CQIL: Inference Latency Optimization with Concurrent Computation of Quasi-Independent Layers

Longwei Zou[1], **Qingyang Wang**[2], **Han Zhao**[3],
Jiangang Kong[3], **Yi Yang**[3], **Yangdong Deng**[1]
[1]Tsinghua University, [3]DiDi Global Inc,
[2]BNU-HKBU United International College
zoulw22@mails.tsinghua.edu.cn, q030026149@mail.uic.edu.cn
{zhaohan,kongjiangang,yangyiian}@didiglobal.com
dengyd@tsinghua.edu.cn

Abstract

The fast-growing large scale language models are delivering unprecedented performance on almost all natural language processing tasks. However, the effectiveness of large language models are reliant on an exponentially increasing number of parameters. The overwhelming computation complexity incurs a high inference latency that negatively affects user experience. Existing methods to improve inference efficiency, such as tensor parallelism and quantization, target to reduce per-layer computing latency, yet overlook the cumulative latency due to the number of layers. Recent works on reducing the cumulative latency through layer removing, however, lead to significant performance drop. Motivated by the similarity of inputs among adjacent layers, we propose to identify quasi-independent layers, which can be concurrently computed to significantly decrease inference latency. We also introduce a bypassing technique to mitigate the effect of information loss. Empirical experiments of the proposed approach on the LLaMA models confirm that Concurrent Computation of Quasi-Independent Layers (CQIL) can reduce latency by up to 48.3% on LLaMA-33B, while maintaining a close level of performance. [1]

1 Introduction

Large Language Models (LLMs) are offering unprecedented power to deliver remarkable performance across diverse tasks of natural language processing. The exceptional performance, however, comes at the cost of increasing model size and, consequently, higher inference latency. For example, the per-token inference time for GPT-4 is approximately three times longer than that of GPT-3.5, according to measurements from the OpenAI API [2]. Such a high latency has a direct impact on user experience, highlighting the urgent need to mitigate inference delays of LLMs.

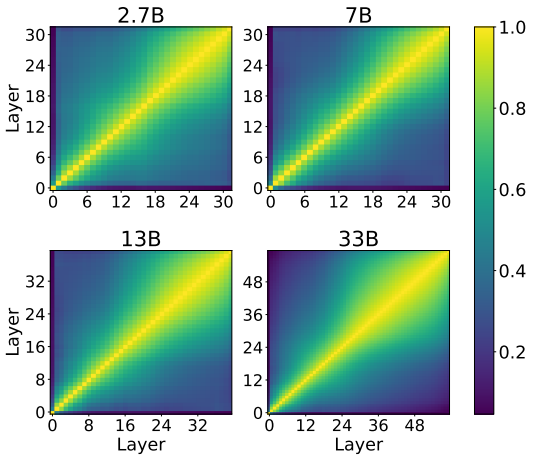

Figure 1: Similarity of inputs across layers in LLaMA-1 models. Sub-figure with title "2.7B" represents the similarity of inputs in Sheared-LLaMA-2.7B(Xia et al., 2023). It highlights that adjacent layers have highly similar input. Notably, such similarity of inputs becomes increasingly evident in larger models at deeper layers, suggesting the quasi-independence of deeper layers and the potential for parallel computation.

LLMs typically consist of a large number of sequentially connected layers with identical structures. For better illustration, we analyze the inference latency of LLMs along two dimensions: per-layer latency and the cumulative latency due to all layers. Existing low-latency inference methods, like tensor parallelism, quantization (Dettmers et al., 2022; Xiao et al., 2023), and unstructured pruning (Han et al., 2016; Frantar and Alistarh, 2023; Sun et al., 2023) and low-rank factorization (Lan et al., 2020; Lv et al., 2023), primarily focus on minimizing per-layer latency. Meanwhile, there are works, such as structured pruning (Ma et al., 2023; Xia et al., 2023) and dynamic early existing (Xin et al., 2020), proposed to address the cumulative latency through removing and/or dynamically

[1]Code is available at https://github.com/Photooon/CQIL

[2]Note that latency can vary depending on time and location; the reported times is based on observations at the authors' location.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 7293–7307

omitting layers. These approaches, unfortunately, often result in considerable performance degradation and potential loss of learned knowledge (Geva et al., 2021). Therefore, it's critical to develop effective methods that can reduce the latency related to the total number of layers while preserving the model performance.

This work is inspired by the observation that adjacent layers in LLMs share significantly similar inputs. Such similarity suggests the possibility of substituting a layer's input with that of a certain preceding layer without significantly altering its output. Such layers sharing input are designated as quasi-independent layers in this work. As a result, the computing dependency between adjacent layers can be eliminated and thus unleash the potential of parallel computation. We introduce a framework of Concurrent Computation of Quasi-Independent Layers (CQIL), to reduce LLM inference latency by parallelizing the computation across layers with similar inputs. Additionally, we develop a bypassing technique to transmit the output of attention modules among input-aligned layers, with the purpose of minimizing the information loss. Extensive experiments demonstrate reductions of inference latency by up to 48.3% on the LLaMA-33B model, with a minor impact on performance. We also discuss the implication of CQIL in the context of ensembles, which may offer deeper insights into the fundamental characteristic of LLMs.

The major contributions of this work are as follows. First, we propose CQIL, a novel approach to enhance the inference efficiency of LLMs through concurrent computation of quasi-independent layers, effectively addressing the challenge posed by the increasing number of layers. Second, our method enables the adaptation of pre-trained LLMs into ensemble-like models with minimal performance loss, which may provide deeper insights for layers' characters in LLMs. Third, We effectively reduce the inference latency of LLaMA models, with minimal impact on the model performance.

2 Related Work

Efficient Inference Approaches Model compression techniques, such as pruning (Frantar and Alistarh, 2023; Sun et al., 2023; Ma et al., 2023; Xia et al., 2023), quantization (Dettmers et al., 2022; Xiao et al., 2023), low-rank factorization (Lan et al., 2020; Lv et al., 2023), and knowledge distillation (Hinton et al., 2015; Sanh et al., 2019; Jiao et al., 2020; Sun et al., 2020), reduce inference latency by trimming parameters in the model. Methods like dynamic early exit (Xin et al., 2020) and speculative decoding (Leviathan et al., 2023; Chen et al., 2023) leverage intermediate layer output and output from smaller models to predict the final outcome ahead of time. Flash Attention (Dao et al., 2022) enhances the efficiency of attention computation by carefully orchestrating computation and memory usage. These methods are orthogonal to our approach, allowing for potential integration for a higher level of improvement. Notably, we test the combination of the pruning approach with our method, detailed in Section 5.6.

