Accurate Sublayer Pruning for Large Language Models by Exploiting Latency and Tunability Information

Seungcheol Park[1] , **Sojin Lee**[1] , **Jongjin Kim**[1] , **Jinsik Lee**[2] , **Hyunjik Jo**[2] and **U Kang**[1]
[1]Seoul National University
[2]LG AI Research
{ant6si, lsjlsj5846, j2kim99, ukang}@snu.ac.kr[1], {jinsik.lee,hyunjik.jo}@lgresearch.ai[2]

Abstract

How can we accelerate large language models (LLMs) without sacrificing accuracy? The slow inference speed of LLMs hinders us to benefit from their remarkable performance in diverse applications. This is mainly because numerous sublayers are stacked together in LLMs. Sublayer pruning compresses and expedites LLMs via removing unnecessary sublayers. However, existing sublayer pruning algorithms are limited in accuracy since they naively select sublayers to prune, overlooking the different characteristics of each sublayer.

In this paper, we propose SPRINT (Sublayer PRuning wIth LateNcy and Tunability Information), an accurate sublayer pruning method for LLMs. SPRINT accurately selects a target sublayer to prune by considering 1) the amount of latency reduction after pruning and 2) the tunability of sublayers. SPRINT iteratively prunes redundant sublayers and swiftly tunes the parameters of remaining sublayers. Experiments show that SPRINT achieves the best accuracy-speedup trade-off, exhibiting up to 23.88%p higher accuracy on zero-shot commonsense reasoning benchmarks compared to existing pruning algorithms.

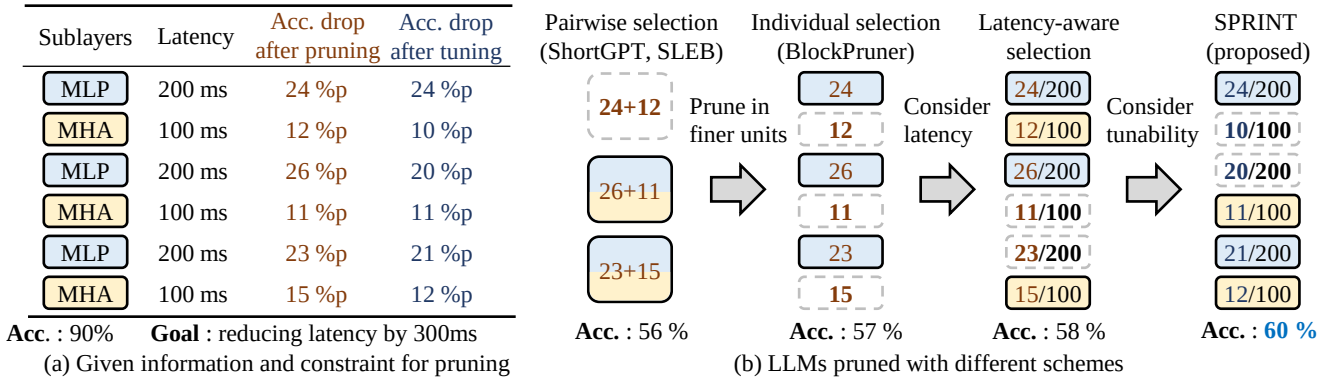
1 Introduction

How can we accelerate large language models (LLMs) without sacrificing accuracy? Recent LLMs have shown impressive performance across various tasks such as translation, code completion, and personal assistant [Brown *et al.*, 2020; Zhang *et al.*, 2022; Team *et al.*, 2023; Chowdhery *et al.*, 2023; Touvron *et al.*, 2023a; Touvron *et al.*, 2023b; Research *et al.*, 2024]. The vast number of parameters in LLMs enables the remarkable capabilities, but slows down the inference speed of LLMs, limiting their practical deployments. Hence, accelerating LLMs is essential to fully leverage their benefits.

Pruning compresses and expedites neural network models via removing unnecessary parameters [Park *et al.*, 2024b; Lee *et al.*, 2021]. LLMs consist of multi-head attention (MHA) and multi-layer perceptron (MLP) sublayers stacked alternatingly. Sublayer pruning identifies unimportant sublayers, and removes them [Song *et al.*, 2024; Men *et al.*, 2024; Zhong *et al.*, 2024]. Note that sublayers are sequentially calculated unlike smaller units such as attention heads or neurons which are processed in parallel [Ma *et al.*, 2023; Ashkboos *et al.*, 2024]. Thus, sublayer pruning algorithms display better accuracy-speedup trade-off than finer-grained ones since removing parallelizable computations prevents GPUs from fully utilizing their computational capability.

The core objective of sublayer pruning is to accurately identify the sublayers to prune. However, existing sublayer pruning algorithms face challenges in preserving the accuracy of a pruned LLM since they fail to consider the characteristics of each sublayer. Figure 1 illustrates the results of different schemes to prune sublayers of an LLM. All the schemes prune less important sublayers to reduce the latency of the model by 300ms. Figure 1(a) summarizes the attributes of each sublayer, and (b) shows how each scheme evaluates the importance of each sublayer. Pairwise selection algorithms [Song *et al.*, 2024; Men *et al.*, 2024] prune MHA and MLP sublayers in pairs, to reduce the number of importance evaluations for identifying the targets. Those algorithms face challenges in preserving the accuracy of the pruned LLM since MHA sublayers induce less accuracy drop than MLP sublayers do when pruned. Individual selection algorithms [Zhong *et al.*, 2024], which evaluate and prune each sublayer separately, still fail to achieve the best accuracy for the following two reasons. First, they overlook the latency difference between MHA and MLP sublayers. They prioritize MHA sublayers for removal due to their seemingly lower impact on accuracy. However, removing a single MLP sublayer provides equivalent latency reduction to removing two MHA sublayers, and they do not consider this effect for accelerating LLMs. Second, they disregard the fact that the accuracy drop caused by pruning is changed after tuning. They select sublayers solely based on the damage before tuning, failing to find sublayers that cause lower damage after tuning. To overcome these limitations, a pruning algorithm needs to consider both the latency and tunability of each sublayer.

