Towards Parallelising Pre-Trained Transformers

Vincenzo Scotti[0000−0002−8765−604X] (✉) and
Mark James Carman[0000−0001−6575−9737]

DEIB, Politecnico di Milano, Via Ponzio 34/5, 20133, Milano (MI), Italy
vincenzo.scotti@polimi.it mark.carman@polimi.it

Abstract. In this paper, we present our preliminary findings on the parallelisation of pre-trained Large Language Models to enhance their performance by exploiting alternative pathways within the stack of features learned by Transformer neural networks. The objective of this study is to investigate whether the independence and compositional flexibility of these pathways can be leveraged to better extract the information embedded in the models. By reordering and parallelising the feature stack, we aim to virtually extend the depth of the network, thereby providing more space for reasoning. Our experiments involved evaluating on the H6 benchmark the parallelised pre-trained models and the fine-tuned models, adapted for parallelisation using Low-Rank Adaptation. The results we collected highlighted that even without fine-tuning, parallelising a model can help improve certain tasks and that fine-tuning can further enhance these gains. These promising results suggest that the parallelisation of pre-trained Large Language Models is a viable approach to improving model performance and that further exploration and refinement of this technique could yield significant benefits.

Keywords: LLM · Transformer · Parallel Transformer

1 Introduction

Large Language Models (LLMs) have emerged as powerful generative text models capable of handling many types of information. Besides their text generator nature, LLMs can be viewed as *lossy compression algorithms* that have memorised vast amounts of information from their pre-training data [3]. However, due to the lossy nature of this compression, the stored information is not always accessible or accurate, and the recall of specific information can vary based on the query formulation [8]. Therefore, these LLMs are less reliable than expected.

LLMs are built upon the *Transformer architecture* [12], which has been revised many times to accommodate different improvements. In this context, we are interested in (sub-)layer independence and transformation compositionality. For example, models of the PaLM [1] family have started exploring the concept of *parallel layers* on large scales: the sub-layer components –*self-attention* and *Feed-Forward Neural Networks* (FFNN)– are made independent and parallelisable within each layer. Differently, other models, like the Universal Transformer [2], have shown how iteratively refining the hidden representations, enabling the same transformations to yield progressively better results. These examples

highlight the potential for (sub-)block independence and iterative refinement, which inspired our exploration of Transformer layers parallelisation.

In this paper, we work on parallelising transformer models to better exploit the knowledge embedded within LLMs. Our preliminary results suggest that parallelisation can yield performance improvements, even without fine-tuning. These findings open the door to further investigations into incorporating parallelisation during training to possibly learn more independent and composable features.

The rest of this paper is structured as follows. In Section 2, we detail our methodology for parallelising Transformer models and the intuition for expected improvements. In Section 3, we describe our experimental setup, including benchmarks and fine-tuning processes, followed by the presentation of our results. Finally, in Section 4, we provide concluding remarks and suggest future directions.

2 Parallelising the Transformer

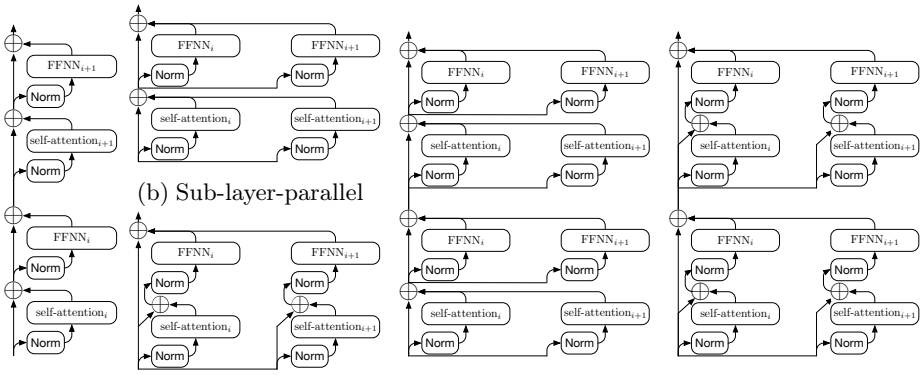

Fig. 1: Visualisation of the different alternatives for Transformer parallelisation (parallelisation rate is two) compared with the base architecture

