize performance on Persian language tasks. In doing so, it aims to provide deeper insights into effective strategies for underrepresented languages using smaller, open-source models.

In the study by Shuttleworth et al. [21], the authors examine the structural and behavioral differences between Low-Rank Adaptation (LoRA) and full fine-tuning methods in adapting pre-trained large language models to downstream tasks. Through spectral analysis of the models' weight matrices, they discover that LoRA introduces “intruder dimensions”—new, high-ranking singular vectors not present in fully fine-tuned models. These intruder dimensions lead to distinct generalization behaviors, where LoRA models, despite achieving similar performance on target tasks, exhibit increased forgetting of the pre-training distribution and reduced robustness in continual learning scenarios compared to fully fine-tuned models. The study concludes that even when LoRA and full fine-tuning attain comparable accuracy, they access different parts of the parameter space, resulting in non-equivalent solutions [21].

While our papar also focuses on parameter-efficient fine-tuning methods, such as LoRA, AdaLoRA, and DoRA, for Persian language models, it diverges from Shuttleworth et al.'s work by concentrating on the application and evaluation of these methods in low-resource settings. Specifically, we assess the effectiveness of these fine-tuning techniques on Persian NLP tasks, aiming to optimize performance with minimal parameter updates. While low-rank adapters often yield results very close to those of fully fine-tuned models in many cases, it is important to note— as highlighted in Shuttleworth et al.'s paper—that while LoRA models may not match the full richness of a fully fine-tuned model, they can still provide very competitive performance. This trade-off becomes particularly significant in low-resource situations, where full fine-tuning may require extensive amounts of data and computational resources. In such settings, LoRA family fine-tuning methods can offer a valuable alternative by achieving strong performance with fewer data and resources. In contrast, Shuttleworth et al. provide a theoretical analysis of the structural differences between LoRA and full fine-tuning, without a specific focus on underrepresented languages or low-resource scenarios. Thus, our study complements their findings by applying parameter-efficient fine-tuning methods in a practical, low-resource context, contributing to the broader understanding of their applicability across diverse languages and settings.

3. Model Selection

In this paper, we target three relatively small to medium-sized models for fine-tuning: Multilingual BERT Base, mT5 Small, and mGPT. All of these open-source models are readily accessible from the Hugging Face model hub. We believe that these three models represent some of the best options across a variety of transformer architectures and parameters. With this selection, we encompass Encoder-only, Encoder-Decoder, and Decoder-only transformer models and providing a diverse range of parameter counts from approximately 170 million to 1.4 billion parameters. This allows us to examine the effects of parameter-efficient methods across a spectrum of model sizes and architectures.

3.1. Multilingual BERT Base

This model is an Encoder-only transformer and a variant of Bidirectional Encoder Representations from Transformers (BERT) developed by Google. It is pre-trained on the top 104 languages, including Persian, using a masked language modeling (MLM) objective and next sentence prediction (NSP) on the largest Wikipedia. With approximately 177 million parameters, its lack of a decoder limits its capability for generative tasks,

such as text generation. However, it is well-suited for various downstream tasks, including sequence classification, token classification, and question answering [4,6].

3.2. mT5 Small

This model is an Encoder-Decoder transformer and a variant of the Text-to-Text Transfer Transformer (T5) from Google [9]. It is pre-trained on 101 languages, including Persian, using the multilingual Colossal Clean Crawled Corpus (mC4). The mT5 model converts all NLP tasks into a text-to-text format, making it versatile for a wide range of applications. It has approximately 170 million parameters and benefits from both an encoder and a decoder, making it an excellent choice for embedding and generative tasks [10].

3.3. mGPT

This model is a Decoder-only transformer and an unofficial, multilingual variant of the Generative Pre-training Transformer (GPT-2) originally developed by OpenAI [11]. It has been retrained by Ai-forever to support multiple languages. It is pre-trained on 61 languages, including Persian, using the multilingual Colossal Clean Crawled Corpus (mC4) and Wikipedia. As an autoregressive model, mGPT reproduces the GPT-3 architecture using sources from GPT-2, incorporating a sparse attention mechanism. With around 1.4 billion parameters, its lack of an encoder makes it less effective for embedding and supervised learning tasks but a strong candidate for generative and unsupervised learning applications [12].

4. Evaluation Metrics

Evaluating the performance of a model requires selecting appropriate metrics based on the task. The following metrics are used to assess classification and text generation tasks.

