FRI Natural language processing course 2021
Project report

University *of Ljubljana*
Faculty *of Computer and Information Science*

Parameter-Efficient Fine-Tuning of Language Models for Slovene

Camile Lenderling, Manfred González, and Joaquín Figueira

Abstract

This work adapts and evaluates several parameter efficient finetuning techniques (PEFT) for large language models pre-trained on the Slovenian language. It focuses on methods based on LoRA, bias update and prefix tuning. The performance of these methods is assessed with SloBENCH, and evaluation framework for the Slovenian language, and focuses on natural language understanding (NLU) and machine translation tasks (MT). Results are compared with previous submissions to the evaluation framework to determine the relative effectiveness of PEFT methods in a Slovenian linguistic context.

Keywords
LLM, LM, PEFT, NLP, Slovene, LoRA, BitFit, p-tuning

Advisors: Slavko Žitnik

Introduction

Large language models and, by extension, language models (LM) in general are becoming ubiquitous in the current technological landscape. As such, the amount of tasks and applications that are being performed by, or with the help of these models is continuously growing. Until recently, LMs were built for specific taks, but with the advent of LLMs this has become both unnecessary, as models are complex enough now that they can perform multiple tasks well out-of-the-box, and unfeasible, as building these models takes an enormous amount of computational resources. However, if one is to achieve the optimal performance on a given task or expand the capabilities of a LM, some adaptations of the model are in order.

These adaptations usually come in the form of a process called finetuning, in which the parameters of an already pre-trained language model (PLM) are fitted for the specific task we want to achieve. Nonetheless, given the massive amount of parameters in modern LLMs, full finetuning (FFT) of the weights of a model is usually too computationally prohibitive for most applications. As a result, in the recent years many techniques have been developed to efficiently tune LMs, focusing specifically in reducing the amount of parameters that actually have to be trained (a comprehensive review of this techniques can be found on [1]). These are usually referred to with the umbrella term parameter efficient fine-tuning techniques (PEFT) and are mainly based on the Lottery Ticket Hypothesis, which states that "large models are needed in pretraining only to induce (in high probability) the existence of sub-networks initialized with the correct inductive bias for learning.". In other words, that PLMs already contain the capacity and linguistic "knowledge" to perform a wide variety of tasks, but this abilities need to be accentuated through the proper learning stimulus that induces the use of the correct sub-networks when performing a particular task.

The purpose of this work is precisely to adapt and evaluate several PEFT techniques with a focus on the Slovenian language. To cover a wide range of approaches the analyzed techniques will span different families: LoRA[2] and its derivatives, bias update[3] and soft prompt based techniques. As a base model for analysis the SloBERT LM will be used, which is a BERT model optimized for Slovenian. For performance assessment the techniques will be assessed on multiple NLP tasks focused on natural language understanding (NLU) and machine translation (MT) using some of the NLP benchmarks provided in the Slobench Slovenian evaluation framework, mainly the Slovene SuperGLUE and Machine Translation (SLO - ENG) benchmarks. A final analysis will also be provided comparing the results of the fine-tuned models with the metrics from previous submissions to Slobench of the same benchmarks.

Methods

LoRA

The field of adapting large language models (LLMs) for specific tasks has seen significant advancements with the introduction of Low-Rank Adaptation (LoRA). Hu et al. [2] pioneered this approach by freezing the pre-trained model weights and integrating trainable rank decomposition matrices into the Transformer architecture. This method not only substantially reduces the number of trainable parameters required for downstream tasks but also lessens the GPU memory requirements, thereby enabling comparable or superior model quality with notable efficiency. The provision of a package to facilitate the integration of LoRA with PyTorch models, including implementations for popular models like RoBERTa, DeBERTa, and GPT-2, marks a significant contribution to the field.

Further enhancing the parameter efficiency of fine-tuning PLMs, Zhang et al. [4] introduced IncreLoRA, an incremental parameter allocation method that judiciously adds trainable parameters based on the importance scores of each module. This approach, distinguished from structured pruning methods, not only improves parameter efficiency but also incorporates early learning and restart warmup techniques to bolster training effectiveness and stability. The method demonstrated superior parameter efficiency and model performance, particularly in low-resource settings, through rigorous experiments on the GLUE benchmark.