We designed our parallelisation approach starting from the base Transformer model: a stack of *Transformer layers* that alternates between self-attention and FFNN sub-layers (see Figure 1a) writing to a residual stream. The main idea behind parallelisation is to exploit either the potential independence of features, by running these (sub-)layers in parallel, or the existence of alternative feature dependencies, by iterating through parallelised blocks multiple times. In this context, we define the parallelisation rate as the number of (sub-)layers running concurrently.

Similarly to parallel layers [1], to exploit possible transformations independence, we initially resorted to sub-layer parallelisation taking self-attention

components of consecutive layers, computing their output in parallel from the same input and summing the outputs to the residual stream before proceeding to do the same with the corresponding FFNN components (see Figure 1b). This technique reduces the effective depth of the Transformer. Alternatively, given possible intra-layer dependencies, we consider layer parallelisation (see Figure 1c), where multiple full layers (each comprising self-attention and its FFNN) are fed with the same input, and their outputs are then summed together. Two main aspects need consideration with these parallelisations: (i) the stacked features may have dependencies across layers, which could be disrupted by parallel processing, and despite the use of averaging, summing features in the residual connections may disrupt the feature scales, potentially causing instability in the model. (ii) find the best possible rank allocation allowed by the available memory.

To address the dependency issues, we iterate multiple times through the parallel layers (see Figures 1d and 1e), effectively recovering the original sequentiality and exploring different orderings of the transformations. This approach can be beneficial when information extracted later in the stack is useful for earlier pattern recognition. Additionally, this method virtually increases the network's depth, providing more room for internal reasoning. Given that LLMs contain vast amounts of information, our hypothesis is that alternative ordering of transformations for a given sequence may help recall the information needed to complete the sequence, such as the knowledge required to answer a question.

To prevent scale disruption, we considered, optionally, averaging the outputs of the parallel layers before adding the result to the residual connection rather than summing them directly. This helps maintain feature scale consistency and stability.

3 Experiments and Results

Table 1: Parallelisation configurations

Parallelisation Parameter	**Configuration ID**															
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Rate	2	2	2	2	2	2	2	2	4	4	4	4	4	4	4	4
Level[a]	SL	SL	SL	SL	L	L	L	L	SL	SL	SL	SL	L	L	L	L
Iterative	✗	✗	✓	✓	✗	✗	✓	✓	✗	✗	✓	✓	✗	✗	✓	✓
Averaged	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓

[a] Parallelisation level can be Sub-Layer (SL) or Layer (L)

For our evaluations, we considered the *H6* benchmark [10] from HuggingFace [13], which encompasses a diverse range of tasks, including *Question Answering* (QA) and mathematics. These benchmarks are designed to test a diverse array of abilities in LLMs. We evaluated medium-scale LLMs: Gemma 7B [10], Llama 3