- Accuracy: Accuracy measures the proportion of correctly predicted instances out of the total instances. It is a straightforward metric that provides an overall sense of the model's correctness but may not be sufficient when dealing with imbalanced datasets.
- Recall: Recall, also known as sensitivity, measures the proportion of correctly identified positive instances out of all actual positive instances in the dataset. It is useful for evaluating how well the model captures all relevant instances.
- Precision: Precision calculates the proportion of correctly predicted positive instances out of all instances predicted as positive. It indicates how reliable the model's positive predictions are, reducing the impact of false positives.
- F1 Score: The F1 Score is the harmonic mean of precision and recall. It balances both metrics, providing a single value that considers both the accuracy of positive predictions and the ability to capture all relevant positives.
- Exact Match (EM): The exact match metric assesses whether the predicted answer span precisely matches the ground truth answer. A high exact match score signifies that the model consistently identifies the correct answer without discrepancies, serving as a strict measure of accuracy in answer extraction.
- ROUGE (Recall-Oriented Understudy for Gisting Evaluation): ROUGE evaluates the quality of generated text by comparing it to reference texts. The Persian ROUGE score functions similarly to the standard ROUGE score but is tailored for Persian texts. This metric evaluates the overlap between the predicted answer and the reference answer, focusing on recall, precision, and F1 metrics for n-grams. A high ROUGE score indicates that the model extracts answers closely aligned with human references, reflecting effective answering and content relevance.

In classification tasks with multiple classes, standard precision, recall, and F1 scores may not reflect performance evenly across all classes, especially in imbalanced datasets. Macro-averaging provides an equal-weighted evaluation by computing the metric for each class independently and then averaging the results. the macro-averaging approach ensures that the model's evaluation is not biased toward dominant classes, making it particularly valuable in tasks with class imbalance.

5. Tasks and Datasets

Natural Language Processing covers a wide range of tasks that can be addressed by language models. However, task selection in this study is constrained by both the model architectures and the availability of data, especially in low-resource languages like Persian. For instance, BERT, as an encoder-only transformer, is typically limited to tasks that rely on embeddings and supervised learning, rather than generative tasks. Additionally, the scarcity of large, high-quality datasets in Persian narrows our task options. Considering these limitations, we focus on three traditional downstream tasks with sufficient data availability: Sentiment Analysis, Named Entity Recognition (NER), and Span Question Answering (QA). These tasks are widely studied and have established benchmarks, making them

suitable for evaluation across multiple models and fine-tuning methods.

5.1. Sentiment Analysis

Sentiment Analysis is the process of determining the sentiment or opinion expressed in a text, often as part of sequence classification tasks in NLP. The task typically involves binary classification (e.g., Positive or Negative) or multi-class classification (e.g., Positive, Negative, or Neutral). In this paper, we use the "asparius/Persian-Food-Sentiment" dataset from Hugging Face, which is a binary sentiment analysis dataset specifically for Persian. The dataset consists of 56.7k training samples, 6.3k validation samples, and 7k test samples, all of which are labeled for sentiment polarity (Positive or Negative). Each sample in the dataset contains two columns: the label (sentiment) and the text [13]. We set the maximum token length (MAX_LEN) to 150 tokens, which is sufficient to cover almost all texts in the dataset while keeping computational efficiency in mind.

The evaluation of the Sentiment Analysis task is conducted using the Accuracy, Macro Recall, Macro Precision, Macro F1 Score.

5.2. Named Entity Recognition

Named Entity Recognition (NER) is a task aimed at identifying and classifying named entities (such as people, organizations, locations) within a text. NER is a multi-class classification task where each entity type (e.g., PERSON, LOCATION) is treated as a separate class. For this task, we use the "mansoorhamidzadeh/Persian-NER-Dataset-500k" from Hugging Face. Although the full dataset contains approximately 500k samples, we use a subset of 100k rows for training, 10k for validation, and 10k for testing to optimize memory usage and training time. Each sample in the dataset has three features: ner_tags (a list of named entity tags), tokens (a list of tokens in the sample), and ner_tags_index (the corresponding index values for the NER tags, mapping to specific named entities) [14]. To manage memory and processing time due to the large input size, we set MAX_LEN to 125 tokens, which covers the majority of the token lists in the dataset while balancing resource usage.

The following metrics were used to evaluate the Named Entity Recognition task: accuracy, macro recall, macro precision, and macro F1 score.

5.3. Span Question Answering

Span-based Question Answering (Span QA) is a task where the model is required to predict the start and end positions of the answer within a given context, based on a provided question. For this task, we use the "SLPL/syntran-fa" dataset from Hugging Face, which contains approximately 48k samples in Persian. We split the dataset into 38k rows for training, 4.8k for validation, and 4.8k for testing. Each sample consists of three features: question, fluent_answer (the context), and short_answer (the answer) [15]. For this task, the input sequence is a concatenation of the context and the question, with a MAX_LEN of 100 tokens. This token limit is sufficient to handle most samples in this dataset, as the combined length of the context and question is generally short. Additionally, this dataset is relatively straightforward, providing a good baseline for evaluating span-based question answering in Persian.

The performances of the Span Question Answering task are evaluated by Macro F1 Score, Exact Match, Persian ROUGE Score

6. Fine-tuning Methods

Fine-tuning pre-trained language models is a crucial step in this work, as it adapts the rich linguistic and contextual knowledge embedded in large models for specific downstream tasks. In this section, we explore various fine-tuning techniques, each offering different trade-offs between computational cost and performance. The goal is to identify the most appropriate solution for each task by balancing efficiency and accuracy. In this paper, we pay special attention to modern Low Rank Adapter methods. These methods offer significant advantages over Adapter-based methods in terms of parameter efficiency and integration simplicity. By introducing low-rank matrices, these methods enables substantial performance improvements with fewer trainable parameters, making it particularly beneficial for large models with limited training resources. Additionally, Low Rank Adapter methods can be seamlessly integrated into existing architectures without the need for complex structural modifications, unlike Adapter-based methods, which require inserting additional modules between layers.