On the deployment front, Xu et al. [5] proposed the Quantization-Aware Low-Rank Adaptation (QA-LoRA) algorithm, aimed at the efficient deployment of LLMs on edge devices. By introducing group-wise operators, QA-LoRA enhances the quantization flexibility while streamlining adaptation, enabling the integration of quantized LLM and auxiliary weights without compromising accuracy. This method stands out by allowing for low-bit inference directly, overcoming the limitations of previous approaches like QLoRA and facilitating faster model deployment on resource-constrained devices.

Lastly, the scalability and efficiency of serving multiple LoRA adapters derived from a base model have been addressed by Sheng et al. [6] through the introduction of S-LoRA. This system, designed for scalable serving, significantly improves throughput and the capacity to serve numerous task-specific fine-tuned models by employing a unified memory management approach and optimized computation strategies. Complementing these efforts, Gao et al. [7] introduced MoE-LoRA with Layer-wise Expert Allocation (MoLA), which optimizes the allocation of LoRA experts across different layers of the Transformer model, thereby enhancing model efficiency and performance across various NLP tasks. These advancements collectively signify a leap forward in the efficient adaptation, deployment, and serving of LLMs, paving the way for broader application and innovation in the domain.

Bias update

BitFit[3] is the first recorded technique to have implemented a sparse-finetunning of LMs using only the bias parameters and it falls under the category of partial fine-tuning according to Xu et al.'s taxonomy[1]. The main mechanism of the technique is simple: during fine-tuning an LM on a particular task update only the bias parameters of the model's encoder layers. These parameters account for only a small fraction of all the parameters of the model (0.1% in the case of BERT). Additionally, the authors found that fitting only a small subset of these bias parameters (mainly the bias of the query encoders of the attention heads and the biases of one of the layers of the MLP inside of the encoder layer) leads to almost no performance drop and modifies only 0.04% of the parameters. According to Xu et al. [1] the technique achieves relatively great results with only a fraction of the memory footprint of other PEFTs. These promising results, combined with the existence of multiple pre-packaged implementations of the technique, led us to choose it as one of the subjects of analysis of this work.

The findings of the initial BitFit paper were further expanded in the works of Lawton et al.[8] using neural architecture search (NAS), more specifically, iterative network pruning. The core idea of the method is to iteratively fine-tune the model using BitFit[1] and then prune it's bias parameters according to a criteria based on the first order approximation of the loss that results from eliminating certain parameters from the network. The authors found that the resulting network architectures could maintain good performance with a large portion of their bias parameters pruned, further solidifying the findings in the initial BitFit paper that only a relatively small number of bias parameters are responsible for the fine-tuned performance. Unfortunately, there is no code or implementation freely available to replicate the results of this technique in our work.

Soft prompts

Soft prompts, represent various methods to efficiently adapt LLMs for down-stream tasks without altering the underlying model architecture or weights. This technique involves appending a sequence of tunable tokens, or ”soft prompts,” to the input of the model. During fine-tuning, these tokens are optimized to guide the model towards generating the desired task-specific output. Two prominent methods for implementing soft prompts are prompt tuning and prefix tuning.

Prompt tuning

Prompt tuning introduces a set of ℓ learnable tokens (soft prompts), denoted as $P = \{P_1, P_2, \dots P_\ell\}$, and concatenates these tokens to beginning of the input to the model $X \in \mathbb{R}^{n \times d}$ to form $\hat{X} \in \mathbb{R}^{(n+\ell) \times d}$. Throughout the fine-tuning process, only the parameters associated with the prompt tokens P are adjusted via gradient descent, while the pre-trained model parameters are kept fixed. Hence, the length of the prompt

[1]The authors also fine-tuned the models using LoRA, but that falls outside of the scope of this analysis.

and the dimensionality of the token embeddings determine the parameter cost for fine-tuning. [9]