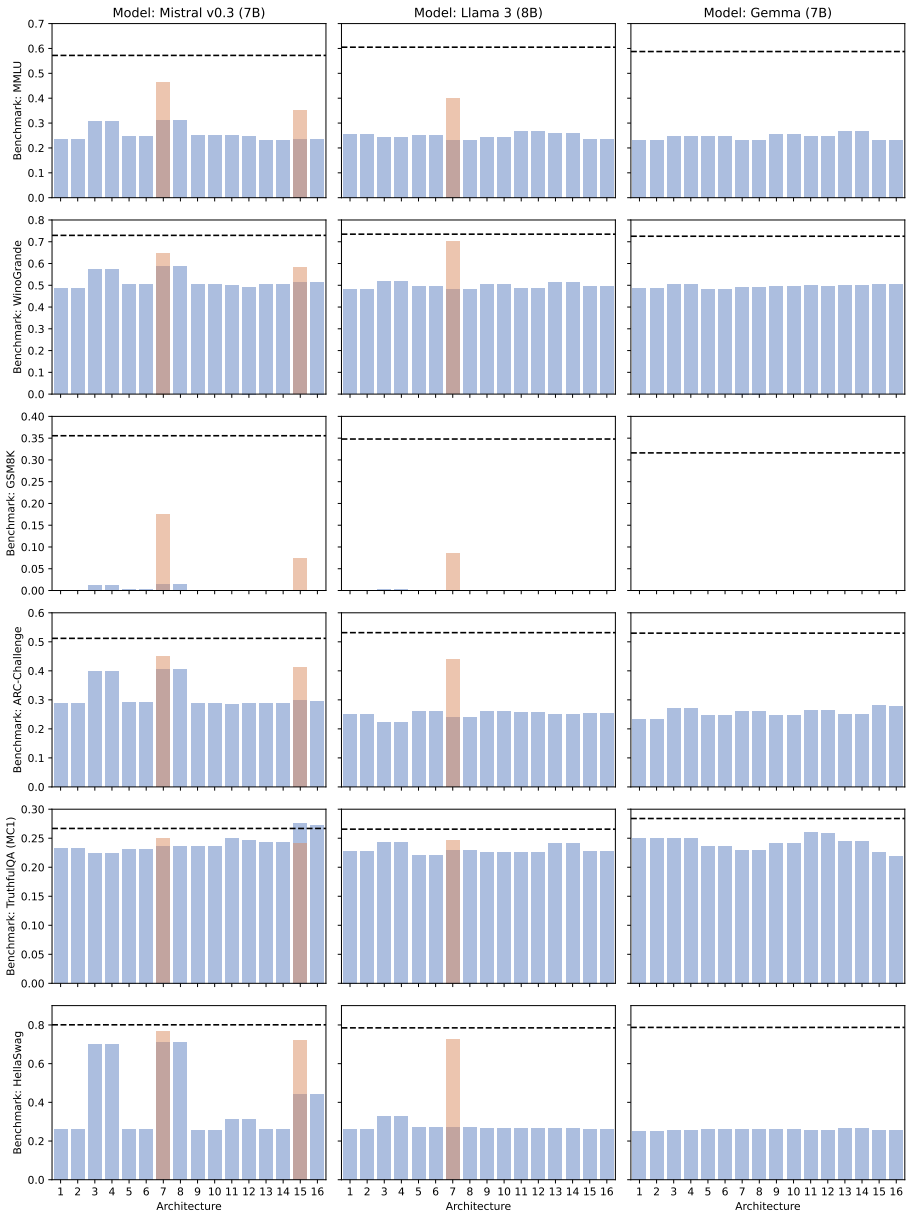

Fig. 2: Results of parallelised architectures on the H6 benchmark (blue bars are the results of parallelised pre-trained models, orange bars are the results of parallelised fine-tuned models, the dashed line corresponds to the results of the base model)

8B [11] and Mistral 7B v0.3 [7] using different parallelisation approaches based on those we presented in Section 2. Additionally, we fine-tuned only a subset of the configurations using *Quantised Low Rank Adapters* (QLoRA) [4] adapters to further enhance its performance and help cope with the parallelisation.

In our evaluations, we compared the base models with various parallelisation configurations, we report the considered configurations in Table 1. We used the *LM Evaluation Harness* [5] tool to automatically run the evaluations (we left the default configurations for all tasks) For fine-tuning, we applied QLoRA with a rank of 16 to all transformations within the transformer. The fine-tuning process was conducted on a subset of the *Open WebText* corpus [6], with validation and testing performed on *WikiText-2* [9], as Open-WebText lacks regular validation and test splits. We fine tuned Mistral 7B v0.3 with layer-level parallelisation, iterative approach and using rates two and four; we Llama 3 8B as well only with a rate of two, to see the effect of fine-tuning at least on one of the other models.