6.1. Full Fine-Tuning

Full fine-tuning involves updating all model parameters (including both the task-specific head and the entire pre-trained language model) during training. This method allows the model to fully adapt to the target task, potentially leveraging the most task-specific knowledge. However, this comprehensive updating comes with increased computational demands and risks of overfitting, particularly on smaller datasets. While it typically leads to faster convergence, the large number of trainable parameters can cause the model to shift

some of its pre-trained linguistic generalization in favor of the task at hand.

6.2. LoRA

Low-Rank Adaptation (LoRA) is a parameter-efficient fine-tuning method that significantly reduces the number of trainable parameters by freezing most of the model and injecting small, trainable low-rank matrices, specifically in the attention layers. This approach reduces the memory and computational cost of fine-tuning large models, making it a feasible option when resources are limited. LoRA is particularly suited for tasks where only subtle adjustments to the pre-trained model are needed to achieve good performance [5]. In our experiments, we employed the Hugging Face PEFT (Parameter-Efficient Fine-Tuning) library, which enables efficient adaptation of large models with minimal trainable parameters. We used settings: r=8 (rank of the low-rank matrices), lora_alpha=32 (scaling factor), and lora_dropout=0.1 (to prevent overfitting).

6.3. DoRA

Weight-Decomposed Low-Rank Adaptation (DoRA) addresses the accuracy gap often observed between traditional full fine-tuning (FT) and Low-Rank Adaptation (LoRA) methods in parameter-efficient fine-tuning (PEFT). While LoRA has gained popularity for minimizing additional inference costs, it frequently falls short in performance compared to FT. DoRA enhances the learning capacity of LoRA by employing a novel weight decomposition analysis that separates pre-trained weights into magnitude and direction components. This innovative approach allows for more effective directional updates while reducing the number of trainable parameters. Empirical results indicate that DoRA consistently outperforms LoRA in fine-tuning across various downstream tasks, achieving improved training stability without incurring extra inference overhead [16].

6.4. AdaLoRA

Adaptive Low-Rank Adaptation (AdaLoRA) further refines the concept of low-rank adaptation by dynamically adjusting the rank at a layer-wise level. AdaLoRA improves parameter-efficient fine-tuning (PEFT) by addressing the limitations of traditional methods like LoRA, which distribute incremental update budgets evenly across all weight matrices, often leading to suboptimal performance. AdaLoRA introduces an adaptive mechanism that allocates the parameter budget based on the importance of different weight matrices. By parameterizing updates through singular value decomposition (SVD), AdaLoRA effectively prunes unimportant updates while avoiding the computational burden of exact SVD. This approach results in more efficient use of resources and enhances fine-tuning performance, particularly in low-budget scenarios. Experiments show that AdaLoRA consistently outperforms baseline methods on tasks like natural language processing, question answering, and language generation [17]. Like LoRA and DoRA, we utilized the same configuration (r=8, lora_alpha=32, lora_dropout=0.1) in our experiments.

7. Results

In our experiments, we used the PyTorch library along with the PyTorch Lightning framework for cleaner and more structured implementation. We used AdamW optimizer, CrossEntropy loss, and the following settings in our experiments:

- Batch size: 32
- Early stopping: Implemented with a fixed patience limit on validation step to prevent overfitting
- Random seed: 42

We utilized Google Colab Pro for our experiments, which provided access to NVIDIA A100 GPUs With 40GB of Video Random Access Memory (VRAM), a high-speed GPU memory essential for handling large models and datasets in deep learning. The CPU configuration included an Intel Xeon processor with 4 virtual CPUs (vCPUs), and the system was equipped with approximately 25GB of RAM.

During training, we fed 50% of the training data randomly in each epoch, simulating a dynamic sampling approach that prevents the model from seeing the same data order in every epoch. This technique aims to improve generalization and reduce the risk of overfitting, especially when paired with early stopping.

Tables 1, 2, 3, 5, 6, 7, 9, 10, and 11 provide a detailed comparison of the number of trainable and non-trainable parameters for each model under different fine-tuning methods. These tables highlight the reduction in trainable parameters when using parameter-efficient fine-tuning methods and computational resources.

The final performance of each model was evaluated based on specific metrics for each task. The results in Table 4, 8, and 12 summarize the evaluation after the training process (either after the set number of epochs or an early stopping mechanism).