We propose SPRINT (Sublayer PRuning wIth LateNcy and Tunability Information), an accurate sublayer pruning method for LLMs. SPRINT preserves the accuracy of the pruned models via accurately selecting a sublayer to remove. As shown in Figure 1(b), SPRINT considers the amount of latency reduction after pruning and accuracy drop after tuning. Moreover, we minimize the cost of SPRINT by 1) ac-



(a) Given information and constraint for pruning

(b) LLMs pruned with different schemes

Figure 1: (a) Comparison of different schemes in pruning for large language models, given the characteristics of each sublayer. Each scheme prunes the least important sublayers judged by its own importance measure. (b) Among all schemes, SPRINT achieves the best accuracy under the same latency constraint by considering both latency and tunability.

tivation checkpointing to mitigate the repetitive computations, and 2) fast candidate selection to reduce the number of time-intensive tuning. We verify the effectiveness of SPRINT with extensive experiments. SPRINT accelerates Llama-2 and Llama-3 models, achieving up to 23.88%p higher accuracy on zero-shot commonsense reasoning benchmarks than baselines, and shows the best accuracy-speedup trade-off.

We summarize our main contributions as follows:

- **Algorithm.** We propose SPRINT, an accurate sublayer pruning method for LLMs. SPRINT accurately identifies less important sublayers in a model by the following four effective techniques: 1) latency-aware importance scoring, 2) tunability-aware sensitivity evaluation, 3) activation checkpointing, and 4) fast candidate selection.
- **Experiments.** We demonstrate that SPRINT achieves the state-of-the-art performance on commonsense reasoning benchmarks. SPRINT accelerates Llama-2 and Llama-3 models, achieving up to 23.88%p higher accuracy on zero-shot commonsense reasoning benchmarks, showing the most favorable accuracy-speedup trade-off.
- **Analysis.** We analyze the pruning patterns of the models pruned by SPRINT. We derive the findings that MLP and lower sublayers in an LLM serve as a critical component of LLM's capabilities, while MHA and upper sublayers contribute less to accuracy.

The rest of this paper is organized as follows. We first define LLM acceleration problem and provide backgrounds. We then propose SPRINT, our pruning method. After presenting experimental results, we conclude. Our source code is available at https://github.com/snudm-starlab/SPRINT.

2 Preliminary

2.1 Problem Definition

Problem 1 (LLM Acceleration Problem)**.** *Given a pretrained large language model F, a sample dataset $\mathcal{D}$, and a latency upper bound τ, the problem is to find an accurate model $\widehat{F}$ whose latency does not exceed τ.* □

2.2 Transformer Architecture

Recent LLMs [Touvron *et al.*, 2023b; Dubey *et al.*, 2024] have a Transformer-based architecture [Vaswani *et al.*, 2017] which consists of multi-head attention (MHA) and multi-layer perceptron (MLP) sublayers stacked alternatingly. A Transformer model F with S sublayers refines an input sequence vector $\boldsymbol{x}$ as in Equation (1).

$$F(\boldsymbol{x}) = \mathcal{G}\left(\left(\circ_{s=1}^{S}(f^{(s)} + I)\right)\left(\mathcal{E}(\boldsymbol{x})\right)\right) \tag{1}$$

$\mathcal{G}$ is a generator module, $\mathcal{E}$ is an embedding look-up table, I represents a residual connection, and $\circ$ is the composition of functions. $f^{(s)}$ denotes the sth sublayer function which is either a self-attention network in an MHA sublayer when s is an odd number, or a feed-forward network in an MLP sublayer when s is an even number. We decompose $f^{(s)}$ into output projection $\boldsymbol{W}^{(s)}$ and remainder $h^{(s)}(\cdot)$ as in Equation (2).

$$f^{(s)}(\boldsymbol{X}^{(s)}) = \boldsymbol{W}^{(s)} h^{(s)}(\boldsymbol{X}^{(s)}) \tag{2}$$

$\boldsymbol{X}^{(s)}$ is an input matrix for the sth sublayer, where $\boldsymbol{X}^{(1)} = \mathcal{E}(\boldsymbol{x})$. Given an intermediate activation $\boldsymbol{Z}^{(s)} = h^{(s)}(\boldsymbol{X}^{(s)})$, $f^{(s)}$ is a linear transformation function with regard to $\boldsymbol{Z}^{(s)}$.

2.3 Sublayer Pruning

Sublayer pruning [Song *et al.*, 2024; Men *et al.*, 2024; Zhong *et al.*, 2024] accelerates LLMs via pruning unnecessary sublayers in them. Sublayer pruning algorithms measure an importance score η for each sublayer, and eliminate those with the lowest scores. The importance scoring leverages sensitivity ζ which represents the performance difference of a model before and after the pruning. It is crucial to accurately score the importance of sublayers since the accuracy of the model heavily depends on which sublayers are pruned.

2.4 Fast In-compression Tuning

Pruning causes the accuracy loss by repeatedly removing parameters from a model. To mitigate the accuracy degradation, tuning the pruned model is essential so that the inference results are similar to those of the uncompressed model [Park *et al.*, 2024a; Park *et al.*, 2024b]. The objective of the tuning

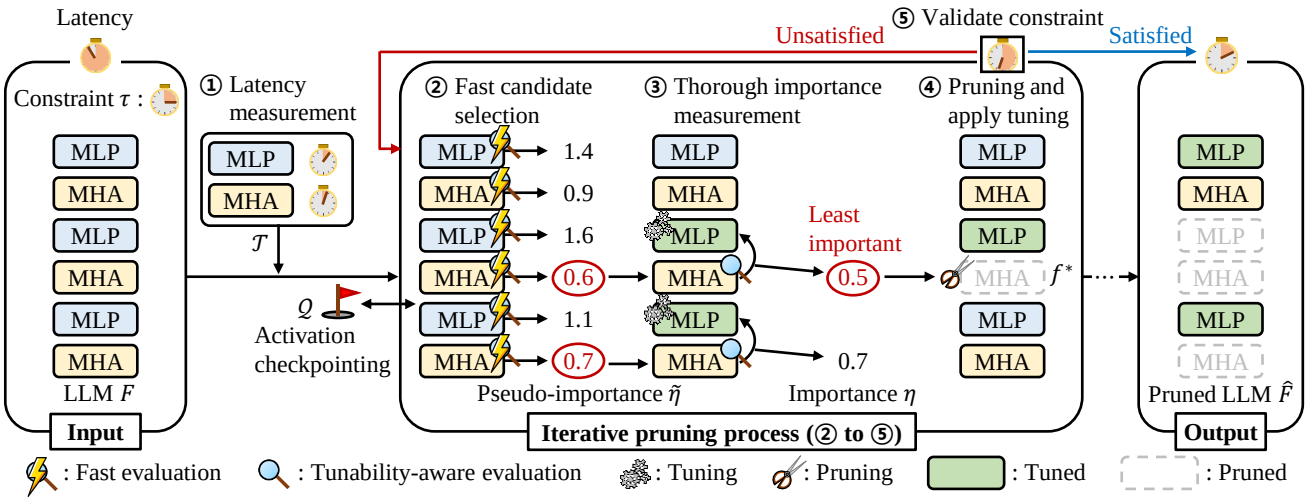


Figure 2: An illustration of the overall process of SPRINT. Given a pretrained LLM and a latency constraint, SPRINT iteratively identifies and prunes the least important sublayer until the pruned model satisfies the latency constraint. SPRINT accurately selects the sublayer to prune by considering the latency and tunability information of sublayers.