Parallelism Beyond removing the model's parameters, latency reduction and throughput enhancement can also be achieved through parallel computation strategies. Typically, parallelism in LLM computation includes data parallelism (Rajbhandari et al., 2020), pipeline parallelism (Harlap et al., 2018; Huang et al., 2019), and tensor parallelism. Among these, only tensor parallelism addresses the inference latency by distributing layers computation across multiple GPUs. Similar to tensor parallelism, our method employs additional GPUs to decrease latency. Therefore, we conducted comparison experiments in Section 5.5. Results show that, compared with tensor parallelism, our method consistently reduces latency across various batch size, making it more amenable to the scenario of online inference. In addition, our approach is orthogonal to tensor parallelism and we could achieve further acceleration by integrating tensor parallelism with CQIL.

Transformers with Parallel Architecture Previous works have explored the acceleration of pretraining phases through the parallelization of transformer architectures. GPT-J (Wang and Komatsuzaki, 2021) and PaLM (Chowdhery et al., 2022) achieve efficiency improvements by parallelizing the attention and feedforward modules within transformer layers, allowing concurrent computation for the beginning projection in these two modules. Other research efforts (Gao et al., 2020; Wang et al., 2023) have enhanced model performance by expanding model width through parallel layers design, and still focus on the pre-training stage. Our work applies concurrent computation of layers to a pretrained LLMs, aiming at reducing latency while maintaining model performance. Our approach thus differs from existing methods that focus on

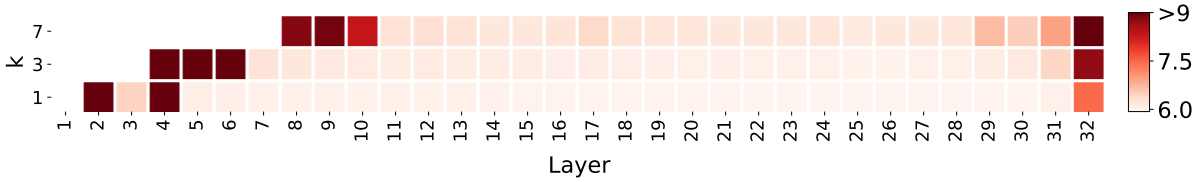

Figure 2: Sensitivity of Input Substitution. We individually replace the input of layer l with that of the layer $l-k$ and evaluate the perplexity. A darker block indicates a higher perplexity and diminished performance. When $k \geq l$, there is no corresponding layer for $l-k$, therefore these parts are left blank in the figure. The original perplexity is around 6. The drawing shows that both bottom and top layers (bottom refers to the direction close to the embedding layer) are sensitive to the input substitution, whereas the majority of middle layers are relatively insensitive.

pre-training transformers with parallel architecture in terms of both motivation and methodology.

3 Preliminary

In this section, we present a systematic investigation that motivates the design of the proposed approach introduced in Section 4. Our discussion proceeds under the assumption of pre-layer normalization, which is adopted by most LLMs (Brown et al., 2020; Zhang et al., 2022b; Touvron et al., 2023). In following experiments, we utilize the LLaMA-1 (Touvron et al., 2023) and Sheared-LLaMA-2.7B (Xia et al., 2023) models, with the input samples randomly selected from the RedPajama (Computer, 2023) dataset. We first investigate the similarity of inputs across layers in LLMs, which suggests the quasi-independence of deeper layers and the potential for parallel computation. Second, we explore the effect of substituting a layer's input with that of its preceding layers, which aids in identifying potential layers for parallel processing with minor performance degradation.

3.1 Similarity of Layer Input

For a LLM with L layers, we define the input to the layer l as $x_l \in \mathbf{R}^{B,T,H}$, where B, T, and H denote batch size, token count, and hidden dimension size, respectively. The output from the layer l is denoted as $x_{l+1} = x_l + F_l(x_l)$. We employ cosine similarity to quantify the similarity of inputs between layers.

Figure 1 reveals the similarity of inputs across layers in LLaMA models with different parameter sizes. The following two primary observations emerge from our experiment results. First, input similarity intensifies with increasing depth within the model, attributed to the cumulative effect of pre-layer normalization, which makes the input difference $F_{l-1}(x_{l-1})$ between adjacent layers $l-1$ and l increasingly negligible compared with the cumulative value $x_l = x_1 + F_1(x_1) + ... + F_{l-1}(x_{l-1})$. Second, larger models exhibit more obvious similarity of inputs, suggesting that layers in such models are quasi-independent and parallel computation may be more readily available in such models.

While prior research (Zhang et al., 2023; Din et al., 2023) has identified output similarity across layers to facilitate inference efficiency through pruning, our approach is based on a different motivation with a more systematic perspective. Experimental results of previous works indicate that layer pruning often result in substantial performance declines. For instance, although the Sheared-LLaMA-1.3B model has a similar hidden dimension size to that of the Sheared-LLaMA-2.7B, it contains eight fewer layers and consequently results in significantly performance drop on downstream tasks. Contrary to removing layers, our approach seeks to parallelize layer computation to reduce inference latency while preserving the model performance. In addition, our method is orthogonal to pruning techniques and the compatibility is further demonstrated in Section 5.6.

We then empirically assess the effect of substituting each layer's input with that of preceding layers, facilitating the identification of layers suitable for parallel computation.