7.1. Sentiment analysis task results

For the sentiment analysis task, we fine-tuned using a simple classification head. The classifier consists of two output neurons, utilizing the state of the last token from the LLM's output, following the standard approach for "SequenceClassification" [18]. We used the following hyper-parameters in our experiments:

- Learning rate: 2e-5
- Number of epochs: 5
- Patience (Number of waits with no improvement): 1

Table 1. Number of parameters and computational resources for Multilingual BERT Base in Sentiment Analysis task

Method	Parameters and Requirement			
	Trainable	Non-Trainable	Total	VRAM
Full fine-tune	177M	**0**	177M	6.20GB
LoRA	326K	177M	177M	3.50GB
DoRA	344K	177M	177M	4.17GB
AdaLoRA	474K	177M	177M	3.42GB

Table 2. Number of parameters and computational resources for mT5 Small in Sentiment Analysis task

Method	Parameters and Requirement			
	Trainable	Non-Trainable	Total	VRAM
Full fine-tune	172M	0	172M	5.40GB
LoRA	344K	172M	172M	8.00GB
DoRA	362K	172M	172M	6.00GB
AdaLoRA	1.9M	172M	174M	6.40GB

Table 3. Number of parameters and computational resources for mGPT 1.4B in Sentiment Analysis task

Method	Parameters and Requirement			
	Trainable	Non-Trainable	Total	VRAM
Full fine-tune	-	-	-	-
LoRA	1.6M	1.4B	1.4B	27.3GB
DoRA	1.7M	1.4B	1.4B	32.8GB
AdaLoRA	2.4M	1.4B	1.4B	27.2GB

Table 4. Post-training metrics on test data in Sentiment Analysis task

Model	Accuracy	F1_score	Precision	Recall
BERT full fine-tune[a]	**0.8665**	**0.8564**	**0.9265**	0.7961
BERT LoRA	0.8485	0.8407	0.8865	0.7994
BERT DoRA	0.8468	0.8377	0.8908	0.7905
BERT AdaLoRA	0.6978	0.6928	0.7044	0.6815
mT5 full fine-tune[a]	0.8555	0.8511	0.8781	0.8257
mT5 LoRA	0.7904	0.7795	0.8221	0.7411
mT5 DoRA	0.7882	0.7770	0.8205	0.7378
mT5 AdaLoRA	0.4957	0.6509	0.4977	**0.9403**
mGPT LoRA	0.8427	0.8332	0.8868	0.7857
mGPT DoRA	0.8418	0.8444	0.8307	0.8585
mGPT AdaLoRA	0.7194	0.7050	0.7431	0.6706

[a] *The model is over-fitted and the early stopping mechanism is invoked.*

7.2. Named Entity Recognition task result

For NER task, we employed a fine-tuning strategy using a classification head that contains 41 output neurons, corresponding to the 41 distinct entity classes present in the dataset. This approach follows the standardized methodology for "TokenClassification" [19], where each token in the input sequence is classified into one of the 41 classes (40 entities classes and a non-entity class). The classification head receives the token-level representations generated by the pre-trained language model and uses these to predict the appropriate class for each token. We used the following hyper-parameters in our experiments:

- Learning rate: 2e-5
- Number of epochs: 6
- Patience: 2

Table 5. Number of parameters and computational resources for Multilingual BERT Base in Named Entity Recognition task

Method	Parameters and Requirement			
	Trainable	Non-Trainable	Total	VRAM
Full fine-tune	177M	0	177M	6.00GB
LoRA	296K	177M	178M	3.80GB
DoRA	314K	177M	178M	3.76GB
AdaLoRA	444K	177M	178M	3.20GB

Table 6. Number of parameters and computational resources for mT5 Small in Named Entity Recognition task

Method	Parameters and Requirement			
	Trainable	Non-Trainable	Total	VRAM
Full fine-tune	146M	0	146M	4.80GB
LoRA	135K	146M	147M	2.80GB
DoRA	141K	146M	147M	2.50GB
AdaLoRA	888K	146M	147M	2.70GB

Table 7. Number of parameters and computational resources for mGPT 1.4B in Named Entity Recognition task

Method	Parameters and Requirement			
	Trainable	Non-Trainable	Total	VRAM
Full fine-tune	-	-	-	-
LoRA	1.7M	1.4B	1.4B	23.8GB
DoRA	1.8M	1.4B	1.4B	30.7GB
AdaLoRA	2.4M	1.4B	1.4B	23.9GB

7.3. Question Answering task result

For the Span Question Answering (QA) task, we employed a fine-tuning strategy using a classification head with 2 output neurons, corresponding to the positions of the start and end tokens in the input sequence. This approach

follows the standardized methodology for extractive "QuestionAnswering" [20], where the model identifies the span of text within the input passage that contains the answer. The classification head receives the token-level representations and uses these to predict the start and end positions of the answer span in token level. We used the following hyper-parameters in our experiments: Learning rate: 1e-5, Number of epochs: 10, Patience: 5.

Table 8. Post-training metrics on test data in Named Entity Recognition task.