is to align the output of the sth sublayer in the pruned model with that in the unpruned model, as described in Equation (3).

$$\arg\min_{\widehat{\boldsymbol{W}}^{(s)}} \|(\widehat{\boldsymbol{X}}^{(s)} + \widehat{\boldsymbol{W}}^{(s)}\widehat{\boldsymbol{Z}}^{(s)}) - \boldsymbol{X}^{(s+1)}\|_F^2, \tag{3}$$

where $\widehat{\boldsymbol{X}}^{(s)}$, $\widehat{\boldsymbol{W}}^{(s)}$, and $\widehat{\boldsymbol{Z}}^{(s)}$ are the input, output projection, and intermediate representation of the sth sublayer in the pruned model, respectively. $\widehat{\boldsymbol{X}}^{(s+1)} = \widehat{\boldsymbol{X}}^{(s)} + \widehat{\boldsymbol{W}}^{(s)}\widehat{\boldsymbol{Z}}^{(s)}$ is the output of sth sublayer in the pruned model, while $\boldsymbol{X}^{(s+1)}$ is the output of the sth sublayer in the unpruned model.

Note that solving Equation (3) does not require time-consuming stochastic gradient descents as in previous works [Hu *et al.*, 2022; Xu *et al.*, 2024]. Instead, the equation is efficiently solved incorporating PyTorch's solver (torch.linalg.lstsq) once $\widehat{\boldsymbol{X}}^{(s)}$, $\widehat{\boldsymbol{Z}}^{(s)}$, and $\boldsymbol{X}^{(s+1)}$ are computed. Thus, fast in-compression tuning is computationally affordable for iterative pruning algorithms.

3 Proposed Method

3.1 Overview

We address the following challenges to prune sublayers in LLMs minimizing the loss of accuracy.

- C1. **Latency Difference of Sublayers.** Existing works ignore the latency difference of MHA and MLP sublayers. How can we compare the importance of sublayers with different latencies to effectively accelerate LLMs?
- C2. **Ignoring the Impact of Tuning.** Existing works ignore the impact of tuning when selecting sublayers to prune, resulting in misselection. How can we incorporate the impact of tuning to accurately select sublayers to prune?
- C3. **Expensive Computational Cost.** Sublayer pruning is computationally expensive since it repeatedly measures the importance scores of all sublayers at each iteration. How can we enhance the efficiency of sublayer pruning?

Algorithm 1 Overall process of SPRINT

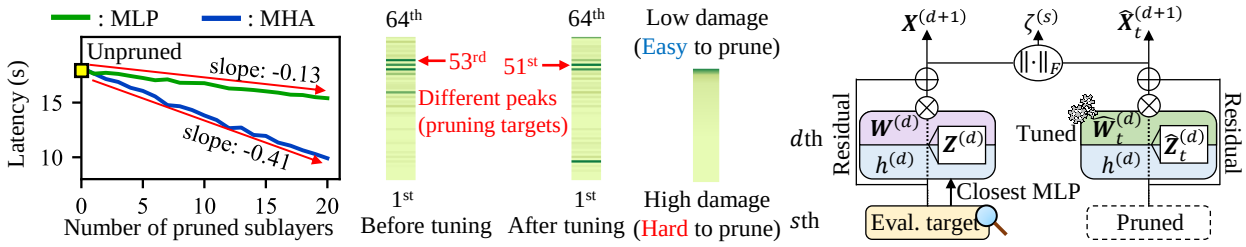
Input: An LLM F, a calibration dataset $\mathcal{D}$, a latency constraint τ, number α of checkpoints, and number β of candidates
Output: A pruned LLM $\widehat{F}$

1: Initialize $\widehat{F}$ as F
2: Initialize a dictionary $\mathcal{Q}$ of α checkpoints
3: Measure latencies $\mathcal{T} = \{t^{(MHA)}, t^{(MLP)}\}$
4: **while** latency$(\widehat{F}) > \tau$ **do**
5: $\quad\mathcal{C}, \mathcal{Q} \leftarrow$ fast_candidate_selection$(\widehat{F}, \mathcal{D}, \mathcal{T}, \beta, \mathcal{Q})$
$\quad\quad\triangleright$ Select a set $\mathcal{C}$ of candidates (Section 3.4)
6: $\quad f^* \leftarrow$ tunability_aware_target_selection$(\mathcal{C}, \widehat{F}, \mathcal{D}, \mathcal{T}, \mathcal{Q})$
$\quad\quad\triangleright$ Find the least important sublayer f^* (Section 3.3)
7: $\quad$Remove f^* from $\widehat{F}$ and apply tuning
8: **end while**
9: **return** $\widehat{F}$

We propose SPRINT to address these challenges. The main ideas of SPRINT are as follows.

- I1. **Latency-aware Importance Scoring.** We consider the amount of latency reduction after pruning each sublayer to precisely identify unimportant sublayers.
- I2. **Tunability-aware Sensitivity Evaluation.** We accurately select sublayers to prune by measuring their sensitivity after tuning.
- I3. **Avoiding Unnecessary Computations.** We propose activation checkpointing and fast candidate selection to avoid the unnecessary computations.

