Sorted LLaMA: Unlocking the Potential of Intermediate Layers of Large Language Models for Dynamic Inference Using Sorted Fine-Tuning (SoFT)

Parsa Kavehzadeh[2], **Mojtaba Valipour**[1,2], **Marzieh Tahaei**[2],
Ali Ghodsi[1], **Boxing Chen**[2], and **Mehdi Rezagholizadeh**[2]
[1]University of Waterloo
[2]Huawei Noah's Ark Lab
{mojtaba.valipour, ali.ghodsi}@uwaterloo.ca,
{parsa.kavehzadeh, mehdi.rezagholizadeh, marzieh.tahaei, boxing.chen}@huawei.com

Abstract

The rapid advancement of large language models (LLMs) has revolutionized natural language processing (NLP). While these models excel at understanding and generating human-like text, their widespread deployment can be prohibitively expensive. SortedNet is a recent training technique for enabling dynamic inference for deep neural networks. We extend SortedNet to generative NLP tasks, making large language models dynamic without any pretraining and by only replacing standard Supervised Fine-Tuning (SFT) with Sorted Fine-Tuning (SoFT). Our approach boosts model efficiency, eliminating the need for multiple models for various scenarios during inference. We show that using this approach, we are able to unlock the potential of intermediate layers of transformers in generating the target output. Our sub-models remain integral components of the original model, minimizing storage requirements and transition costs between different computational/latency budgets. By applying this approach on LLaMA 2 13B for tuning on the Stanford Alpaca dataset and comparing it to normal tuning and early exit via PandaLM benchmark, we show that Sorted Fine-Tuning can deliver models almost twice as fast as the original model while maintaining performance.

1 Introduction

Large language models are revolutionizing the way we interact with information in today's world Hoffmann et al. [2022], Brown et al. [2020], Penedo et al. [2023], Scao et al. [2022]. New models are continually emerging, demonstrating their capabilities not only in understanding but, more importantly, in generating human-like text. Notably, models such as ChatGPT, LLaMA 2 70B Touvron et al. [2023], and Falcon 180B Almazrouei et al. [2023] have had a profound impact on the applicability of large language models (LLMs). However, deploying these expansive language models can become prohibitively expensive.

Enabling dynamic inference, where the computational resources allocated to a given query vary at inference time, can significantly enhance the practicality of employing such models in real-time scenarios. This enables the use of smaller models when the budget is limited, or latency is critical. It's important to emphasize that, given the substantial number of parameters in these large models, a viable dynamic inference strategy should not need loading different models during inference.

Previous research has explored methods for training dynamic models capable of adapting to evolving resource constraints Cai et al. [2019], Hou et al. [2020], Xin et al. [2020], Fan et al. [2019]. However, existing approaches often rely on complex training procedures or necessitate modifications to the

37th Conference on Neural Information Processing Systems - ENLSP (NeurIPS 2023).

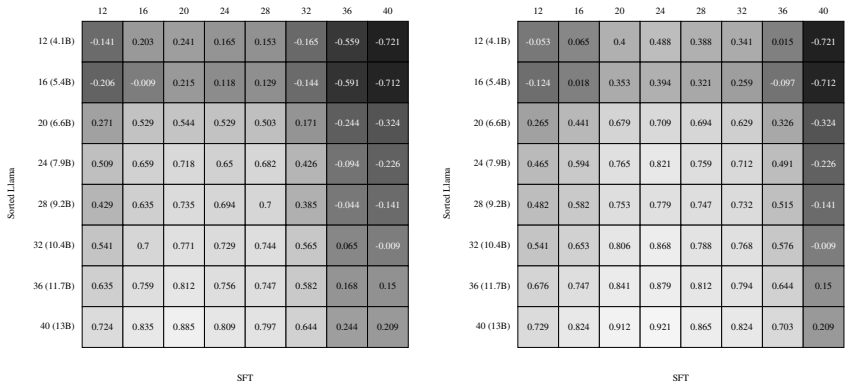

Figure 1: Zero-Shot SoFT vs. Early-Exit SFT (Left) and Zero-Shot SoFT vs. Zero-Shot SFT (Right). Note that for our SoFT method the output prediction layer is shared between all sub-models where as for early-exit a separate prediction head is learned per sub-model making inference inefficient.