Model	Accuracy	F1_score	Precision	Recall
BERT full fine-tune[a]	**0.9155**	**0.5512**	**0.6349**	**0.5366**
BERT LoRA	0.8755	0.2464	0.3566	0.2183
BERT DoRA	0.8774	0.2592	0.4040	0.2326
BERT AdaLoRA	0.8507	0.1919	0.3515	0.1669
mT5 full fine-tune[a]	0.8972	0.3545	0.4861	0.3058
mT5 LoRA	0.8265	0.0943	0.1282	0.0845
mT5 DoRA	0.8244	0.0913	0.1307	0.0823
mT5 AdaLoRA	0.8002	0.0555	0.0765	0.0518
mGPT LoRA	0.8564	0.2725	0.4893	0.2269
mGPT DoRA	0.8572	0.2860	0.4304	0.2475
mGPT AdaLoRA	0.8398	0.2468	0.4117	0.2032

[a] *The model is over-fitted and the early stopping mechanism is invoked.*

Table 9. Number of parameters and computational resources for Multilingual BERT Base in Question Answering task.

Method	Parameters and Requirement			
	Trainable	Non-Trainable	Total	VRAM
Full fine-tune	177M	0	177M	5.30GB
LoRA	296K	177M	177M	2.60GB
DoRA	314K	177M	177M	2.98GB
AdaLoRA	444K	177M	177M	2.50GB

Table 10. Number of parameters and computational resources for mT5 Small in Question Answering task.

Method	Parameters and Requirement			
	Trainable	Non-Trainable	Total	VRAM
Full fine-tune	172M	0	172M	6.20GB
LoRA	345K	172M	172M	3.57GB
DoRA	363K	172M	172M	4.00GB
AdaLoRA	1.9M	172M	174M	4.45GB

Table 11. Number of parameters and computational resources for mGPT 1.4B in Question Answering task.

Method	Parameters and Requirement			
	Trainable	Non-Trainable	Total	VRAM
Full fine-tune	-	-	-	-
LoRA	1.6M	1.4B	1.4B	20.0GB
DoRA	1.7M	1.4B	1.4B	24.6GB
AdaLoRA	2.4M	1.4B	1.4B	20.1GB

Table 12. Post-training metrics on test data in Question Answering task.

Model	F1_score	Rouge	Exact_match
BERT full fine-tune[a]	**0.9684**	**0.9952**	**0.9881**
BERT LoRA	0.9010	0.9875	0.9617
BERT DoRA	0.9497	0.9852	0.9611
BERT AdaLoRA	0.8255	0.9458	0.8493
mT5 full fine-tune[a]	0.9098	0.9884	0.9679
mT5 LoRA	0.1696	0.2886	0.0704
mT5 DoRA	0.1864	0.3124	0.0762
mT5 AdaLoRA	0.0130	0.0785	0.0018
mGPT LoRA	0.4492	0.6476	0.3633
mGPT DoRA	0.3747	0.6607	0.3550
mGPT AdaLoRA	0.2955	0.5388	0.1881

[a] *The model is over-fitted and the early stopping mechanism is invoked.*

8. Discussion

8.1. Results overview

In this section, we discuss the results obtained across various fine-tuning methods and models, focusing on model sizes, VRAM consumption, and learning performance. Despite using the same PEFT configurations across models, VRAM usage varied across different fine-tuning techniques and tasks due to the distinct approaches these methods employ. As expected, full fine-tuning consumed the most VRAM. Among the parameter-efficient methods, DoRA utilized more VRAM than LoRA, while AdaLoRA consumed the least. However, when examining the number of learned parameters, AdaLoRA generally outnumbered the other methods, followed by DoRA, with LoRA learning the fewest parameters.

At first glance, AdaLoRA's ability to learn more parameters while using less memory seems promising. However, our results show that AdaLoRA underperformed across all tasks and models, sometimes failing to learn effectively, as indicated by near-zero progress in certain cases. While not definitive, a likely explanation is that AdaLoRA's adaptive parameter allocation may sometimes underestimate the importance of key weight matrices, leading to an underrepresentation of critical features. Additionally, its SVD-based parameterization introduces approximations that might remove subtle yet important signal details, which can be especially impactful in low-budget settings. Finally, the sensitivity of its adaptive mechanism to hyperparameter settings and model-specific factors could result in inconsistent improvements compared to more uniformly distributing methods like LoRA and DoRA. In contrast, DoRA consistently yielded better results than both LoRA and AdaLoRA, and while its performance was not as strong as full fine-tuning, it still provided a reasonable trade-off between parameter efficiency and memory consumption.

Throughout the experiments, we used fixed hyper-parameters for all tasks to maintain a controlled

comparison across models and methods. While tuning hyper-parameters individually for each model might have improved performance, we opted for fixed settings to ensure fairness in our comparisons. As shown in Tables 4, 6, and 12, full fine-tuning delivered the best results in all tasks. However, it exhibited signs of early overfitting, which suggests that training could be halted earlier. On the other hand, models trained with DoRA and LoRA did not show signs of overfitting, which is promising, as it indicates these models could continue to improve with additional training while significantly reducing computational resources.