Algorithm 1 and Figure 2 show the overall process of SPRINT. Given a pretrained LLM F and a latency constraint τ, SPRINT returns the pruned LLM $\widehat{F}$ satisfying the latency constraint. SPRINT initializes α activation checkpoints $\mathcal{Q}$ to store reusable activations (line 2, details in Section 3.4). Then, SPRINT measures the amounts $\mathcal{T}$ of latency reduction resulting from the removal of sublayers (line 3) for latency-aware



(a) Latency gains (b) Effect of tuning (c) Sensitivity measurement

Figure 3: (a) Latency change after pruning MHA and MLP sublayers in Llama-3 8B model. MHA sublayers impact more than MLP sublayers after pruning. (b) The rankings of sublayers according to the amount of accuracy loss caused by pruning, before and after tuning. Tuning impacts the rankings, changing pruning targets. (c) An illustration of tunability-aware sensitivity measurement process. SPRINT performs fast in-compression tuning on the closest upper MLP sublayer while measuring sensitivities.

importance scoring (details in Section 3.2). SPRINT repeats the iterative process of 1) scoring the importance of sublayers and 2) removing the least important one (lines 4-8) until the latency constraint is met. For each iteration, SPRINT first selects β candidate sublayers $\mathcal{C}$ to prune by scoring the pseudo-importance of each sublayer (line 5, see Section 3.4), and updates the checkpoints $\mathcal{Q}$. SPRINT then scores the importance of each candidate in $\mathcal{C}$ with tunability-aware sensitivity evaluation (line 6, details in Section 3.3). Based on the importance scores, SPRINT prunes the least important sublayer f^* and tunes the remaining model (line 7). SPRINT returns the pruned model $\widehat{F}$ (line 9) which satisfies the constraint.

3.2 Latency-aware Importance Scoring

Observation. How can we identify the most appropriate sublayer to prune for accelerating LLMs with minimal accuracy loss? Existing works [Song *et al.*, 2024; Zhong *et al.*, 2024; Men *et al.*, 2024] measure the importance of sublayers without considering the latency difference of sublayers. Figure 3(a) shows the latencies of Llama-3 8B models after pruning different numbers of MHA and MLP sublayers. As shown in the figure, pruning an MHA sublayer yields over three times greater latency reduction than pruning an MLP sublayer; thus, it is beneficial to prune an MHA sublayer instead of an MLP sublayer if they cause the same damage. Therefore, it is essential to consider latency when selecting sublayers to prune.

Our solution. We incorporate the latencies of sublayers into our importance scoring process and assign lower importance scores for the sublayers with higher latencies to promote pruning the high-latency sublayers. The importance $\eta^{(s)}$ of the sth sublayer is defined as follows:

$$\eta^{(s)} = \zeta^{(s)} / t^{(s)}, \tag{4}$$

where $\zeta^{(s)}$ is the sensitivity of the sth sublayer which approximates the amount of accuracy degradation after pruning it. $t^{(s)}$ is the amount of reduced latency through pruning the sth sublayer. Hence, $\eta^{(s)}$ reflects the cost-effectiveness of the sth sublayer in contributing to accuracy. As shown in Figure 3(a), sublayers of the same type offer almost the same degree of latency reduction; $t^{(s)}$ is either $t^{(MHA)}$ or $t^{(MLP)}$ depending on the sublayer's type. SPRINT measures $t^{(MHA)}$ and $t^{(MLP)}$ by comparing the latencies of unpruned and partially pruned models before starting its iterative pruning process.

3.3 Tunability-aware Sensitivity Evaluation

Observation. How can we accurately estimate the sensitivities of sublayers? Sublayer pruning algorithms measure sensitivities to approximately estimate the accuracy loss after pruning each sublayer. The lost accuracy is recovered via tuning, and each sublayer has a different capability for recovering. However, existing works [Men *et al.*, 2024; Song *et al.*, 2024; Zhong *et al.*, 2024] ignore the effect of tuning when estimating the sensitivities. Figure 3(b) compares the ranking in accuracy degradation after pruning each sublayer before and after tuning. As shown in the figure, the index of the peak sublayer that evokes the lowest damage is changed after tuning. This indicates that pruning sublayers with the lowest sensitivity without considering the effect of tuning removes useful sublayers that exhibit low accuracy degradation after tuning. Therefore, sublayer pruning algorithms must account for the effect of tuning by prioritizing the removal of sublayers that result in minimal accuracy drop after tuning.

Our solution. SPRINT compares the activations of the original model and the pruned model after tuning to measure the sensitivities of sublayers. SPRINT exploits the fast in-compression tuning [Park *et al.*, 2024a] in Section 2.4 to efficiently incorporate tunability information into the sublayer selection process. Figure 3(c) shows the sensitivity measurement process of SPRINT for the sth sublayer which we call as the evaluation target. SPRINT finds the closest MLP sublayer (dth sublayer in the Figure 3(c)) above the evaluation target. After that, SPRINT measures the sensitivity $\zeta^{(s)}$ of the sth sublayer by computing the normalized distance between outputs $\boldsymbol{X}^{(d+1)}$ and $\widehat{\boldsymbol{X}}_t^{(d+1)}$ of the MLP sublayer before and after pruning, respectively, as in Equation (5). We exploit $\widehat{\boldsymbol{X}}_t^{(d+1)}$ obtained by fast in-compression tuning to take the tunability into account.

$$\zeta^{(s)} = ||\boldsymbol{X}^{(d+1)} - \widehat{\boldsymbol{X}}_t^{(d+1)}||_F / ||\boldsymbol{X}^{(d+1)}||_F, \tag{5}$$

Note that we use outputs of only MLP sublayers, since an MLP sublayer has three times more number of parameters than that of an MHA sublayer, and thus has a stronger tuning capability than MHA. For fast in-compression tuning, SPRINT finds $\widehat{\boldsymbol{W}}_t^{(d)}$ that minimizes the distance between the output $\widehat{\boldsymbol{X}}^{(d+1)} = (\widehat{\boldsymbol{X}}^{(d)} + \widehat{\boldsymbol{W}}^{(d)} \widehat{\boldsymbol{Z}}^{(d)})$ after pruning and the