Prefix-tuning

Prefix-tuning introduces the idea of appending a set of soft prompts $P = \{P_1, P_2, \ldots P_\ell\}$, not to the input layer but to the hidden states within the multi-head attention layers of the model. This is different from prompt tuning, which concatenates soft prompts directly to the input. To promote stability during training, a feed-forward network (FFN) is used to parameterize these soft prompts. During fine-tuning, two distinct sets of prefix vectors, $\hat{P_K}$ and $\hat{P_V}$, are concatenated to the attention layer's original key (K) and value (V) vectors, respectively. Hence, the only parameters that require optimization are those of $\hat{P_K}$, $\hat{P_V}$, and the FFN. Once the model is fine-tuned, the FFN is no longer needed, and only the optimized key and value prefix vectors are kept for model inference. [10]

1. Proposed Methodology

Our proposed methodology is as follows (sketch):

1. Select and preprocess datasets from the SloBench framework.
2. Format data for NLP tasks using appropriate tokenizers for SloBERTa and other multilingual LMs.
3. Establish performance baselines for the untuned SloBERTa and other selected multilingual LLMs.
4. Using HuggingFace's PEFT library, apply various prominant PEFT techniques such as LoRa, Bias Update, and Soft Prompts to fine-tune the LMs on different downstream tasks.
5. Use SloBench to evaluate fine-tuned models, comparing performance across different PEFT methods and NLP tasks.

2. Results

This section outlines the benchmarking results for various Slovene and multilingual large language models (LLMs) applied to different NLP tasks, including Sentiment Analysis (SA), Named Entity Recognition (NER), Dependency Parsing (DP), Natural Language Inference (NLI), and Coreference Resolution (CR). We evaluated the performance using popular Parameter-Efficient Fine-Tuning (PEFT) methods to determine their effectiveness across these different langauage processing tasks.

2.1 Natural Language Inference (NLI)

NLI involves determining whether a "premise" sentence logically supports, contradicts, or is neutral to a "hypothesis" sentence. Using the SI-NLI dataset, this study uses 5,937 Slovene sentence pairs from the Slovenian reference corpus ccKres, each manually annotated with labels of "entailment," "*contradiction*," or "*neutral*." For computational processing, the experiments were performed on an NVIDIA GeForce RTX 3050 Ti[2] with PyTorch and the Hugging Face Transformers library.

Table 1. Performance Comparison on SI-NLI

Model	PEFT Method	Accuracy	F1 Score	Training Time (m)
SloBERTa	LoRa	69.7%	69.7%	32.6
SloBERTa	Prefix Tuning	36.7%	34.2%	24.6
SloBERTa	IA3	31.5%	21.7%	24.9
BBMU	LoRa	50.8%	50.7%	26.3

2.2 Name Entity Recognition (NER)

The objective of NER is to identify specific key elements in the words of a sentence. For instance, an NER model may be able to identify when a text mentions or refers to a person, a particular location, an organization, etc. In this experiment we use the SSJ500K training dataset to benchmark several PEFT techniques in the same LM as in the previous task: the slovene language specific *SloBERTa* and the multilingual Bert-Base-Multilingual-Uncased (BBMU). The corpus contains about 500.000 manually annotated tokens for token-level classication in the slovene language. Furthermore, it contains several subsets for different tasks; we used the NER subset which contains roughly 9,500 manually annotated sentences using the entities of the CoNLL-2003 shared task. The experiment was performed using kaggle's computing resources with a P100 GPU and using the libraries Hugging Face PEFT, transformers and PyTorch.

Table 2. Performance Comparison on SSJ500K's NER subset The table shows the result of training the *SloBERTa* and *Bert-Based-Multilingual-Uncased* (BBMU) models using different PEFT techniques and Full Fine-Tunning (FFT) as a baseline in the NER subset of the SSJ500k corpus. All results where generated using three training epochs with batches of 16 samples.