In examining the results (see Figure 2), we noticed that Mistral 7B v0.3 was the only model to improve over the baseline on one of the tasks (TruthfulQA), while maintaining results close to the baseline on other tasks, but still slightly worse. Llama 3 8B and Gemma 7B were more adversely affected by parallelisation. Fine-tuning seems to mitigate some of the issues arising from the abrupt introduction of parallelisation, which was never used during pre-training. However, the results still remain below the baseline. Interestingly, averaging and parallelisation level do not seem to have a significant impact on the results, whereas iterating through the parallelised layers shows a more noticeable effect.

The improvements observed in TruthfulQA align with our intuition, but a deeper analysis is required to confirm these findings. Fine-tuning appears to help the models cope with the abrupt changes due to parallelisation, although we still miss a proper hyperparameter search. In fact, while fine-tuning improved performance on other benchmarks for Mistral 7B v0.3, it also decreased performance above the baseline on TruthfulQA. Additionally, we did not perform prompt tuning, which might have influenced the results, as seen in the original Gemma 7B experiments [10]. Our primary focus at this stage was to explore the feasibility of parallelisation rather than achieving the best possible performance.

4 Conclusion

In this paper, we presented our preliminary findings on the parallelisation of pre-trained LLMs to enhance their performance by leveraging feature independence or alternative pathways within the Transformer architecture. The proposed parallelisation approaches aim at better exploiting the information memorised during pre-training through better use of learnt patterns inside the Transformer to improve model performance and reliability. Our experiments with various parallelisation configurations, including sub-layer and layer parallelisation, iterative and non-iterative approaches, and output averaging, demonstrated that parallelising these models can yield performance improvements and that iterating through the parallelised layers is crucial for the results. Despite these promising

initial results, fine-tuning with QLoRA adapters did not yet achieve the expected performance improvements, indicating the need for further refinement. Future works will focus on enhancing the fine-tuning process and exploring the potential benefits of incorporating parallelisation techniques during the pre-training phase.

Source code `https://github.com/vincenzo-scotti/parallel_transformers`

Acknowledgments. This paper is supported by PNRR-PE-AI FAIR project funded by the NextGeneration EU program.

Disclosure of Interests. The authors have no competing interests to declare.

References

1. Chowdhery, A., Narang, S., Devlin, J., et al.: Palm: Scaling language modeling with pathways. J. Mach. Learn. Res. **24**, 240:1–240:113 (2023)
2. Dehghani, M., Gouws, S., Vinyals, O., et al.: Universal transformers. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net (2019)
3. Delétang, G., Ruoss, A., Duquenne, P., et al.: Language modeling is compression. CoRR **abs/2309.10668** (2023)
4. Dettmers, T., Pagnoni, A., Holtzman, A., et al.: Qlora: Efficient finetuning of quantized llms. In: Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., Levine, S. (eds.) Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023 (2023)
5. Gao, L., Tow, J., Abbasi, B., et al.: A framework for few-shot language model evaluation (12 2023)
6. Gokaslan, A., Cohen, V.: Openwebtext corpus (2019)
7. Jiang, A.Q., Sablayrolles, A., Mensch, A., et al.: Mistral 7b. CoRR **abs/2310.06825** (2023)
8. Kojima, T., Gu, S.S., Reid, M., et al.: Large language models are zero-shot reasoners. In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022 (2022)
9. Merity, S., Xiong, C., Bradbury, J., et al.: Pointer sentinel mixture models. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net (2017)
10. Mesnard, T., Hardin, C., Dadashi, R., et al.: Gemma: Open models based on gemini research and technology. CoRR **abs/2403.08295** (2024)
11. Meta Llama: Introducing meta llama 3: The most capable openly available llm to date (apr 2024)
12. Vaswani, A., Shazeer, N., Parmar, N., et al.: Attention is all you need. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA. pp. 5998–6008 (2017)
13. Wolf, T., Debut, L., Sanh, V., et al.: Huggingface's transformers: State-of-the-art natural language processing. CoRR **abs/1910.03771** (2019)