3.2 Sensitivity of Input Substitution

As shown in Figure 1, adjacent layers have similar inputs. It is thus appealing to identify possible layers for parallel computation, i.e., quasi-independent layers. In this section, we substitute the input of a layer with that of preceding layers to evaluate the sensitivity of each layer and thus justify the feasibility of quasi-independent layers. Specifically, the output of the model can be written as

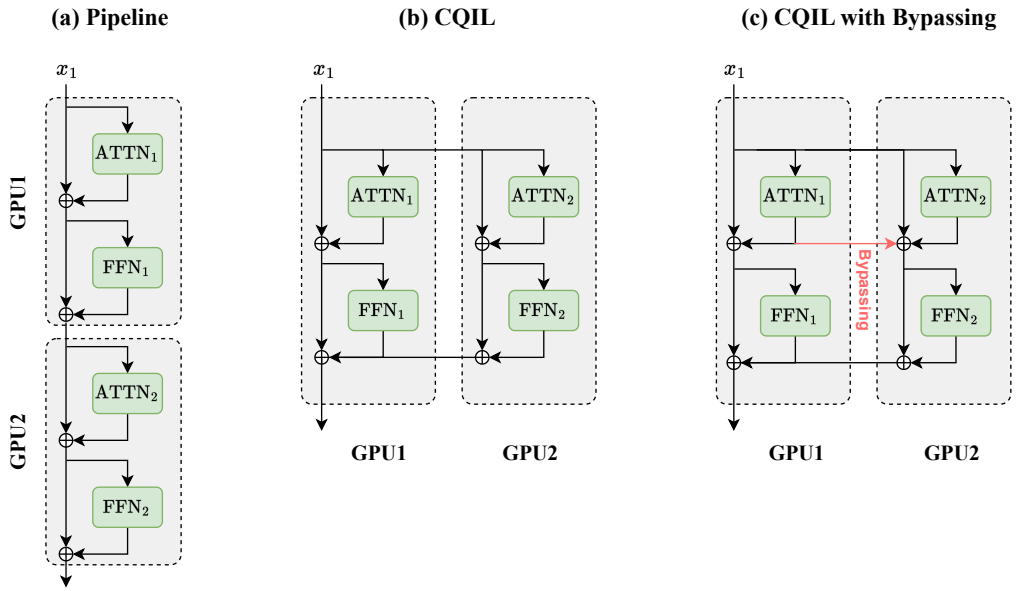

Figure 3: The proposed method. (a-c) depict the pipeline inference as well as the CQIL with and without the bypassing technique. The pipeline inference represents the standard setup, where layers are processed sequentially. In contrast, CQIL substitutes the input for layer 2 with that of layer 1, enabling concurrent computation across two GPUs for latency reduction. Given that both layers produce attention outputs concurrently, and the attention output of layer 1 serves as an input for layer 2 in the original model, the bypassing technique transmits the attention output from GPU1 to GPU2, minimizes the information loss and improves the model performance.

$x_{L+1} = x_1 + F_1(x_1) + ...F_l(x_l) + ...F_L(x_L)$. For layer l, we replace the input of F_l with x_{l-k}, which is the input of layer $l-k$, and keep other terms unchanged. This process is repeated for every layer individually, assessing their adaptability to input changes through perplexity measurements on validation set. When perplexities of the original and the input-substituted models are similar, it is feasible to concurrently compute these two layers.

As illustrated in Figure 2, our substitution trials with the LLaMA-7B model indicate that the input for a majority of the middle layers can be effectively replaced by those of their immediate predecessors while maintaining a similar performance. It is observed that both the bottom and top layers are sensitive to input changes. The bottom layers display less input similarity compared to subsequent layers. In addition, substitution in bottom layers leads to a propagation of errors from substituted inputs through subsequent layers, thereby amplifying their susceptibility to input changes. The top layers, due to their direct connection with the output distribution, also experience a notable impact on performance if their input are substituted.

The empirical findings suggest that most middle layers offer a tolerance for input substitution. In other words, it is feasible to relax the computing dependence by using the same input to quasi-independent layers, revealing the potential of parallel computation for latency reduction.

4 Methodology

4.1 Problem Statement

Given a LLM with pre-layer normalization, we can formulate the computation of layer l as Eq. 1 and Eq. 2. ATTN and FFN represent the attention and feedforward modules in each layer. n denotes the number of heads, and d_k specifies the dimension size of each head.

$$x_{l+1} = x_l + \text{ATTN}_l(x_l) + \text{FFN}_l(x_l + \text{ATTN}_l(x_l)) \tag{1}$$

$$\begin{aligned}
\text{ATTN}(x) &= \text{MHA}(\text{Norm}(x)) \\
\text{MHA}(x) &= \text{Concat}(\text{head}_1, ..., \text{head}_n)\boldsymbol{W}^O \\
\text{head}_i &= \text{Attention}(x\boldsymbol{W}_i^Q, x\boldsymbol{W}_i^K, x\boldsymbol{W}_i^V) \\
\text{Attention}(Q,K,V) &= \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \\
\text{FFN}(x) &= \boldsymbol{W}_2 f(\boldsymbol{W}_1(\text{Norm}(x)) + b_1) + b_2
\end{aligned} \tag{2}$$

Algorithm 1: Concurrent Computation of Quasi-Independent Layers

Input : input x_1, model with L layers, group size p, start layer s, end layer e and bypassing distance d.

Output : output x_L

for $l = 1 \rightarrow s$, *step*=1 **do**
  $a_l = x_l + \text{ATTN}_l(x_l)$
  $x_{l+1} = a_l + \text{FFN}_l(a_l)$
end
for $l = s + 1 \rightarrow e$, *step*=p **do**
  reqs1, reqs2, attns = [], [], []
  for $i = 0 \rightarrow p - 1$, *step*=1 **do**
    reqs1.add(non_block_exec(
    $a_{l+i,attn} = \text{ATTN}_{l+i}(x_l)$, gpu=$i$))
  end
  for $i = 0 \rightarrow p - 1$, *step*=1 **do**
    reqs1[i].wait()
    attns.add($a_{l+i,attn}$)
    $bp = \text{sum}(\text{attns}[-\min(d, i) + 1 :])$
    reqs2.add(non_block_exec(
    $a_{l+i,ffn} = \text{FFN}_{l+i}(x_l + bp)$, gpu=$i$))
  end
  $x_{l+p} = x_l$
  for $i = 0 \rightarrow p - 1$, *step*=1 **do**
    reqs2[i].wait()
    $x_{l+p} = x_{l+p} + a_{l+i,attn} + a_{l+i,ffn}$
  end
end
for $l = e + 1 \rightarrow L$, *step*=1 **do**
  $a_l = x_l + \text{ATTN}_l(x_l)$
  $x_{l+1} = a_l + \text{FFN}_l(a_l)$
end

Our objective is to replace the input x_l in Eq. 1 with $x_i, i < l$. This substitution implicitly enables the parallel computation of layers l and i and thus reduces the inference latency.