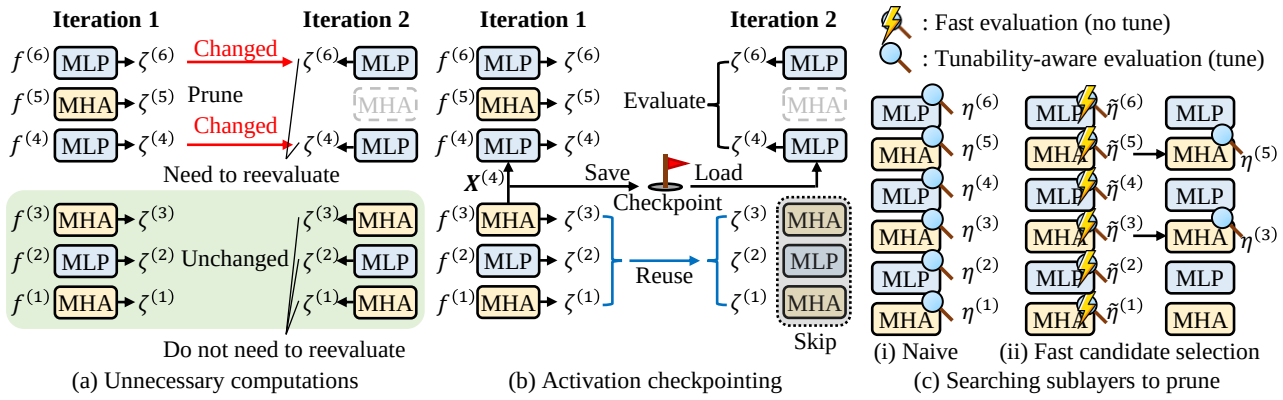


Figure 4: (a) Unnecessary computations in iterative sublayer pruning. The sensitivity of a sublayer is unchanged if its closest upper MLP sublayer is below the pruned sublayer. (b) An illustration of activation checkpointing. SPRINT avoids unnecessary computations by caching sensitivities and activations in the previous iterations. (c) Comparison between the naive approach and the fast candidate selection to find the sublayer to prune. The fast candidate selection reduces the number of layers to tune.

output $\boldsymbol{X}^{(d+1)}$ before pruning, as in Equation (3). $\widehat{\boldsymbol{W}}^{(d)}$ and $\widehat{\boldsymbol{Z}}^{(d)}$ are the weight of the out projection and the intermediate activation of the dth sublayer after pruning, respectively. Note that each row of $\widehat{\boldsymbol{W}}^{(d)}$ forms an independent subproblem and we tune the weights in only $c\%$ of rows to avoid overfitting. We select the rows with abundant outliers, which represent larger activations than others, to maximize the impact of tuning with the given percentage of rows. We exploit the sum of the outlier-aware weight-wise scores [Sun *et al.*, 2023; Yin *et al.*, 2024] of weights in each row for selection. We save the tuned weights of each sublayer during the importance scoring process and apply the tuned weights corresponding to the pruning of the least important sublayer.

3.4 Avoiding Unnecessary Computations

Observation 1. How can we minimize the computation for measuring the sensitivities of sublayers? Pruning a sublayer does not affect the sensitivity of other sublayers whose closest upper MLP sublayers are located below the pruned one. For instance, as shown in Figure 4(a), $\zeta^{(1)}$ to $\zeta^{(3)}$ do not need to be reevaluated if $f^{(5)}$ is pruned in the previous iteration. However, naively computing the sensitivities of all sublayers at each iteration entails redundant computation for sensitivities which are already obtained in the previous iteration.

Our solution 1 (Activation Checkpointing). We propose activation checkpointing to prevent unnecessary recomputations. Before starting the iterations, SPRINT places checkpoints between sublayers. At each iteration, SPRINT stores the activations at the checkpoints. SPRINT reuses the sensitivities from the previous iteration for each sublayer f whose closest upper MLP sublayer is beneath the pruned sublayer, since the sensitivity of f is not changed after pruning. Then, SPRINT updates the sensitivities of the remaining sublayers using the stored activation. For example, assume we pruned $f^{(5)}$ at iteration 1, as shown in Figure 4(b). Note that SPRINT reuses $\zeta^{(1)}$ to $\zeta^{(3)}$ from the iteration 1 in the iteration 2 since they are not changed. Then, SPRINT loads the stored activation $\boldsymbol{X}^{(4)}$ and starts updating the sensitivities from $f^{(4)}$. The checkpoints are uniformly placed, and the number α of checkpoints controls the trade-off between the memory usage and the time consumption during pruning.

Observation 2. How can we minimize the cost of sensitivity measurement? SPRINT performs an in-compression tuning to evaluate the sensitivity of each sublayer. Naively computing the sensitivities of all sublayers as illustrated in Figure 4(c-i) leads to excessive number of tunings at each iteration, making the sensitivity measurement too expensive.

Our solution 2 (Fast Candidate Selection). We propose fast candidate selection to selectively measure the sensitivities, minimizing the number of tuning in the sensitivity measurement. Instead of evaluating tunability-aware sensitivities for all sublayers, SPRINT first selects the candidates of the least sensitive sublayers swiftly without tuning. Then, SPRINT measures the tunability-aware sensitivity only for the candidate sublayers. For example, in Figure 4(c-ii), SPRINT first finds two candidates without tuning and then selects the sublayer to prune with the tunability-aware evaluation, reducing the number of tunings from 6 to 2. This process can be viewed as reducing the computational costs by approximately finding the sublayer to prune.

To instantly select the candidate sublayers, SPRINT measures the pseudo-importance of each sublayer.The pseudo-sensitivity $\tilde{\zeta}^{(s)}$ of sth sublayer is the normalized distance between outputs $\boldsymbol{X}^{(d+1)}$ and $\widehat{\boldsymbol{X}}^{(d+1)}$ of the MLP sublayer before and after pruning, respectively. Note that $\widehat{\boldsymbol{X}}^{(d+1)}$ is obtained without tuning, unlike $\widehat{\boldsymbol{X}}_t^{(d+1)}$ in Equation (5). The pseudo-importance $\tilde{\eta}^{(s)}$ of the sth sublayer is $\tilde{\zeta}^{(s)}/t^{(s)}$, where $t^{(s)}$ is the latency of the sublayer. SPRINT selects β sublayers with the least pseudo-importance scores, where β is a hyperparameter representing the number of candidates. A higher β leads to the more accurate search for the sublayer to prune while requiring a higher computational cost.

4 Experiments

We perform experiments to answer the following questions.