When examining model performance, BERT consistently outperformed all other models across tasks, regardless of the fine-tuning method. The full fine-tuned mT5 was competitive with the LoRA and DoRA versions of BERT, but it did not reach the performance of the full fine-tuned BERT. Moreover, the parameter-efficient versions of mT5 generally performed worse than their BERT counterparts, and the learning speed of mT5 was noticeably slower compared to BERT.

mGPT, on the other hand, showed significant improvement across metrics, but due to its substantially larger number of parameters, the allocated number of epochs was insufficient for thorough training. This under-fitting was evident in the validation and test loss curves. Despite this, we believe mGPT has the potential to surpass BERT given enough training time, although its computational requirements are far higher. In our experiments, parameter-efficient versions of mGPT required between 20GB to 40GB of VRAM for training, and we were unable to fully fine-tune mGPT on an A100 GPU with 40GB VRAM due to these demands.

8.2. Performance-Efficiency Score

While metrics like F1-score, accuracy, precision, etc are essential for evaluating task-specific performance, they do not account for the significant variations in computational resources required by different models. The trade-off between model performance and computational efficiency is a crucial consideration, especially with the increasing deployment of large-scale pre-trained models. To address this gap, we propose the Performance-Efficiency Score (PES) —a novel metric designed to capture both model effectiveness and resource efficiency. PES integrates traditional performance metrics with the total number of trainable parameters and VRAM usage, providing a more holistic evaluation. By incorporating a logarithmic scaling of resource usage, PES ensures that models with excessive resource demands are appropriately penalized, while still rewarding smaller, more efficient models that achieve competitive performance. A possible formula and structure for it is shown in Equations (1):

$$PES = \frac{\sum_i W_i M_i}{\log P \times \log V \times \sum_i W_i} \quad (1)$$

Where M_i represents task performance metrics such as F1 score, accuracy, or any other relevant evaluation metric. The impact coefficient W_i is dynamically adjusted according to the relative importance and contribution of each metric to the specific task, ensuring a balanced evaluation across different performance dimensions. P denotes the number of LLM parameters in millions, and V represents the VRAM usage in gigabytes.

For the Sentiment Analysis (SA) task, we assigned an equal weight of 2.5 to each metric, as it is a binary as symmetric classification problem. For Named Entity Recognition (NER), we used a weight of 1 for accuracy, 3 for F1 score, 2 for precision, and 4 for recall, prioritizing the correct identification of entities rather than non-entities, given that the majority of tokens are non-entities. In the Question Answering (QA) task, we applied a weight of 3 to the F1 score, 2 to the ROUGE score, and 5 to exact match, as more stringent metrics like exact match hold greater significance. The results are shown in Figure 1.

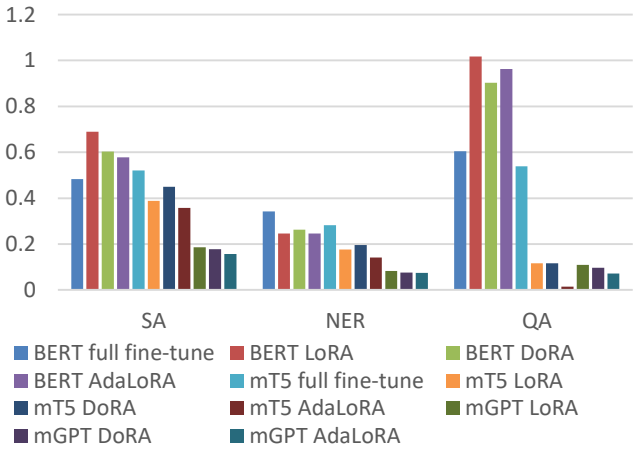

Figure 1. Performance-Efficiency Score for Models BERT, mT5, and mGPT in tsaks Sentiment Analysis, Named Entity Recognition, and Question Answering with fine-tuning methods full fine-tuning, LoRA, DoRA, and AdaLoRA.

Based on the results obtained across all tasks, it is evident that BERT fine-tuned with LoRA, followed closely by DoRA, emerges as the most effective and resource-efficient approach. LoRA demonstrates a strong balance between performance and computational efficiency,

offering competitive accuracy with a significantly lower number of trainable parameters and reduced VRAM usage compared to full fine-tuning. DoRA, while slightly more resource-intensive than LoRA, shows comparable performance and adapts well to the complexity of the tasks, making it a viable alternative where more flexibility is required.

9. Conclusion and Future work

In this paper, we evaluated the performance of several parameter-efficient fine-tuning methods—LoRA, DoRA, and AdaLoRA—on three different tasks: Sentiment Analysis, Named Entity Recognition (NER), and Question Answering (QA). Our results demonstrate that full fine-tuning consistently offers the highest performance in terms of accuracy and task-specific metrics across all tasks. However, parameter-efficient methods like LoRA and DoRA provide competitive results while significantly reducing computational costs, particularly in terms of the number of trainable parameters and VRAM consumption.