4.2 Layer Partition and Parallelism

Considering a LLM with L layers, we partition layers into K groups $G_1, G_2, ..., G_K$. Within each group, layers share the same input, allowing for concurrent computation of them. The computation for group G_k, where $1 \leq k \leq K$, can be formalized as shown in Eq. 3. ATTN and FFN operations within each summation in this equation are executed in parallel. Specifically, when the size of each group is 1, Eq. 3 reverts to the sequential computation in Eq. 1.

$$x_{G_{k+1}} = x_{G_k} + \sum_{l \in G_k} \text{ATTN}_l(x_{G_k}) + \sum_{l \in G_k} \text{FFN}_l(x_{G_k} + \text{ATTN}_l(x_{G_k})) \tag{3}$$

The number of potential partition schemes can be considerable. Therefore, we employ a straightforward approach to divide layers into groups. Specifically, with a maximum group size of p, start layer s, and end layer e, we sequentially arrange layers from s to e into groups of size p, while organizing remaining layers into groups of size 1. For instance, the LLaMA-7B model with 32 layers, can be segmented into groups as $\{1\} \rightarrow \{2\} \rightarrow ... \rightarrow \{8\} \rightarrow \{9, 10\} \rightarrow \{11, 12\} \rightarrow ... \rightarrow \{29, 30\} \rightarrow \{31\} \rightarrow \{32\}$, with $p = 2, s = 9, e = 30$. Such partitioning strategy helps prevent low GPU utilization and significant performance degradation from parallelizing the bottom and top layers. Algorithm 1 further expound the concurrent computation process. Hyperparameter p is determined according to the number of GPUs. The choice of hyperparameters s, e depends on balancing performance against acceleration, discussed further in Section 5.4.

4.3 Bypassing

The transformer layer includes an attention module and a feedforward module. Notably, parallel computation of p layers in group G_k allows concurrent accesses to attention module outputs $\text{ATTN}_l(x_{G_k}), l \in G_k$. We could transfer $\text{ATTN}_l(x_{G_k})$ to the feedforward module of layer j, $j > l$ and $j \in G_k$, thereby minimizing information loss. We designate such a approach as bypassing by following the terminology of computer architecture. We define the bypassing distance d as the maximum of $j - l$. With bypassing, the formulation for G_{k+1} becomes Eq. 4.

$$x_{G_{k+1}} = x_{G_k} + \sum_{l \in G_k} \text{ATTN}_l(x_{G_k}) + \sum_{l \in G_k} \text{FFN}_l(x_{G_k} + \text{ATTN}_l(x_{G_k}) + \sum_{l' \in G_k, 1 \leq l - l' \leq d} \text{ATTN}_{l'}(x_{G_k})) \tag{4}$$

Table 1: Downstream tasks performance and latency reduction. $p = 1$ refers to the original model. Results indicate that our method effectively reduce the inference latency of LLaMA models, while preserving the model performance. Additionally, as the model size and the number of GPUs increases, CQIL achieves further latency reductions.

Model	Partition Strategy			Commonsense&Reading Comprehension					
	p	s	e	SciQ	PIQA	WinoGrande	ARC-E	ARC-C	HellaSwag
LLaMA-7B	1	1	32	93.0	79.2	70.0	72.9	44.8	76.2
	2	13	30	90.8	78.3	69.0	70.1	41.3	73.5
	4	15	30	89.5	76.0	68.3	65.5	39.6	71.0
LLaMA-13B	1	1	40	91.3	80.1	72.8	74.8	47.6	79.1
	2	11	38	89.3	79.4	69.6	72.1	44.9	76.8
	4	15	38	90.7	78.8	70.2	71.0	43.7	75.5
LLaMA-33B	1	1	60	94.6	82.3	76.0	79.0	52.1	82.6
	2	11	58	94.5	80.4	71.8	76.1	51.5	80.9
	4	19	58	94.0	79.4	74.7	75.0	50.1	80.8

Model	Partition Strategy			Continued		LM	World Knowledge	Downstream Tasks	Latency
	p	s	e	LogiQA	BoolQ	LAMBADA	MMLU (5)	Average Score	Reduction
LLaMA-7B	1	1	32	30.0	75.1	73.5	35.1	65.0	0%
	2	13	30	29.0	74.4	72.9	32.9	63.2	27.0%
	4	15	30	29.5	72.6	69.9	33.2	61.5	36.0%
LLaMA-13B	1	1	40	32.0	77.9	76.2	46.7	67.8	0%
	2	11	38	29.3	74.6	73.7	40.7	65.0	34.0%
	4	15	38	30.3	76.8	73.7	43.1	65.4	43.2%
LLaMA-33B	1	1	60	31.8	82.6	77.6	58.2	71.7	0%
	2	11	58	28.9	80.5	76.9	50.5	69.2	38.6%
	4	19	58	30.1	81.0	75.4	52.0	69.3	48.3%

Moreover, the number of transmissions for bypassing is $d(2p - d - 1)/2$. To avoid communication bottlenecks, we set $d = 1$ across all experiments unless otherwise specified. The effect of bypassing distances are detailed in Section 5.3.

4.4 Fine-tuning

Without additional training, models with concurrent computation of quasi-independent layers maintain a close level of performance to the original. To achieve better performance, we fine-tune the model using LoRA(Hu et al., 2022) on the pre-training dataset. Fine-tuning with just 0.5B tokens is sufficient to almost reach the original performance, suggesting that middle layers in original LLMs may inherently function as ensembles. We discuss such implication further in Section 6. We believe that continual pre-training with more tokens will yield further performance enhancements. However, due to computational resource constraints, we leave the continual pre-training for the model with CQIL as future work.

5 Experiment

5.1 Experimental Setup

Model configurations and Dataset We apply CQIL on Sheared-LLaMA-1.3B, Sheared-LLaMA-2.7B (Xia et al., 2023), LLaMA-7B, LLaMA-13B and LLaMA-33B (Touvron et al., 2023). Concurrent computation of layers are implemented with Pytorch distributed communication package (Paszke et al., 2019) and Huggingface Transformers library (Wolf et al., 2019). We use RedPajama (Computer, 2023), the replicated dataset of LLaMA1 models, as the training dataset, and adopt the same data mixture used for training LLaMA1. We construct a held-out validation dataset with 500 sequences of 2,048 tokens.

Baselines Our method is orthogonal to most existing works on efficient inference, making it hard to be directly compared with them. Given that both tensor parallelism and our strategy employ additional GPUs to reduce latency, we evaluate the relative efficiency of our method against tensor parallelism. The effectiveness of tensor parallelism implementations can vary significantly. Based on empirical assessments, we select the DeepSpeed inference library (Rajbhandari et al., 2020) as the benchmark for tensor parallelism. DeepSpeed inference library partitions attention and feedforward projection matrices and computes them distributively, which has proven to reduce latency in our testing environment. Furthermore, we explore combination of our method and the pruning technique, showing further potential for latency reduction.

