Fine-tuning Large Language Model without distorting its weights. Replaceable Small Language Model

Anonymous ACL submission

Abstract

If Small Language Model (SLM) and Large Language Model (LLM) both have the same vocabulary of tokens, SLM can follow the LLM, making output predictions of the unified Language Model better correspond to the new goal. When we do Supervised Learning with a Fine-Tuning dataset and updating only SLM weights, we adjust only SLM and not LLM, and one can replace SLM to fit different tasks

Methods and Materials

"English Classic Writers" and "All Shakespeare" text corpora, data type - float16-mix, token size – 2 symbols, vocabulary size – 1092 tokens, vocabulary embedding - 1024, token embedding – 768, time intervals (context block) - 430, layers – 17, number of parameters 36.3M, learning rate – 3e-5, batch size – 64, pytorch library, videocard - NVidia GPU 3060 RTX.

1 Introduction

This work is an early prototype and intended for research groups or small companies that want to keep one shareable pre-trained Large Language Model(Douglas, 2023)(Song et al., 2023)(Wang et al., 2023) across different departments, but adjust it to different tasks, not re-training it. Recently there was some concern about fine-tuning a Large Language model, as the weights of the original model were adjusting to new data in an aggressive manner(Jiang et al., 2020)(Zhang et al., 2022). One of the approaches according to (Chen et al., 2020) is to keep a smaller distance between the original weights and fine-tuned weights or to replace some weights with the original weights during training for better generalization.(Lee et al., 2019) Similar work to ours have been done recently by Xu et al (Xu et al., 2021), where they take a subset of parameters, a Child Network, inside LLM and mask weights of the rest parameters to eliminate them from back propagation in the original LLM. They call this process as giving birth to the Child Network. Our approach in this sense is different and simpler as we use pre-trained LLM (that can reside on some virtual server occupying several Graphical Processing Units) to help to train pre-initialized replaceable Small Language Model(Schick and Schütze, 2020) for task specific purposes.

This can be thought as Knowledge Distillation(Hinton et al., 2015) but where the student model cannot work without the teacher, you need to add it or replace it in order to meet task specific goals.

2 Large Language Model

2.1 Abbreviations

LLM - Large Language Model
SLM - Small Language Model

2.2 Simplified Dataset and Token Embedding table

We took Classic Literature in ASCII from Kaggle(Scott, 2014), merged different English Writers together in a single text corpus, derived 2 symbol tokens from it with raw vocabulary size of approximately 4000, and then analyzed the space it occupies due to the Token Embedding table being put on a single GPU device. Then we cleaned and processed united text corpus expanding contractions, removing urls, xml tags, meta information in square brackets, digits, etc, making the number of unique tokens equal to 1092, and 39.3M tokens in total in the united corpus.

2.3 Simplfied LLM

In parallel work(Timur, 2023) we simplified the Attention mechanism to the Feed Forward neural network operating in the Time domain with future nodes shut down. It was a comparatively small LLM with vocabular embedding size - 1024, token embedding size - 768, time intervals (context

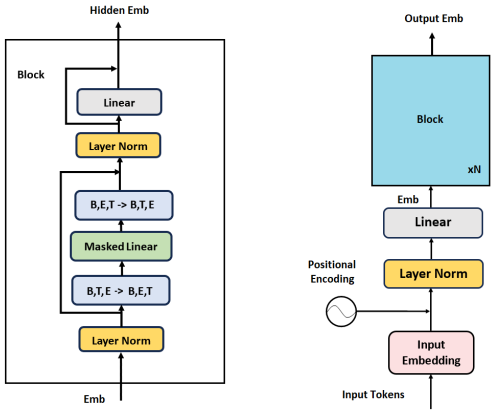

Figure 1: Unified LLM and SLM Architecture

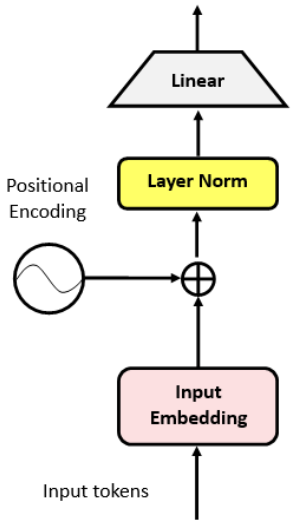

Figure 3: Token and Position Embedding unification layer

block) - 430 and 17 sequential layers with a total number of parameters 36,3M it was fitted in NVidia 3060 RTX with 12Gb RAM. We trained it on our "English Classic Writers" text corpus to the point where the model produced sensible sentences with context understanding of the past. It took approximately 500,000 iterations with batch size of 64 and learning rate of 3e-5.

3 Replaceable Small Language Model

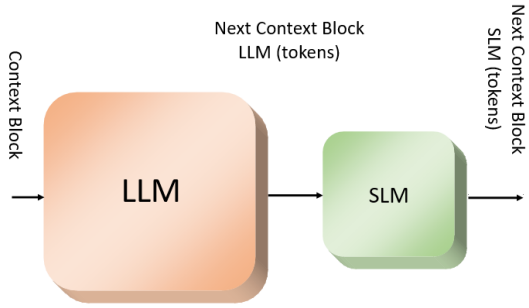

Figure 2: Unified LLM and SLM Architecture

Our SLM has the following architecture: vocabulary Embedding size - 1024, Token Embedding size 384, time intervals (Context block) - 370, number of sequential layers - 7 with 12M parameters in total fit with LLM on the same GPU.