original model architecture. SortedNet Valipour et al. [2023] introduces a novel approach to training deep neural networks that leverages the inherent modularity of these networks to construct sub-models with varying computational loads. This method sorts sub-models hierarchically based on their computation/accuracy characteristics, facilitating efficient deployment during inference. Furthermore, it employs an efficient updating scheme that combines random sampling of sub-models with gradient accumulation to minimize the training cost. Consequently, with a single round of training, numerous models can be obtained within a single model.

While the SortedNet approach has primarily been applied to vision and language understanding tasks, given the significant impact of generative language models in today's AI landscape, the efficacy of this method for generative tasks in NLP is of considerable interest. In fact, being able to make a large language model dynamic without the need for pretraining and only at the cost of a round of Supervised Fine-Tuning can open doors to efficient inference of these models without incurring additional expenses associated with common model compression methods like knowledge distillation and pruning, among others. Moreover, since all the resultant models are components of the original model, the storage requirements and the cost associated with transitioning between different computation demands become minimal. Otherwise, managing multiple models for various scenarios during inference becomes impractical.

In this study, we challenge the conventional approach of relying solely on the last layer's contextual embeddings and use Sorted Fine-Tuning (SoFT) in place of Supervised Fine-Tuning to enhance the performance of these models across multiple layers. By doing so, we aim to provide new insights into the efficiency and effectiveness of middle layers in producing high-quality results for specific downstream tasks. Our proposed approach has the potential to optimize the usage of these models, ultimately enhancing their overall performance.

In this paper we employ LLaMA 2 13B and perform both standard Supervised Fine-Tuning (SFT) and SoFT on the Stanford Alpaca dataset Taori et al. [2023], while maintaining equivalent costs for the two approaches. For SoFT, we target 8 sub-models and share the LLM head among them to ensure cost parity. We utilize the PandaLM benchmark Wang et al. [2023] to assess the performance of the sub-models. We show that with sorted tuning we can train many in one LLaMa models with no over-head in fine-tuning. The contributions of this paper can be summarized as follows:

- Extending the SortedNet method for tuning auto-regressive language models for generative tasks by sharing a single LLM head layer among sub-models.
- Generating 8 nested sub-models, ranging from 12 to 40 layers, from LLaMA 2 13B by applying Sorted Fine-Tuning on the Stanford Alpaca dataset and at a cost equivalent to Supervised Fine-Tuning.
- Evaluating the performance of the sub-models and demonstrating the effectiveness of SortedNet tuning in enhancing the ability of intermediate layers for text generation.

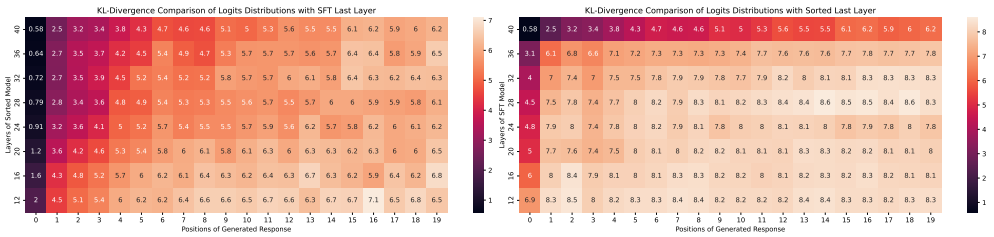

Figure 2: A comparison of sub-models based on output KL-divergence distance of probability distributions.