Training We fine-tune models using LoRA on Nvidia A100 GPUs (80GB). Details of fine-tuning are demonstrated in Appendix C.

Table 2: Effect of bypassing. The performance on downstream tasks gradually improves as d increases. Due to the communication cost, there is a slight decline in latency reduction as d grows.

Bypassing Distance (d)	Downstream Tasks Average Score	Latency Reduction
0	60.9	35.9%
1	61.2	35.7%
2	61.4	35.6%
3	61.5	34.9%

Evaluation We evaluate the downstream tasks performance with lm-evaluation-harness package (Gao et al., 2023). We evaluate 0-shot accuracy of SciQ (Welbl et al., 2017), PIQA (Bisk et al., 2020), Winogrande (Sakaguchi et al., 2020), ARC Easy, ARC Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), LogiQA (Liu et al., 2020), BoolQ (Clark et al., 2019), and LAMBADA (Paperno et al., 2016). We show accuracy of 5-shot MMLU (Hendrycks et al., 2021). For the latency, we measure the inference speed on a machine with 8 Nvidia A100 GPUs connected by NVLink and report the average latency reduction across batch sizes ranging from 1 to 256.

5.2 Downstream Tasks Performance and Latency Reduction

Table 1 shows that our method maintains a close level of performance to that of the original model while achieving significant latency reduction. LLaMA-13B and LLaMA-33B utilizing $p = 4$ outperform those with $p = 2$, attributed to employing a larger s which keeps more bottom layers unchanged. With $p = 4$, our method processes four layers in parallel, achieving further latency improvements over $p = 2$. Additionally, as the model size increases, our method demonstrates further latency reductions. This improvement is attributed to the increased quasi-independence of layers observed in larger models.

5.3 Effect of Bypassing

Table 2 elucidates the effect of bypassing distances on LLaMA-7B with $p = 4, s = 14, e = 29$. The results suggest that increasing bypassing distance d leads to a gradual improvement of performance, attributed to diminished information loss. Meanwhile, there's a slight decrease in latency reduction as the communication cost rises with larger d. However, even at $d = 3$, the additional communication cost associated with the bypassing method is still minor in our testing environment. We also recognize that the communication cost depends on the speed of links between GPUs. Therefore, we suggest to select d based on the inference environment.

Table 3: Trade-Off between performance and acceleration. Model performance and latency reduction consistently vary along with s and e.

p	s	e	Downstream Tasks Average Score	Latency Reduction
2	13	30	63.2	27.0%
	9		60.6	32.4%
	11		62.0	30.3%
	15		63.7	23.4%
	17		63.9	21.0%
		24	63.4	18.0%
		26	63.3	20.9%
		28	63.3	23.9%
		32	62.8	26.9%

5.4 Trade-Off between Performance and Acceleration

CQIL involves three hyperparameters, p, s, and e. The selection of p is based on the number of available GPUs. Choices for s and e depend on balancing performance against acceleration. As shown in Table 3, the performance on downstream tasks and latency reduction vary consistently along with s and e. A larger s and a smaller e, thereby more layers are unchanged, lead to improved performance and decrease in latency. We suggest to select s and e according to the balance between performance and latency reduction.

5.5 Comparison with Tensor Parallelism

Both tensor parallelism and our method utilize additional GPUs to reduce latency. Therefore, we assess the latency reduction ratio between these two approaches. Given that DeepSpeed inference framework involves optimizations beyond tensor parallelism, it is challenging to make a direct comparison of latency. Therefore, we first measure the latency of both methods using a single GPU under various batch sizes, then evaluate the relative latency reduction ratio with an increasing number of GPUs. The comparison is carried out on the LLaMA-13B model, which is sufficiently large for tensor parallelism to be applied across four GPUs. The maximum batch size of tensor parallelism is 64 due to the limit of available GPU memory.

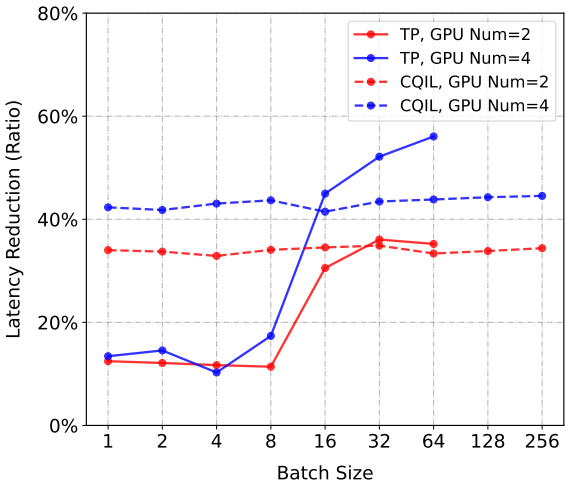

Figure 4: Comparison with Tensor Parallelism. CQIL achieves consistent latency reduction on all batch sizes, benefiting the online inference.

As shown in Figure 4, tensor parallelism does not significantly accelerate the inference for small batch sizes. In contrast, CQIL consistently offers latency reductions for LLaMA-13B, making it more amenable to the scenario of online inference. Moreover, tensor parallelism is complementary to our approach, allowing for the parallel computation of bottom layers, which are sensitive to input substitution. Integrating tensor parallelism with our approach may reduce the latency further. Due to time constraints, we leave the integration of tensor parallelism with CQIL for future work.

5.6 Combined with Pruning

Pruning has been extensively studied as an efficient technique to accelerate model inference. Recently, Xia et al. (2023) has proposed targeted structured pruning to produce efficient LLMs. Our method is orthogonal to pruning and thus possible to be integrated with it. Table 4 presents the result of applying CQIL on the Sheared-LLaMA-2.7B model with $p = 2, s = 13, e = 30$, and Sheared-LLaMA-1.3B with $p = 2, s = 13, e = 22$. Results show that our approach achieves further latency reduction for pruned models, proving its effectiveness.

6 Discussion

Currently, the explanation of the effectiveness of transformer based LLMs can be classified into two views, pipeline (Tenney et al., 2019; Jawahar et al., 2019; Geva et al., 2021) and ensemble (Veit et al., 2016; Greff et al., 2017; Bhojanapalli et al., 2021). Based on following observations, our research suggests that LLMs works as a combination of both pipeline and ensemble mechanisms.