BERT-based models, particularly with LoRA and DoRA fine-tuning, emerged as the most efficient solutions, balancing performance and resource efficiency. This is especially important for resource-constrained environments, where full fine-tuning is impractical. Our findings show that lightweight fine-tuning methods like LoRA and DoRA offer the best overall balance, with LoRA being the most resource-efficient while maintaining competitive performance, followed closely by DoRA, which showed slightly better results at a marginally higher resource cost.

In addition, we introduce the Parameter Efficiency Score (PES) to quantify the trade-offs between performance and computational resources. PES allowed us to evaluate and compare the efficiency of different models and fine-tuning methods by incorporating both task performance metrics and resource usage. Our results show that LoRA and DoRA outperform other methods in terms of PES, making them highly attractive for low-resource settings.

Furthermore, we suggest that for downstream and embedding-based tasks in low-resource environments, using small Encoder models like BERT with full fine-tuning is a highly effective option. For scenarios involving larger models, where hardware limitations are a concern, using LoRA family adapters, especially LoRA and DoRA, proves to be a very efficient approach. However, it is crucial to consider the potential trade-offs such as longer training times and the need for more data to avoid under-fitting when using these efficient fine-tuning methods. These findings can guide future efforts in adapting large language models for resource-constrained environments.

In future work, we aim to explore additional parameter-efficient fine-tuning methods and revisit existing techniques, applying them to other multilingual language models with hyper-parameters and training settings specifically optimized for each model and method to achieve optimal performance. We also plan to improve AdaLoRA's performance by refining its adaptive parameter allocation and reducing approximation errors resulting from the SVD-based parameterization. These refinements will facilitate more accurate identification of critical weight matrices and contribute to more robust outcomes. Our approaches will be evaluated on more complex tasks and larger datasets. Additionally, expanding our research to include languages beyond Persian and the multilingual models already tested will allow us to generalize our findings across diverse linguistic contexts. To further enhance the model's practical applicability, future evaluations will involve testing on completely unseen datasets. We also intend to investigate the impact of fine-tuning methods on the interpretability of Persian-specific linguistic structures, such as morphology and syntax, to gain a deeper understanding of how these techniques influence language representation and task-specific performance. Furthermore, we aim to enhance the PES metric by incorporating additional factors related to hardware resource consumption. Specifically, we will consider factors such as training time, energy usage, memory requirements, and GPU utilization. These considerations will enable a more comprehensive and accurate assessment of model efficiency. By integrating these factors, we seek to ensure that the PES metric evaluates not only the model's performance in terms of accuracy and parameters but also provides a more holistic view of the computational resources required to achieve that performance. These improvements will be tested and incorporated into PES in future experiments to ensure proper accounting of resource efficiency when evaluating model performance. This will offer a more well-rounded perspective on both performance and resource optimization in practical applications.

References

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in Neural *Information Processing Systems*, 2017, pp. 5998–6008.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics, 2019.

[5] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-Rank Adaptation of Large Language Models," arXiv preprint arXiv:2106.09685, 2021.

[6] T. Pires, E. Schlinger, and D. Garrette, "How Multilingual Is Multilingual BERT?," arXiv preprint arXiv:1906.01502, 2019.

[7] A. Abaskohi, S. Baruni, M. Masoudi, N. Abbasi, M. Babalou, A. Edalat, S. Kamahi, S. Mahdizadeh Sani, N. Naghavian, D. Namazifard, P. Sadeghi, Y. Yaghoobzadeh, "Benchmarking Large Language Models for Persian: A Preliminary Study Focusing on ChatGPT," arXiv preprint arXiv:2404.02403, 2024.

[8] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal Journal of Machine Learning Research, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language Models are Few-Shot Learners," in Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901, 2020.

[9] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer,", vol. 21, pp. 1–67, 2020.

[10] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel, "mT5: A massively multilingual pre-trained text-to-text transformer," arXiv preprint arXiv:2010.11934, Oct. 22, 2020.

[11] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language Models are Unsupervised Multitask Learners," OpenAI, 2019.

[12] O. Shliazhko, A. Fenogenova, M. Tikhonova, V. Mikhailov, A. Kozlova, and T. Shavrina, "mGPT: Few-Shot Learners Go Multilingual," arXiv preprint arXiv:2204.07580, 2022.

[13] M. Farahani, M. Gharachorloo, M. Farahani, and M. Manthouri, "ParsBERT: Transformer-based model for Persian language understanding," arXiv preprint arXiv:2005.12515, 2020.

[14] M. Hamidzadeh, "Persian-NER-Dataset-500k," Hugging Face, 2024. [Online]. Available: https://huggingface.co/datasets/mansoorhamidzadeh/Persian-NER-Dataset-500k. [Accessed: Feb. 9, 2025].

[15] S .Sabouri, "syntran-fa," Hugging Face, [Online]. Available: https://huggingface.co/datasets/SLPL/syntran-fa. [Accessed: Feb. 9, 2025].

[16] S.-Y. Liu, C.-Y. Wang, H. Yin, P. Molchanov, Y.-C. F. Wang, K.-T. Cheng, and M.-H. Chen, "DoRA: Weight-Decomposed Low-Rank Adaptation," arXiv preprint arXiv:2402.09353, 2024.