2 Methodology

This paper focuses on making generative LLMs many-in-one by unlocking the potential of intermediate layers through SortedNet approach Valipour et al. [2023]. Let's consider a language model $f(x;\theta)$ with the parameters θ and the input x. The following is the sorted training procedure:

Forming Sub-Networks First, we need to form the sub-networks of the LLM. For the sake of simplicity and without loss of generality, we focus on the depth-wise sub-networks. Supposed that the sub-network $f_n(x;\theta_n)$ refers to the first n layers of $f(x;\theta)$. In this paper, the language model is considered to be LLaMA 2 13B. Since LLaMA 2 is comprised of 40 layers, we define the sub-networks to be $n \in \mathbf{B} = \{12, 16, 20, 24, 28, 32, 26, 40\}$.

Calculating the Output of Sub-Networks The output of each sub-model will be predicted by using the shared output prediction head from the last layer (original network). Bear in mind that in the LLaMA model, there is an RMSNorm layer Zhang and Sennrich [2019] before the output prediction head. This RMSNorm is added before the shared prediction head of every sub-model.

Objective Function Let $L_n(x;\theta_n)$ be the loss for the n^{th} sub-model for input batch x. To train the network, we define the loss as the summation of the losses of all these sub-models. For the experiments conducted in the papar, $|\mathbf{B}| = 8$. Note that these sub-models have shared parameters through a nested style i.e: $\theta_1 \subset \theta_2 ... \subset \theta_n$.

$$\mathcal{L} = \sum_{n \in \mathbf{B}} L_n(x;\theta_n)/|\mathbf{B}| \tag{1}$$

Training Dataset We utilized the Stanford Alpaca dataset Taori et al. [2023], which includes demonstrations of 52K instruction-following examples.

Evaluation In this paper, in addition to the embedding of the last layer, we evaluate the quality of the embeddings of intermediate outputs spanning from block 1 to n. PandaLM benchmark Wang et al. [2023] is used for the comparison of the output of different sub-models. PandaLM deploys a large language model (Fine-Tuned LLaMA 7b) to judge the quality of generated text from two sources. PandaLM provides a validation set consisting of 170 instructions [1], denoted as T, to evaluate target models for instruction-following tasks. In order to ensure that the order of the models responses does not influence the the judgment of the PandaLM evaluator, we reported an average score under both the Model 1 first and the Model 2 first scenarios. The output of the PandaLM evaluation is the number of wins, denoted as W, the number of losses, denoted as L, and the number of ties in the validation set. The final reported score has been calculated using the following formula:

$$Score = (W - L)/T \tag{2}$$

The final score is a number between -1 and 1, in which 1 represents a strong win rate and -1 means a poor performance of the model.

[1] github.com/WeOpenML/PandaLM/blob/main/data/testset-inference-v1.json

Baseline The primary objective of an LLM in this paper is to follow the provided instructions by a query. Therefore, following the setup of Alpaca Taori et al. [2023], we fine-tuned LLaMA 2 13B on the Stanford Alpaca Dataset with two setups: (1) Regular Supervised Fine-Tuning (SFT) as the baseline, focusing only on the training of the last layer of the network as the common practice in the literature; (2) Sorted Fine-Tuning (SoFT), calculating loss for multiple outputs from layer 12 to layer 40 (last layer) with four intervals, and training multiple models simultaneously as explained in the previous section.

3 Experiments

This section delves into the specifics of the experiments conducted, and the analysis provided to better understand the effect of Sorted Fine-Tuning over the performance of a large language model like LLaMA 2 Touvron et al. [2023].

3.1 What is the effect of sorting information across layers of a generative model?

As mentioned before, we generated responses for all the layers $n \in \mathbf{B}$ for both SFT and SoFT-based trained models. Then we conducted a pair-wise comparison between all the sub-models in the two models using the PandaLM evaluator. As the results suggest in Figure 1, sorted training has a significant impact on unlocking the potential of intermediate layers in generating the desired output.