Table 4: Combined with Pruning. CQIL achieves further latency reduction while maintaining minimal downstream tasks performance drop.

Model	Downstream Tasks Average Score	Latency Reduction
LLaMA-7B	65.0	0%
Sheared-LLaMA-2.7B	58.8	55.1%
CQIL-Sheared-LLaMA-2.7B	57.4	66.0%
Sheared-LLaMA-1.3B	53.1	74.5%
CQIL-Sheared-LLaMA-1.3B	52.7	78.5%

First, our findings reveal that the bottom layers of LLMs display distinct input/output characteristics, rendering them challenging to be parallelized. On the other hand, the middle and top layers demonstrate considerable similarity in their inputs, making it possible for straightforward conversion to parallel execution without sacrificing the model performance. This observation is compatible with previous work (Zhang et al., 2022a), proving the increased robustness in higher layers within the transformer architecture.

In other words, our contributions substantiate that LLMs employ a pipeline mechanism at lower levels and an ensemble strategy at higher levels, particularly through the successful parallelization of the middle and upper layers. This work thus provide further evidence to deepen our comprehension on the internal mechanism of LLMs.

7 Conclusion

In this paper, we propose an efficient concurrent computation framework on quasi-independent layers that are pervasive in LLMs with the purpose of reducing inference latency while maintaining model performance. To mitigate the potential information loss resulting from input alignment and improve the performance, we develop the bypassing technique, transmitting attention outputs among input-aligned layers. Our experimental results justify the effectiveness of CQIL method. In the future, we will apply CQIL on even larger models and implement the integration of tensor parallelism with our approach to achieve further latency reductions.

Limitations

This work has three primary limitations. First, CQIL requires additional GPUs to reduce the inference latency, limiting its application in the environment equipped with only a single GPU. Second,

not every layer in LLMs could be computed concurrently, especially for layers at the bottom and top. These layers are computed on a single GPU, leaving remaining GPUs ignored. To efficiently utilize all GPUs, we suggest to apply tensor parallelism for these layers. Due to time constraints, the integration of tensor parallelism and our approach is leaved for future work. Finally, the computing FLOPS remain unchanged. Therefore, from an energy consumption perspective, CQIL does not result in electricity savings.

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *CoRR*, abs/2309.16609.

Srinadh Bhojanapalli, Ayan Chakrabarti, Daniel Glasner, Daliang Li, Thomas Unterthiner, and Andreas Veit. 2021. Understanding robustness of transformers for image classification. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 10211–10221. IEEE.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *CoRR*, abs/2302.01318.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *CoRR*, abs/2204.02311.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2924–2936. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.

Together Computer. 2023. Redpajama: an open dataset for training large language models.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *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*.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *CoRR*, abs/2208.07339.

Alexander Yom Din, Taelin Karidi, Leshem Choshen, and Mor Geva. 2023. Jump to conclusions: Short-

cutting transformers with linear transformations. *CoRR*, abs/2303.09435.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10323–10337. PMLR.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Peng Gao, Chiori Hori, Shijie Geng, Takaaki Hori, and Jonathan Le Roux. 2020. Multi-pass transformer for machine translation. *CoRR*, abs/2009.11382.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 5484–5495. Association for Computational Linguistics.

Klaus Greff, Rupesh Kumar Srivastava, and Jürgen Schmidhuber. 2017. Highway and residual networks learn unrolled iterative estimation. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A. Roberts. 2024. The unreasonable ineffectiveness of the deeper layers. *CoRR*, abs/2403.17887.

Song Han, Huizi Mao, and William J. Dally. 2016. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Aaron Harlap, Deepak Narayanan, Amar Phanishayee, Vivek Seshadri, Nikhil R. Devanur, Gregory R. Ganger, and Phillip B. Gibbons. 2018. Pipedream: Fast and efficient pipeline parallel DNN training. *CoRR*, abs/1806.03377.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Xu Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. 2019. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 103–112.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3651–3657. Association for Computational Linguistics.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 4163–4174. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 19274–19286. PMLR.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3622–3628. ijcai.org.

Xiuqing Lv, Peng Zhang, Sunzhu Li, Guobing Gan, and Yueheng Sun. 2023. Lightformer: Light-weight transformer using svd-based weight transfer and parameter sharing. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada,*

July 9-14, 2023, pages 10323–10335. Association for Computational Linguistics.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *CoRR*, abs/2305.11627.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2024. Shortgpt: Layers in large language models are more redundant than you expect. *CoRR*, abs/2403.03853.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2020, Virtual Event / Atlanta, Georgia, USA, November 9-19, 2020*, page 20. IEEE/ACM.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8732–8740. AAAI Press.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. 2023. Powerinfer: Fast large language model serving with a consumer-grade GPU. *CoRR*, abs/2312.12456.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2023. A simple and effective pruning approach for large language models. *CoRR*, abs/2306.11695.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2158–2170. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4593–4601. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Andreas Veit, Michael J. Wilber, and Serge J. Belongie. 2016. Residual networks behave like ensembles of relatively shallow networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 550–558.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`.

Dong Wang, Zixiang Wang, Ling Chen, Hongfeng Xiao, and Bo Yang. 2023. Cross-parallel transformer: Parallel vit for medical image segmentation. *Sensors*, 23(23):9488.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text, NUT@EMNLP 2017, Copenhagen, Denmark, September 7, 2017*, pages 94–106. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. *CoRR*, abs/2310.06694.

Guangxuan Xiao, Ji Lin, Mickaël Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 38087–38099. PMLR.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. Deebert: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2246–2251. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4791–4800. Association for Computational Linguistics.

Chiyuan Zhang, Samy Bengio, and Yoram Singer. 2022a. Are all layers created equal? *J. Mach. Learn. Res.*, 23:67:1–67:28.

Kaiyan Zhang, Ning Ding, Biqing Qi, Xuekai Zhu, Xinwei Long, and Bowen Zhou. 2023. Crash: Clustering, removing, and sharing enhance fine-tuning without full large language model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 9612–9637. Association for Computational Linguistics.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022b. OPT: open pre-trained transformer language models. *CoRR*, abs/2205.01068.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 19–27. IEEE Computer Society.