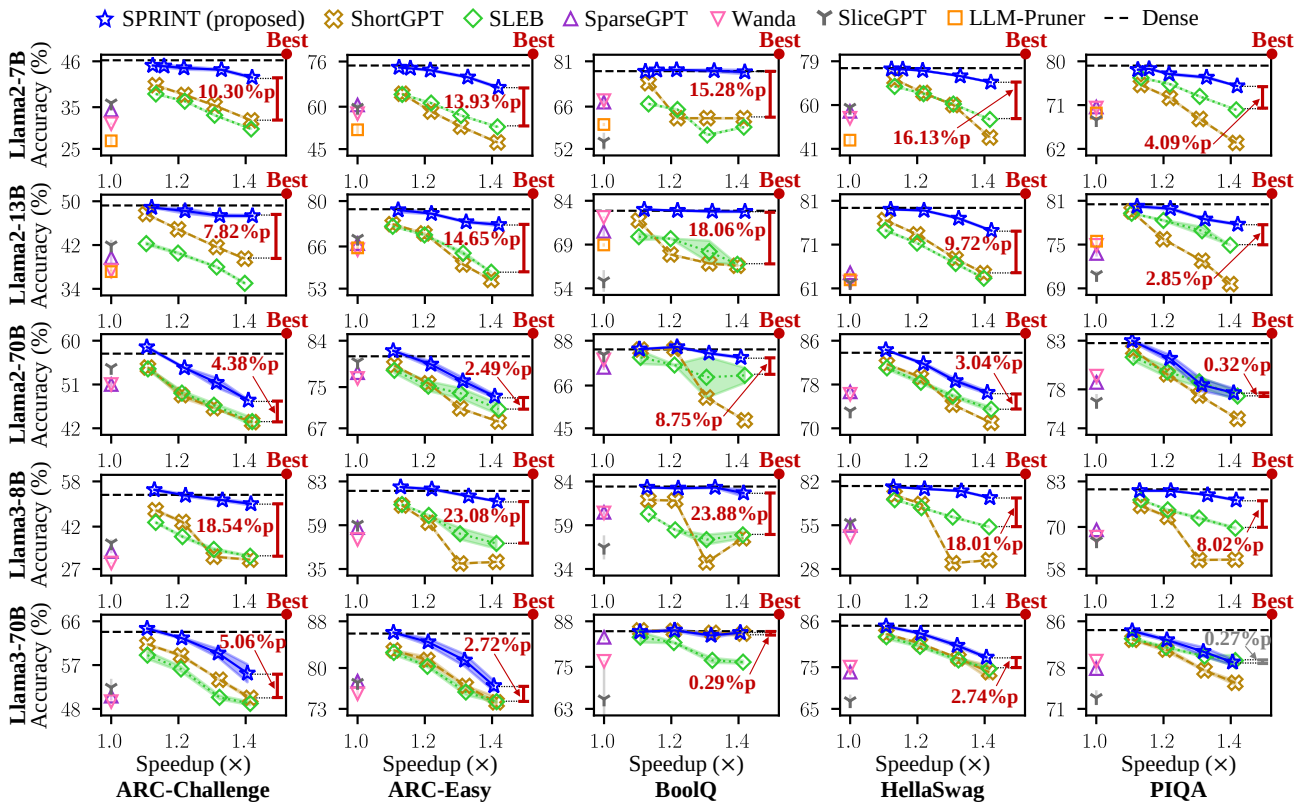


Figure 5: Accuracy-speedup trade-off curves of SPRINT and competitors. SPRINT shows the best trade-off among all the methods.

Q1. **Accuracy.** How accurate is SPRINT compared to baselines with the similar acceleration level?

Q2. **Pruning Efficiency.** How fast does SPRINT prune LLMs compared to baselines?

Q3. **Ablation Study.** Does each main idea of SPRINT contribute to the performance?

Q4. **Pruning Pattern Analysis.** Which sublayers are important to maintain the accuracy of LLMs?

4.1 Experimental Setup

Setup. We use Llama-2 [Touvron *et al.*, 2023b] and Llama-3 [Dubey *et al.*, 2024] model families as pruning targets. We randomly sample 128 token sequences of length 2048 from Wikitext2 [Merity *et al.*, 2016] dataset for sensitivity measurement and tuning. We use NVIDIA A100 80GB GPU for all experiments. We report zero-shot reasoning accuracies on ARC-Challenge, ARC-Easy [Clark *et al.*, 2018], BoolQ [Clark *et al.*, 2019], HellaSwag [Zellers *et al.*, 2019], and PIQA [Bisk *et al.*, 2020] benchmarks. We measure the latencies to generate 512 tokens from 1024 input tokens [Lin *et al.*, 2024b] and report speedups of pruned models.

Baselines. We compare SPRINT with three sublayer pruning algorithms: ShortGPT [Men *et al.*, 2024], SLEB [Song *et al.*, 2024], and BlockPruner [Zhong *et al.*, 2024]. ShortGPT and SLEB utilize the pairwise selection scheme while BlockPruner selects pruning targets individually as illustrated in Figure 1. We also include four fine-grained pruning algorithms as baselines for comprehensive analysis: SparseGPT [Frantar and Alistarh, 2023], Wanda [Sun *et al.*, 2023], SliceGPT [Ashkboos *et al.*, 2024], and LLM-Pruner [Ma *et al.*, 2023]. These algorithms prune LLMs in parallelizable units smaller than sublayers such as channels or attention heads, resulting in insufficiently accelerated models.

Hyperparameters. We use random seeds from 0 to 2 and report the average values. We set the checkpointing hyperparameter α to 8 and the candidate hyperparameter β to 5 for all models.

4.2 Accuracy

Figure 5 shows the accuracies of SPRINT and baselines across various inference speedup settings. As shown in the figure, SPRINT achieves the best trade-off curve among all the methods, and significantly outperforms the competitors under 40% speedup setting with the maximum accuracy gap of 23.88%p. It is notable that SPRINT achieves the best performance among all the algorithms on challenging multiple-choice benchmarks, such as ARC-Challenge, ARC-Easy, and HellaSwag, across all models of various sizes.

4.3 Pruning Efficiency

Figure 6 visualizes compression time and average accuracy on five commonsense reasoning tasks of the pruned models generated by SPRINT and competitors with 40% speedup. We estimate the pruning time of BlockPruner for 70B models since it takes more than a week to prune them (see Supplementary Material for details). Brown dashed lines in Figure 6 denote the estimated pruning time.

Note that SPRINT offers the best accuracy-pruning time trade-off across all models. SPRINT prunes the LLM up to

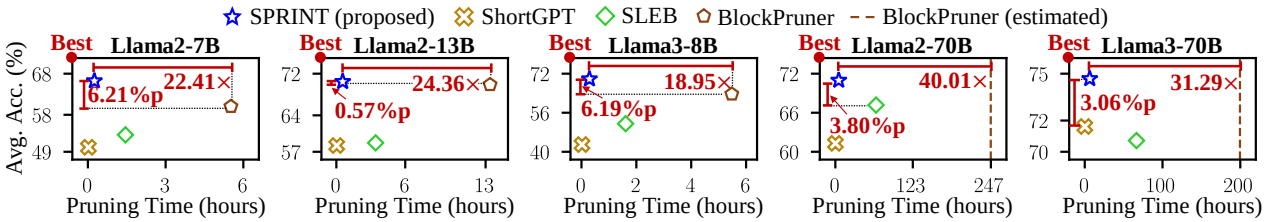


Figure 6: Pruning time and accuracy trade-off of SPRINT and competitors under 1.4× acceleration. SPRINT is closest to the "Best" point with the highest accuracy and short pruning time.