Our SLM takes the next tokens that were predicted by the LLM, and processes them through its own Token and Position Embedding tables and sequential layers producing task specific next tokens. If the Vocabulary embedding size of SLM and LLM are equal - which was the case - we do not need to start from scratch and to train its Token Embedding and Position Encoding tables, we can just copy them. This is due to the presence of the linear layer that projects sum of Vocabulary Embedding and Position Encoding into one uniform vector or Token Embedding prior to sequential Transformer Blocks.

4 Training

Throughout the experiments we tried different architectures for SLM, e.g. SLM encoder and SLM decoder, but the SLM decoder that follows LLM decoder architecture produced the most promising results. There were high concerns about calculating *derivatives* as logits that pass through sampling from the categorical distribution cannot produce any gradients. But all worries were dispelled, cause in the decoder SLM we do not train the LLM, its output tokens only serve as inputs for the SLM: an optimizer takes parameters of SLM, and does backward propagation updating them during training, while LLM is put in an evaluation mode.

It is important to mention that usually the Language Model is trained by the Context Block(Wang et al., 2023) shifted one token forward. Which was the case for the LLM. But the SLM, which already received the next Context Block from the LLM was not trained with Context Block shifted 2 tokens forward (as should be the case), but instead was trained with next Context block from the Fine-Tuning text corpus: batch of Context block was

taken as an input, and a batch of Context block shifted one token forward was taken as the target. Since the model was much smaller, the idea was not to deduce the next token, but to converge predictions of the bigger LLM to the new target better.

5 Results and Conclusions

Taking into account our computational resources, the simplicity of the models and the length of the Context Block we did not aim to beat state-of-the-art models. Instead we wanted to showcase a possible architecture of using replaceable Small Language Model Encoder for different tasks.

We believe that represented SLM outputs can be further improved with careful prompting(White et al., 2023) and with Reinforcement Learning with Human Feedback(Christiano et al., 2017), but more computational resources are needed for the latter. We used simplified version of Attention mechanism where cross-attention(Chen et al., 2021)(Gheini et al., 2021) were not considered.

Limitations

While the Token Embedding size was 768 and the output of the model was 1092 to suit token numbers in vocabulary, there is respective necessity to increase embedding size (width) to 2048-4096 to better suit the state-of-the-art vocabulary's dimension of 50k-100k tokens. To increase number of time series (length) is also important as well as to increase number of layers (depth). So width, length and depth play a vital role. With only Feed Forward architecture and with a single embedding vector, there is one unsolved drawback: we need to re-think the cross-attention mechanism.

Ethics Statement

This work offers a solution that would help the scientific and industrial community reduce the size and ease of computation of large linguistic models that do not require cross-attention mechanism, which may ultimately have an environmental impact.

Acknowledgements

This work was done by means of computational resources of the Department of Computer Algorithms and Artificial Intelligence, University of Szeged. The authors are the *Stipendium Hungaricum* and *Hungary State Scholarship* holders. We are grateful for the opportunity that was provided.

References

Chun-Fu Richard Chen, Quanfu Fan, and Rameswar Panda. 2021. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 357–366.

Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7870–7881, Online. Association for Computational Linguistics.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Michael R Douglas. 2023. Large language models. *arXiv preprint arXiv:2307.05782*.

Mozhdeh Gheini, Xiang Ren, and Jonathan May. 2021. Cross-attention is all you need: Adapting pretrained Transformers for machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1754–1765, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, Online. Association for Computational Linguistics.

Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. 2019. Mixout: Effective regularization to finetune large-scale pretrained language models. *arXiv preprint arXiv:1909.11299*.

Timo Schick and Hinrich Schütze. 2020. It's not just size that matters: Small language models are also few-shot learners. *arXiv preprint arXiv:2009.07118*.

Jason Scott. 2014. Classic literature in ascii. Data retrieved from Kaggle, https://www.kaggle.com/datasets/mylesoneill/classic-literature-in-ascii.

Lei Song, Chuheng Zhang, Li Zhao, and Jiang Bian. 2023. Pre-trained large language models for industrial control. *arXiv preprint arXiv:2308.03028*.

Ishuov Timur. 2023. Nano fft model. Data retrieved from github, https://github.com/timurgepard/nanoFFT.

Haifeng Wang, Jiwei Li, Hua Wu, Eduard Hovy, and Yu Sun. 2023. Pre-trained language models and their applications. *Engineering*, 25:51–65.

Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C Schmidt. 2023. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*.

Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. 2021. Raise a child in large language model: Towards effective and generalizable fine-tuning. *arXiv preprint arXiv:2109.05687*.

Haojie Zhang, Ge Li, Jia Li, Zhongjin Zhang, Yuqi Zhu, and Zhi Jin. 2022. Fine-tuning pre-trained language models effectively by optimizing subnetworks adaptively. *Advances in Neural Information Processing Systems*, 35:21442–21454.