A Attention-FFN Parallelism

Given that both the attention and feedforward modules use pre-layer normalization, aligning their inputs is feasible. However, the computation times for these two modules usually do not match, leading to one GPU being underutilized if they are processed concurrently on separate GPUs. Therefore, attention-ffn parallelism is not employed in our study. Nevertheless, parallelism between the attention and feedforward modules holds potential for acceleration, especially when two computational resources are available but their speeds differ. For example, Song et al. (2023) proposes to utilize both CPU and GPU for LLMs inference on personal computers. The formulation of attention-ffn parallelism is shown in Eq.5.

$$x_{l+1} = x_l + \text{ATTN}_l(x_l) + \text{FFN}_l(x_l) \quad (5)$$

Our empirical experiments show that, the attention and feedforward modules can be computed concurrently in most layers. Specifically, we parallelize attention and feedforward modules for layers ranging from 13 to 30 in the LLaMA-7B model and achieve an average downstream task score of 63.7 (compared to the original model's score of 65.0). Such parallelism may enhance the efficiency of existing CPU-GPU collaborative inference frameworks, paving the way for further acceleration.

Table 5: Pre-training Parallel Architecture.

Model	LAMBADA	PTB	WikiText103
GPT2	70.8	139.4	56.2
CQIL-FT-GPT2	84.6	156.2	73.5
CQIL-Pretrained-GPT2	84.5	189.1	81.4

B Pre-training Parallel Architecture

Our work focuses on converting pre-trained LLMs into a model with parallel layers. An alternative strategy is to pre-train an architecture with parallel layers from scratch, potentially yielding comparable results and latency improvements. We explored this direction through experiments. We first pre-trained a 12 layers GPT-2(Radford et al., 2019) model with 100K steps on Wikipedia and BookCorpus(Zhu et al., 2015) datasets. Then the model was converted to the parallel architecture with $p = 2, s = 3, e = 12$ and fine-tuned with 5K steps. Meanwhile, we pre-trained the parallel architecture with 105K steps from scratch. We evaluate zero-shot perplexities on LAMBADA, PTB and WikiText103 as the performance of downstream tasks. Results in Table 5 indicate that training a parallel architecture from scratch does not achieve

the downstream task performance attained by fine-tuning the transferred model. This performance gap may arise from the enhanced fitting capabilities inherent to deeper models during pre-training. The finding underscores the critical role of transitioning from pre-trained LLMs to the parallel architecture.

C Details of Fine-tuning

Table 6: Fine-tuning Details.

Model	p	s	e	Learning Rate	LoRA Rank
Sheared-LLaMA-1.3B	2	13	22	1e-5	2
Sheared-LLaMA-2.7B	2	13	30	1e-5	2
LLaMA-7B	2	13	30	1e-5	2
	4	15	30	1e-5	32
LLaMA-13B	2	11	38	1e-4	64
	4	15	38	1e-4	64
LLaMA-33B	2	11	58	1e-5	64
	4	19	58	1e-5	64

We fine-tune all models with context length of 2,048, weight decay of 0.1, batch size of 32, dropout probability of 0, warmup steps of 0 and steps of 8000. The learning rate is constant and Adam optimizer is used. For simplicity, lora alpha is always equal to lora rank. We fine-tune all linear matrices without biases in attention and feed-forward modules. Mixed precision of bf16 and DeepSpeed(Rajbhandari et al., 2020) framework are used for fine-tuning. More details could be found in Table 6.

D The Potential Performance Degradation of Pruning

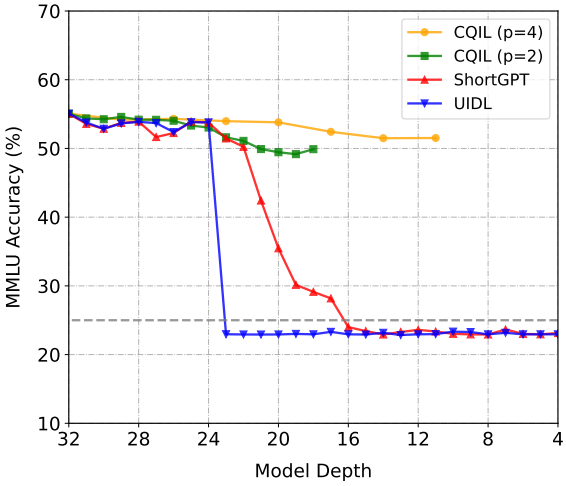

Figure 5: MMLU (zero-shot) comparison with similarity based pruning methods. Dashed gray line represents the random guessing score.

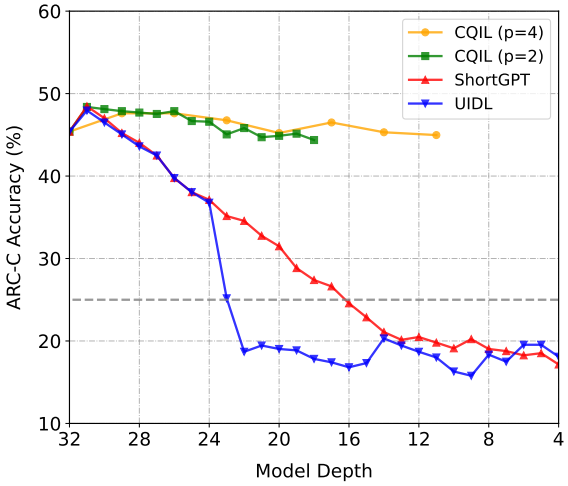

Figure 6: ARC-Challenge (zero-shot) comparison with similarity based pruning methods. Dashed gray line represents the random guessing score.

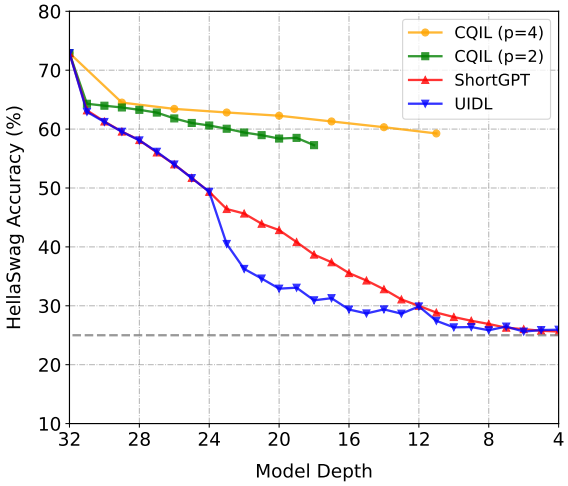

Figure 7: HellaSwag (zero-shot) comparison with similarity based pruning methods. Dashed gray line represents the random guessing score.