Method	SPRINT-l	SPRINT-t	SPRINT-e	SPRINT
Acc. (%)	66.96	67.62	69.82	69.82
Time (s)	1070.29	212.00	2147.28	1052.21

Table 1: Performance of SPRINT and its variants for accelerating Llama-3 8B by 1.4×. See Section 4.4 for details.

40.01 times faster and achieves up to 6.21%p higher accuracy than BlockPruner. SPRINT further outperforms other methods with similar pruning times by larger margins.

4.4 Ablation Study

To prove the effectiveness of each main idea, we compare SPRINT with its three variants: SPRINT-l, SPRINT-t, and SPRINT-e. SPRINT-l is SPRINT without latency-aware importance scoring; it prunes the sublayer with the lowest sensitivity without considering the latency. SPRINT-t is SPRINT without tunability-aware sensitivity evaluation, measuring the sensitivity of each sublayer by directly comparing its output before and after pruning. SPRINT-e is SPRINT without activation checkpointing and fast candidate selection, calculating the importance scores of all sublayers without storing activations for each step of iterative pruning process.

Table 1 summarizes the performance of SPRINT and its variants when accelerating the Llama-3 8B by 40%. SPRINT and SPRINT-e outperform other variants in terms of accuracy, proving that both latency-aware importance scoring and tunability-aware sensitivity evaluation contribute to the accuracy. SPRINT is twice faster than SPRINT-e, confirming the effectiveness of activation checkpointing and fast candidate selection for the pruning cost. In summary, all three main ideas of SPRINT contributes to the performance.

4.5 Pruning Pattern Analysis

Figure 7 depicts the pruning patterns of SPRINT on Llama models. Orange, blue, and gray squares represent MHA, MLP, and pruned sublayers, respectively. We observe two main patterns related to the type and position of sublayers. First, SPRINT prunes more MHA sublayers than MLP sublayers in general; MLP sublayers are pruned only in 70B models. Second, SPRINT prunes sublayers located mainly in the upper-middle parts of models. These two patterns show that MLP sublayers and sublayers located near the bottom significantly contribute to the capability of LLMs. Moreover, as more extensive pruning occurs in the upper layers as observed by the second pattern, sensitivities of lower layers often do not need to be updated. Thus, our proposed activation checkpointing aligns well with the LLM's characteristics.

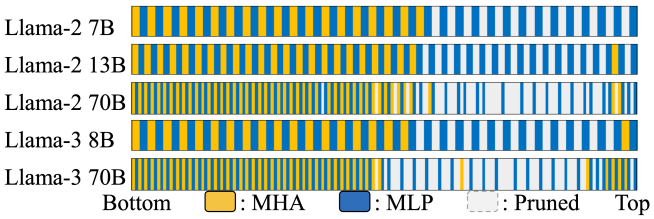


Figure 7: Pruning patterns of SPRINT on Llama models (best viewed in color). SPRINT primarily prunes MHA sublayers located between the middle and upper parts of the model.

5 Related Work

We review techniques for accelerating LLMs: quantization, knowledge distillation, and dynamic inference. Quantization [Piao *et al.*, 2022; Frantar *et al.*, 2023; Lee *et al.*, 2023; Lin *et al.*, 2024a; Shao *et al.*, 2024; Kim *et al.*, 2025b; Kim *et al.*, 2025a] reduces the bit-width of weights and activations in them; it accelerates computation by leveraging hardwares designed for low-bit operations. Quantization is compatible with pruning; unifying both methods achieves greater acceleration [Frantar and Alistarh, 2023].

Knowledge distillation (KD) [Ko *et al.*, 2024; Yoo *et al.*, 2019; Cho and Kang, 2022; Liu *et al.*, 2024; Kim *et al.*, 2021; Jeon *et al.*, 2023; Jang *et al.*, 2023] improves the accuracy of compressed models by transferring knowledge from uncompressed models. KD is also compatible with pruning, by effectively compensating for the error induced by pruning.

Dynamic inference [Schuster *et al.*, 2022; Varshney *et al.*, 2023; Raposo *et al.*, 2024; Fu *et al.*, 2024] accelerates LLMs via dynamically adjusting the amount of computations based on the input. Dynamic inference exhibits minimal accuracy degradation since it determines the amount of computations according to the importance of inputs. However, they have a significant drawback in that their efficiency is diminished when multiple inputs requiring different computations are fed [Song *et al.*, 2024]. In contrast, sublayer pruning consistently preserves efficiency regardless of the number of inputs.

6 Conclusion

We propose SPRINT, an accurate sublayer pruning method for accelerating LLMs. SPRINT addresses the inaccurate sublayer selection problem of existing sublayer pruning methods by factoring in latency and tunability information. We propose activation checkpointing and fast candidate selection techniques to shorten the running time of SPRINT. We demonstrate that SPRINT achieves the best accuracy-speedup trade-off when pruning Llama-2 and Llama-3 models.

Contribution Statement

Seungcheol Park, Sojin Lee, and Jongjin Kim contributed equally to this work as co-first authors. Jinsik Lee and Hyunjik Jo provided helpful discussion for developing our method. U Kang supervised the project and carefully reviewed the manuscript.

Acknowledgements

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) [No.RS-2020-II200894, Flexible and Efficient Model Compression Method for Various Applications and Environments], [No.RS-2021-II211343, Artificial Intelligence Graduate School Program (Seoul National University)], and [No.RS-2021-II212068, Artificial Intelligence Innovation Hub (Artificial Intelligence Institute, Seoul National University)]. This work was supported by Youlchon Foundation. The Institute of Engineering Research at Seoul National University provided research facilities for this work. The ICT at Seoul National University provides research facilities for this study. U Kang is the corresponding author. This work was improved by the helpful input and collaboration of researchers from LG AI Research.

References

[Ashkboos *et al.*, 2024] Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. *arXiv*, 2024.

[Bisk *et al.*, 2020] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *AAAI*, 2020.