Model	PEFT Method	Precision	F1 Score	Training Time (s)
SloBERTa	FFT	85.1%	87.3%	180.8
SloBERTa	LoRA	85.8%	87.9%	145.2
SloBERTa	Prefix Tuning	80.3%	83.7%	114.85
SloBERTa	IA3	82.5%	85.6%	123.3
BBMU	FFT	80.8%	82.5%	250.2
BBMU	LoRA	79.3%	80.7%	184.6
BBMU	Prefix Tunning	74.4%	80.3%	153.8
BBMU	IA3	68.1%	74.5%	160.6

The results show decent performance of almost all methods with the *SloBERTa* model, with LoRa even surpassing full fine-tuning with the same amount of training epochs and batch size. Furthermore, all methods reduced considerably the training time over full fine tuning. In fact, Prefix Tuning

[2]We will conduct the experiments on the cluster next.

almost halfs the training time of FFT for both models (training time is 63% of FFT for *SloBERTa* and 61.5% for BBMU). Furthemore, of all techniques LoRA performs by far the best. This is expected as LoRa has proven efficacy. Regarding the other methods, it's surprising that such a simple method as Prefix Tuning can outperform IA3 by such a wide margin in one of the models, specially considering IA3 is intended to be an improvement on LoRA. In further experiments we'll seek to verify this finding by tuning the hyperparameters of the techniques more thoroughly.

Regarding the performance comparison between the two models, it seems *SloBERTa* performs better overall in all experiments. This is to be expected given *SloBERTa* was specifically trianed to perform well in slove. Additionally, it seems that it responds better to PEFT techniques, hinting at the fact that these techniques may not be as efficient when some cross-lingual transfer learning is required, such as the one required to fine-tune the BBMU model.

2.3 Dependency Parsing

Using a different subset from the previous SSJ500k dataset we developed further experiments to evaluate the techniques' performance in Depedency Parsing. In particular, we wanted to predict for each word, the relation to its root. The subset we used contains about 11 400 synthetically annotated sentences with their corresponding dependency tree. Each word has assigned both a relation to its head or root and an index identifying the mentioned root. We used the same experimental setup as in the previous NER task. The results can be seen in table 3.

Table 3. Performance Comparison on SSJ500K's Depedency Parsing subset The table shows the result of training the *SloBERTa* and BBMU models using different PEFT techniques and Full Fine-Tunning (FFT) as a baseline in the Dependency Parsing task subset of the SSJ500k corpus. All results where generated using three training epochs with batches of 32 samples.

Model	PEFT Method	Precision	F1 Score	Time (s)
SloBERTa	FFT	93.1%	93.1%	218.6
SloBERTa	LoRA	94.0%	94.0%	174.34
SloBERTa	Prefix Tuning	91.1%	91.0%	150.9
SloBERTa	IA3	92.1%	92.0%	157.5
BBMU	FFT	90.3%	90.3%	305.5
BBMU	LoRA	90.6%	90.6%	235.4
BBMU	Prefix Tunning	85.7%	85.3%	207.1
BBMU	IA3	86.0%	85.6%	211.7

In this task we observe similar results as in the previous one. As before LoRA achieves by far the best results in both models, surpassing FFT with significantly lower training times. However, it seems Prefix Tuning never performs better than IA3. The reason for this may be that the task is relatively more complex, requiring the model to discover deeper relationships between the tokens than in NER, so a more complex approach that focuses on enhancing the attention mechanism of the models may be a better strategy to tackle it. Overall, it seems that PEFT techniques perform quite well, specially with the *SloBERTa* model, obtaining similar or even superior results with significantly reduced training times.

Regarding the comparison between the models, similar results are observed as with the NER task: the performance is better in the slovene specific *SloBERTa* and the PEFT methods are less effective in the multilingual model.

2.4 Recognizing Textual Entailment (RTE)

In this section, we analyze how LoRA fine tunning influnces over the F1 Scores over the SuperGlue Human based RTE dataset.

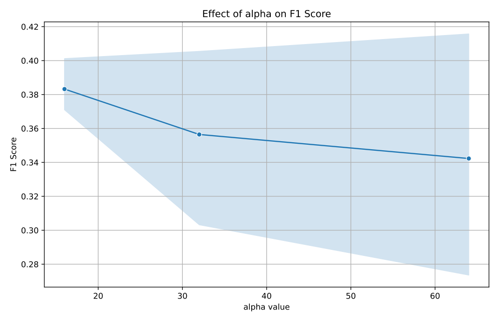
In this experiment, the *SloBERTa* model is fine-tuned on the RTE dataset utilizing the Low-Rank Adaptation (LoRA) approach to explore the impact of various hyperparameters on the model's performance, particularly focusing on the F1 score. The experiment iteratively tests combinations of three key hyperparameters: r, α, and *weight decay*.