Sorted LLaMA (aka SoFT) is outperforming regular fine-tuning (SFT) in nearly all layer comparisons by a meaningful margin, as shown through automated evaluation in figure 1 Right.

It might be noted that the Layer 12 performance of SFT is slightly better compared to Layer 12 of Sorted LLaMA. We argue this is happening because the output of early layers in SFT are mostly gibberish and the PandaLM evaluator has not been trained on such data, hence the automatic evaluation results for this layer is not meaningful. As we go to higher layers in SFT, the generated text becomes meaningful, which makes the comparison with the Sorted LLaMA layer counterpart more reasonable.

Moreover, to make the result of SFT layers more meaningful, inspired by Early-Exit Xin et al. [2020], we also tried the scenario in which a separate classifier head is dedicated to all sub-models of SFT. Here, these classification heads are tuned after SFT tuning for one epoch while keeping the base model frozen. Note that this setting suffers from significant memory overhead both during tuning and inference compared to our SoFT method. In fact, the extra number of parameters for early exit is $|B| - 1 \times D \times V$, where $|B|$ is the number of sub-models, D is the hidden size of the model and V is the vocabulary size. For LLaMA 2 13B, this is equivalent to 1B extra parameters.

The results of comparing sorted with early exit is shown in figure 1 Left. Despite having far more parameters early-exit SFT under-performs SoFT for most sub-models. According to the results, the sub-model in Sorted LLaMA with 32 layers performs almost as well as regular fine-tuning of the full-size model. This showcases the impressive ability of our proposed paradigm to generate powerful, small sub-models that have similar performance to the original model.

3.2 A comparison between the learned probability distribution of SoFT versus SFT

The goal of sorted tuning is to make sub-models similar to the full model. To explore the efficacy of our SoFT in closing the gap between submodels and the full model, we measure the similarity between probability distributions of each token in each sub-model versus the full model using the Kullback–Leibler (KL) divergence. Figure 2 compares the probability distribution of Sorted LLaMA and SFT sub-models at different output positions.

Figure 2 (Right) shows the comparison among different SFT layers and the last Sorted LLaMA layer. The figure shows only SFT's full-size output distribution is close to the sorted full-size model, while the other layers' distribution diverges fast in the initial steps compared to the SoFT. Figure 2 (Left) compares the output distribution of all sorted layers to the last SFT layer. Compared to Figure 2 (Right), Figure 2 (Left) Sorted LLaMA can preserve the output distribution close to the SFT full-size model even in lower layers for initial output tokens.

4 Conclusion

In this work, we present sorted LLaMA, many-in-one LLaMA models for dynamic inference obtained by using Sorted fine-tuning instead of supervised fine-tuning. Sorted LLaMA unlocks the potential representation ability of intermediate layers, offering dynamic adaptation without pre-training or additional expenses related to model compression. As all sub-models remain integral components of the original model, the burden of storage requirements and transition costs between different computational demands is minimized, making the management of multiple models during inference a practical reality. Our systematic evaluation challenged conventional wisdom by focusing on the effectiveness of middle layers in producing high-quality results for specific downstream tasks.

References

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*, 2023. URL `https://arxiv.org/abs/2306.01116`.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Maitha Alhammadi, Mazzotta Daniele, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. The falcon series of language models: Towards open frontier models. 2023.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33: 9782–9793, 2020.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. Deebert: Dynamic early exiting for accelerating bert inference. *arXiv preprint arXiv:2004.12993*, 2020.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*, 2019.

Mojtaba Valipour, Mehdi Rezagholizadeh, Hossein Rajabzadeh, Marzieh Tahaei, Boxing Chen, and Ali Ghodsi. Sortednet, a place for every network and every network in its place: Towards a generalized solution for training many-in-one neural networks. *arXiv preprint arXiv:2309.00255*, 2023.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, et al. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization. *arXiv preprint arXiv:2306.05087*, 2023.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.