[17] Q. Zhang, M. Chen, A. Bukharin, N. Karampatziakis, P. He, Y. Cheng, W. Chen, and T. Zhao, "AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning," arXiv preprint arXiv:2303.10512, 2023.

[18] Hugging Face, "Auto classes: AutoModelForSequenceClassification," Hugging Face, [Online]. Available: https://huggingface.co/docs/transformers/model_doc/auto#transformers.AutoModelForSequenceClassification. [Accessed: Feb. 9, 2025].

[19] Hugging Face, "Auto classes: AutoModelForTokenClassification," Hugging Face, [Online]. Available: https://huggingface.co/docs/transformers/model_doc/auto#transformers.AutoModelForTokenClassification. [Accessed: Feb. 9, 2025].

[20] Hugging Face, "Auto classes: AutoModelForQuestionAnswering," Hugging Face, [Online]. Available: https://huggingface.co/docs/transformers/model_doc/auto#transformers.AutoModelForQuestionAnswering. [Accessed: Feb. 9, 2025].

[21] R. Shuttleworth, J. Andreas, A. Torralba, and P. Sharma, "LoRA vs Full Fine-tuning: An Illusion of Equivalence," arXiv preprint arXiv:2410.21228, 2024.

مدل‌های زبانی چندزبانه در پردازش زبان طبیعی فارسی: مقایسه عملکرد تکنیک‌های تنظیم دقیق

علیرضا قاسمی و جواد سلیمی سرتختی*

گروه هوش مصنوعی، دانشکده مهندسی برق و کامپیوتر، دانشگاه کاشان، کاشان، ایران.

ارسال ۲۰۲۴/۱۲/۰۹؛ بازنگری ۲۰۲۵/۰۱/۱۷؛ پذیرش ۲۰۲۵/۰۲/۱۲

چکیده:

این مقاله به ارزیابی عملکرد روش‌های مختلف تنظیم دقیق برای مدل‌های زبانی چندزبانه در وظایف پردازش زبان طبیعی زبان فارسی می‌پردازد. در زبان‌های کم‌منبع مانند فارسی که با کمبود داده‌های غنی و کافی برای آموزش مدل‌های بزرگ مواجه هستند، انتخاب تکنیک‌های مناسب تنظیم دقیق که از بیش‌برازش جلوگیری کرده و مانع از یادگیری الگوهای سطحی و ضعیف توسط مدل می‌شوند، اهمیت زیادی دارد. هدف اصلی این پژوهش مقایسه کارایی رویکردهای تنظیم دقیق مانندFull-Finetune ، LoRA، AdaLoRA و DoRA در یادگیری مدل و عملکرد وظایف مختلف است. این تکنیک‌ها در سه وظیفه مختلف پردازش زبان طبیعی فارسی شامل تحلیل احساسات، تشخیص موجودیت‌های نامدار و پاسخ به سؤالات متنی اعمال شده‌اند. برای این منظور، آزمایش‌هایی روی سه مدل چندزبانه مبتنی بر معماری Transformer با ساختارها و مقیاس‌های پارامتری متفاوت انجام شده است BERT-base :چندزبانه (~۱۶۸ میلیون پارامتر) با ساختار تنها رمزگذار ، mT5 small-(~۳۰۰ میلیون پارامتر) با ساختار رمزگذار-رمزگشا و mGPT (~۱,۴ میلیارد پارامتر) با ساختار تنها رمزگشا. هر یک از این مدل‌ها از زبان فارسی پشتیبانی می‌کنند اما تفاوت در ساختار و نیازهای محاسباتی آن‌ها بر کارایی روش‌های مختلف تنظیم دقیق تأثیر می‌گذارد. نتایج نشان می‌دهد که مدل BERT-base چندزبانه با تنظیم دقیق کامل، به طور مداوم در تمامی وظایف از سایر مدل‌ها در معیارهای پایه پیشی می‌گیرد، به ویژه با توجه به چالش‌های خاص این وظایف مبتنی بر تعبیه‌سازی علاوه بر این، روش‌های تنظیم دقیق سبک مانند LoRA و DoRA عملکرد رقابتی بالایی را ارائه می‌دهند، در حالی که به طور قابل توجهی سربار محاسباتی را کاهش داده و در امتیاز کارایی-عملکرد معرفی‌شده در این مقاله، از سایر مدل‌ها بهتر عمل می‌کنند. این مطالعه به درک بهتر روش‌های تنظیم دقیق، به ویژه برای پردازش زبان فارسی، کمک می‌کند و راهنمایی‌های عملی برای به‌کارگیری مدل‌های زبانی بزرگ در وظایف پایین‌دستی در زبان‌های کم‌منبع ارائه می‌دهد.

کلمات کلیدی: تکنیک‌های تنظیم دقیق، مدل‌های زبانی بزرگ در زبان‌های کم‌منبع، مدل‌های زبانی چندزبانه، BERT چندزبانه.