- r **(rank):** This parameter determines the rank of the adaptation in LoRA, which affects the model's capacity to learn new patterns without significantly increasing complexity. Values explored were [8, 16, 32], enabling an investigation into how increasing the rank influences learning capacity and generalization.

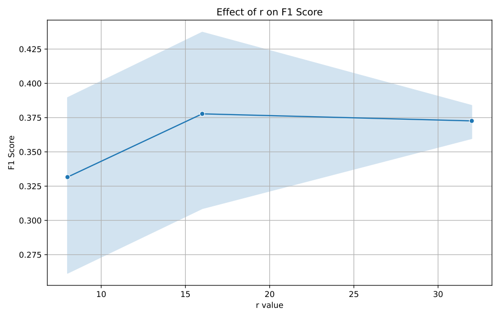
- α **(scaling factor):** This controls the scaling of the updates in LoRA, impacting how aggressively the model adapts to the new dataset. Tested values were [16, 32, 64], providing insights into finding a balance between overly subtle and overly aggressive updates.

- **Weight Decay (regularization):** Used to prevent overfitting by penalizing larger weights, with values set at [0.01, 0.1, 0.2]. This parameter explores how varying levels of regularization affect overfitting and model performance on validation data.

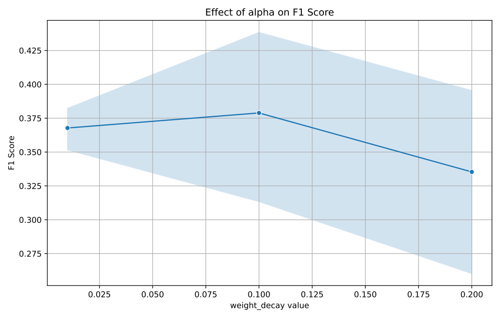
The function used for this experiment orchestrates a grid search over these parameters, training the *SloBERTa* model for each configuration and logging the F1 score to identify the optimal setup. The configuration leading to the highest F1 score is noted as the best, reflecting the most effective balance of learning capacity, adaptability, and regularization for this task. Metrics and configurations are saved, and the best model setup is printed, showcasing the impact of tuning LoRA parameters on model efficacy in a sequence classification task on the RTE dataset. The results can be observed in the figure 1 where the best parameters to be used under this configuration are r=16, α=64, weight decay=0.1. The best F1 Score achieved with *SloBERTa* was about 0.52 over this dataset, the future work will be focused on compare more models to see which one performs better in this dataset that represents a challenge since it is small and in Slovene.



(a) Influence of α o in the F1-Score



(b) Influence of *r* or in the F1-Score



(c) Influence of weight decay on thein F1-Score

Figure 1. Different LoRA fine- tunning configurations, results over the F1 Scores using the SuperGlue Human based RTE dataset

References

[1] Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment, 2023.

[2] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[3] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models, 2021.

[4] Feiyu Zhang, Liangzhi Li, Junhao Chen, Zhouqiang Jiang, Bowen Wang, and Yiming Qian. Increlora: Incremental parameter allocation method for parameter-efficient fine-tuning, 2023.

[5] Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, Xiaopeng Zhang, and Qi Tian. Qa-lora: Quantization-aware low-rank adaptation of large language models, 2023.

[6] Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, Joseph E. Gonzalez, and Ion Stoica. S-lora: Serving thousands of concurrent lora adapters, 2023.

[7] Chongyang Gao, Kezhen Chen, Jinmeng Rao, Baochen Sun, Ruibo Liu, Daiyi Peng, Yawen Zhang, Xiaoyuan Guo, Jie Yang, and VS Subrahmanian. Higher layers need more lora experts, 2024.

[8] Neal Lawton, Anoop Kumar, Govind Thattai, Aram Galstyan, and Greg Ver Steeg. Neural architecture search for parameter-efficient fine-tuning of large pre-trained language models, 2023.

[9] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning, 2021.

[10] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation, 2021.