During the review phase of our manuscript, two similarity based pruning methods(Men et al., 2024; Gromov et al., 2024) were posted. The findings reported in these studies indicate that the impact on performance is negligible when pruning up to half of the layers in LLMs. Considering the shared motivation but distinct approaches of these methods, we conduct the comparative experiments to demonstrate the potential problems of pruning.

Specifically, we applied the two layer pruning methods for Qwen-7B(Bai et al., 2023). Subsequently, the pruned models were fine-tuned using LoRA with rank of 64, constant learning rate of 1e-4, batch size of 32 context length of 2,048, and steps of 8000. For CQIL, we set $e = 30, d = 1$

Table 7: Downstream tasks performance of models without fine-tuning.

Model	Partition Strategy			Commonsense&Reading Comprehension					
	p	s	e	**SciQ**	**PIQA**	**WinoGrande**	**ARC-E**	**ARC-C**	**HellaSwag**
LLaMA-7B	1	1	32	93.0	79.2	70.0	72.9	44.8	76.2
	2	13	30	89.8	77.3	67.6	68.4	41.3	54.9
	4	15	30	87.4	75.7	65.7	63.1	41.0	70.5
LLaMA-13B	1	1	40	91.3	80.1	72.8	74.8	47.6	79.1
	2	11	38	87.9	78.5	69.5	70.2	44.5	76.1
	4	15	38	86.7	77.9	69.0	64.5	41.6	73.8
LLaMA-33B	1	1	60	94.6	82.3	76.0	79.0	52.1	82.6
	2	11	58	90.8	78.5	70.7	68.9	46.2	77.2
	4	19	58	90.8	79.3	72.6	70.7	47.3	78.7

Model	Partition Strategy			Continued		LM	World Knowledge	Downstream Tasks
	p	s	e	**LogiQA**	**BoolQ**	**LAMBADA**	**MMLU (5)**	**Average Score**
LLaMA-7B	1	1	32	30.0	75.1	73.5	35.1	65.0
	2	13	30	29.8	73.4	68.5	31.4	60.2
	4	15	30	27.5	44.7	56.7	29.8	56.2
LLaMA-13B	1	1	40	32.0	77.9	76.2	46.7	67.8
	2	11	38	31.6	73.2	69.4	39.3	64.0
	4	15	38	28.6	63.7	60.1	44.6	61.1
LLaMA-33B	1	1	60	31.8	82.6	77.6	58.2	71.7
	2	11	58	23.8	77.1	71.3	47.0	65.2
	4	19	58	25.4	80.4	61.3	45.7	65.2

Table 8: Numerical values of input substitution experiments (Layer 1-16).

Layer	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
k=1		868.2	6.4	14.1	6.1	6.0	6.0	6.0	6.0	6.0	6.0	6.0	6.0	6.0	6.0	6.0
k=3				16.3	13.7	11.2	6.2	6.2	6.1	6.1	6.1	6.1	6.1	6.1	6.1	6.1
k=7								8.8	8.9	8.3	6.3	6.3	6.3	6.2	6.2	6.2

Table 9: Numerical values of input substitution experiments (Layer 17-32).

Layer	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
k=1	6.0	6.0	6.0	6.0	6.0	5.9	5.9	5.9	5.9	5.9	5.9	5.9	6.0	6.0	6.0	7.5
k=3	6.1	6.1	6.0	6.0	6.0	6.0	6.0	6.0	6.0	6.0	6.0	6.0	6.1	6.1	6.4	8.7
k=7	6.4	6.3	6.2	6.2	6.2	6.2	6.2	6.2	6.1	6.2	6.2	6.2	6.7	6.5	6.9	10.6

and progressively decrease the s to obtain models with different depth. The CQIL applied models are fine-tuned under identical settings. For ease of reference, we denote these two approaches as ShortGPT(Men et al., 2024) and UIDL(Gromov et al., 2024), respectively. To assess the performance of downstream tasks, the fine-tuned models are evaluated on MMLU, ARC-Challenge and HellaSwag datasets in a zero-shot setting.

The result, as illustrated in Figure 5, demonstrates that the MMLU score has minimal degradation with the pruning of fewer than 9 layers, suggesting the potential redundancy of LLMs. However, as shown in Figures 6 and 7, the performance of pruned models consistently decline on other benchmark datasets. These observations suggest that while the MMLU dataset is substantial, it may not fully capture the comprehensive capabilities or knowledge inherent to LLMs. Consequently, we argue for the necessity of retaining LLM layers to maintain their true functional potential.

E Downstream Tasks Performance of CQIL-Models without Fine-Tuning

Without additional training, models with CQIL still preserve a close level of performance to the original. Results without fine-tuning are shown in Table 7.

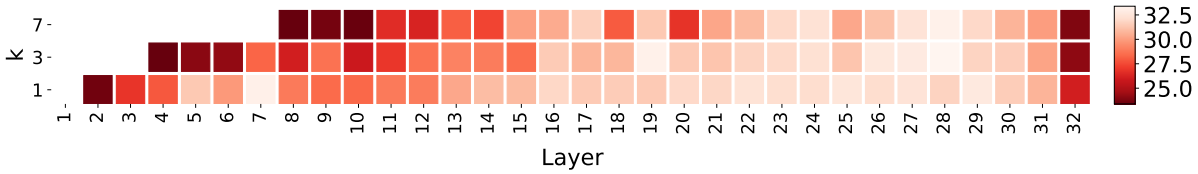

Figure 8: MMLU Sensitivity of Input Substitution. We individually replace the input of layer l with that of the layer $l - k$ and evaluate the MMLU score with zero-shot setting. A darker block indicates a lower MMLU score and diminished performance. Note that the zero-shot MMLU score of the original model is 33.4.

F Additional Details of Input Substitution Experiments

To clarify the details of Figure 2, we give the numerical values of each substitution experiment in Table 8 and 9. Moreover, we conducted the sensitivity experiment on LLaMA-7B with MMLU under zero-shot setting. The results in Table 8 are similar to Figure 2, revealing that the majority of middle layers are relatively insensitive.

G Additional Bypassing Experiments

Table 10: Effect of bypassing on LLaMA-7B with $p = 2$.

Bypassing Distance (d)	**Downstream Tasks Average Score**	**Latency Reduction**
0	62.6	27.6%
1	63.2	27.0%

We conducted additional experiments for LLaMA-7B with $p = 2$. The results in Table 10 confirm that bypassing technique improves the performance on all LLaMA-7Bs with different p values.