[Brown *et al.*, 2020] 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. In *NeurIPS*, 2020.

[Cho and Kang, 2022] Ikhyun Cho and U Kang. Pea-kd: Parameter-efficient and accurate knowledge distillation on bert. *PLOS ONE*, 17(2), 2022.

[Chowdhery *et al.*, 2023] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[Clark *et al.*, 2018] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv*, 2018.

[Clark *et al.*, 2019] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv*, 2019.

[Dubey *et al.*, 2024] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv*, 2024.

[Frantar and Alistarh, 2023] Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *ICML*, 2023.

[Frantar *et al.*, 2023] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. OPTQ: Accurate quantization for generative pre-trained transformers. In *ICLR*, 2023.

[Fu *et al.*, 2024] Qichen Fu, Minsik Cho, Thomas Merth, Sachin Mehta, Mohammad Rastegari, and Mahyar Najibi. Lazyllm: Dynamic token pruning for efficient long context llm inference. *arXiv*, 2024.

[Hu *et al.*, 2022] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022.

[Jang *et al.*, 2023] Jun-Gi Jang, Chun Quan, Hyun Dong Lee, and U Kang. Falcon: lightweight and accurate convolution based on depthwise separable convolution. *Knowl. Inf. Syst.*, 65(5):2225–2249, 2023.

[Jeon *et al.*, 2023] Hyojin Jeon, Seungcheol Park, Jin-Gee Kim, and U. Kang. Pet: Parameter-efficient knowledge distillation on transformer. *PLOS ONE*, 18(7), 2023.

[Kim *et al.*, 2021] Junghun Kim, Jinhong Jung, and U. Kang. Compressing deep graph convolution network with multi-staged knowledge distillation. *PLOS ONE*, 16, 2021.

[Kim *et al.*, 2025a] Minjun Kim, Jaehyeon Choi, Jongkeun Lee, Wonjin Cho, and U Kang. Zero-shot quantization: A comprehensive survey. In *IJCAI*, 2025.

[Kim *et al.*, 2025b] Minjun Kim, Jongjin Kim, and U Kang. Synq: Accurate zero-shot quantization by synthesis-aware fine-tuning. In *ICLR*, 2025.

[Ko *et al.*, 2024] Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. Distillm: Towards streamlined distillation for large language models. In *ICML*, 2024.

[Lee *et al.*, 2021] Hyun Dong Lee, Seongmin Lee, and U. Kang. Auber: Automated bert regularization. *PLOS ONE*, 16(6), 2021.

[Lee *et al.*, 2023] Jung Hyun Lee, Jeonghoon Kim, Se Jung Kwon, and Dongsoo Lee. Flexround: Learnable rounding based on element-wise division for post-training quantization. In *ICML*, 2023.

[Lin *et al.*, 2024a] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. In *MLSys*, 2024.

[Lin *et al.*, 2024b] Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. Qserve: W4a8kv4 quantization and system co-design for efficient llm serving. *arXiv*, 2024.

[Liu *et al.*, 2024] Jiaheng Liu, Chenchen Zhang, Jinyang Guo, Yuanxing Zhang, Haoran Que, Ken Deng, Zhiqi Bai, Jie Liu, Ge Zhang, Jiakai Wang, Yanan Wu, Congnan Liu, Jiamang Wang, Lin Qu, Wenbo Su, and Bo Zheng. DDK: distilling domain knowledge for efficient large language models. In *NeurIPS*, 2024.

[Ma *et al.*, 2023] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. In *NeurIPS*, 2023.

[Men *et al.*, 2024] Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv*, 2024.

[Merity *et al.*, 2016] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv*, 2016.

[Park *et al.*, 2024a] Seungcheol Park, Hojun Choi, and U Kang. Accurate retraining-free pruning for pretrained encoder-based language models. In *ICLR*, 2024.

[Park *et al.*, 2024b] Seungcheol Park, Jaehyeon Choi, Sojin Lee, and U Kang. A comprehensive survey of compression algorithms for language models. *arXiv*, 2024.

[Piao *et al.*, 2022] Tairen Piao, Ikhyun Cho, and U Kang. Sensimix: Sensitivity-aware 8-bit index & 1-bit value mixed precision quantization for bert compression. *PloS one*, 17(4), 2022.

[Raposo *et al.*, 2024] David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv*, 2024.

[Research *et al.*, 2024] LG Research, Soyoung An, Kyunghoon Bae, Eunbi Choi, Stanley Jungkyu Choi, Yemuk Choi, Seokhee Hong, Yeonjung Hong, Junwon Hwang, Hyojin Jeon, et al. Exaone 3.0 7.8 b instruction tuned language model. *arXiv*, 2024.

[Schuster *et al.*, 2022] Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *NeurIPS*, 2022.

[Shao *et al.*, 2024] Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. In *ICLR*, 2024.

[Song *et al.*, 2024] Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. Sleb: Streamlining llms through redundancy verification and elimination of transformer blocks. *arXiv*, 2024.

[Sun *et al.*, 2023] Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv*, 2023.

[Team *et al.*, 2023] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv*, 2023.

[Touvron *et al.*, 2023a] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv*, 2023.

[Touvron *et al.*, 2023b] 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*, 2023.

[Varshney *et al.*, 2023] Neeraj Varshney, Agneet Chatterjee, Mihir Parmar, and Chitta Baral. Accelerating llama inference by enabling intermediate layer decoding via instruction tuning with lite. *arXiv*, 2023.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[Xu *et al.*, 2024] Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, Xiaopeng Zhang, and Qi Tian. Qa-lora: Quantization-aware low-rank adaptation of large language models. In *ICLR*, 2024.

[Yin *et al.*, 2024] Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay Kumar Jaiswal, Mykola Pechenizkiy, Yi Liang, Michael Bendersky, Zhangyang Wang, and Shiwei Liu. Outlier weighed layerwise sparsity (OWL): A missing secret sauce for pruning llms to high sparsity. In *ICML*, 2024.

[Yoo *et al.*, 2019] Jaemin Yoo, Minyong Cho, Taebum Kim, and U Kang. Knowledge extraction with no observable data. In *NeurIPS*, 2019.

[Zellers *et al.*, 2019] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv*, 2019.

[Zhang *et al.*, 2022] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv*, 2022.

[Zhong *et al.*, 2024] Longguang Zhong, Fanqi Wan, Ruijun Chen, Xiaojun Quan, and Liangzhi Li. Blockpruner: Fine-grained pruning for large language models. *arXiv*, 2024.