ORTHORANK: TOKEN SELECTION VIA SINK TOKEN ORTHOGONALITY FOR EFFICIENT LLM INFERENCE

Anonymous authors
Paper under double-blind review

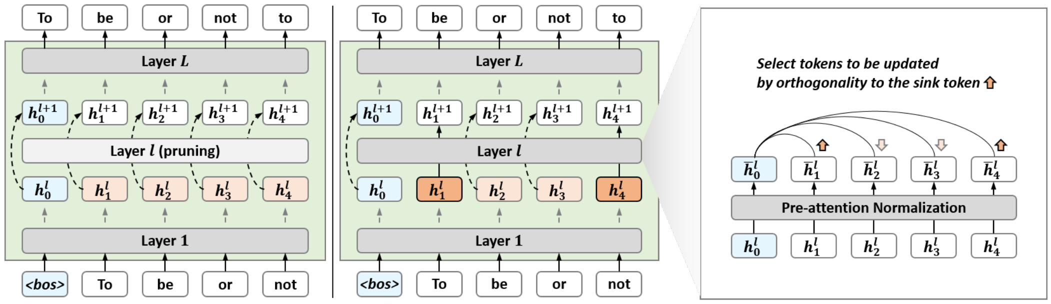
ABSTRACT

Attention mechanisms are central to the success of large language models (LLMs), enabling them to capture intricate token dependencies and implicitly assign importance to each token. Recent studies have revealed the sink token, which receives disproportionately high attention despite their limited semantic role. In this paper, we first expand the relationship between the sink token and other tokens, moving beyond attention to explore their similarity in hidden states, considering the layer depth. We observe that as the layers get deeper, the cosine similarity between the normalized hidden states of the sink token and those of other tokens increases, and that the normalized hidden states of the sink token exhibit negligible changes. These imply that other tokens consistently are directed toward the sink token throughout the layers. Next, we propose a dynamic token selection method, called `OrthoRank`, using these findings to select important tokens. Specifically, in a certain layer, we define token importance by the speed at which the token moves toward the sink token. This is converted into orthogonality with the sink token, meaning that tokens that are more orthogonal to the sink token are assigned greater importance. Finally, through extensive experiments, we demonstrated that our method results in lower perplexity and higher zero-shot accuracy compared to layer pruning methods at the same sparsity ratio with comparable throughput.

1 INTRODUCTION

Large language models (LLMs) have shown remarkable performance across various tasks (Thirunavukarasu et al., 2023; Wu et al., 2024; 2023; Labrak et al., 2024; Nam et al., 2024) However, despite this, the computational cost of LLM inference remains a significant challenge, especially for real-time applications. To address this challenge, many lightweight methods have been proposed for LLMs. Among the various methods, layer pruning is a simple and effective approach to reduce computational costs by removing layers that have less impact on the model. The impact is quantified by either measuring the similarity between the input and output at each layer (Siddiqui et al., 2024; Men et al., 2024), or by evaluating how the removal of a layer effects the final output (Song et al., 2024; Kim et al., 2024). Song et al. (2024) proposed an iterative pruning method based on these metrics, while Kim et al. (2024) introduced a one-shot pruning approach followed by additional tuning using LoRA (Hu et al., 2022). However, these methods have a limitation. They rely on selecting layers for pruning based on a calibration set, which statistically determines the layers that can be bypassed without computation. As a result, they do not effectively reflect the specific characteristics of the input tokens. For instance, at each layer, certain tokens may require computation while others do not, but layer pruning is unable to identify and process these tokens accordingly.

Motivated by the need for token level processing, early exit (Schuster et al., 2022; Chen et al., 2024) and mixture of depth (Raposo et al., 2024) have proposed dynamic computation paths based on token-level characteristics. Early exit determines that a token aligns with the final output, bypassing the remaining layers. Mixture of depth uses routers at each layer to decide whether a token should be computed or skipped. While these methods offer viable solutions, they rely on training additional routers or classifiers, or require the entire model to be trained specifically for early exit. Although these techniques have contributed to LLM acceleration, such as in speculative decoding, their practical use is limited because they require additional training across a wide range of existing models. This paper, therefore, begins by questioning:



(a) Layer pruning.

(b) `OrthoRank`.

Figure 1: Overview of our approach (OrthoRank). OrthoRank first determines the orthogonality of tokens to the sink token after normalization at each layer. Based on this, the top K tokens are selected for computation, while the remaining tokens bypass the layer.

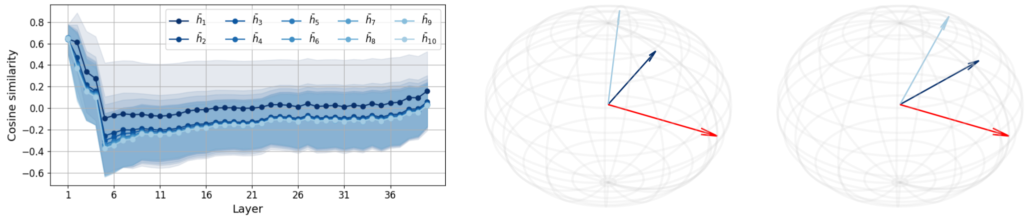
Can we identify which tokens advantageous to compute at each layer without extra training?

To explore this, we analyze the internal workings of LLMs to determine whether each token requires an update within a layer. Our focus is on one of the most distinctive phenomena in LLM behavior: the attention sink (Xiao et al., 2024), which was first studied by investigating attention distributions and identifying the presence of attention sinks. This phenomenon shows that the initial token in an input sequence receives a disproportionately large share of attention, despite often lacking meaningful semantic value. This occurs because, in autoregressive models, the initial token is visible to nearly all subsequent tokens, leading to 'excessive' attention scores. Since then, this phenomenon has been further explored (Sun et al., 2024; Cancedda, 2024), calibrated (Yu et al., 2024), and leveraged in various ways (Son et al., 2024; Zhang et al., 2024) to improve LLM efficiency and enhance understanding of their mechanisms. Through further investigation, we observe that sink tokens and other tokens exhibit a distinctive cosine similarity pattern (Section 2).

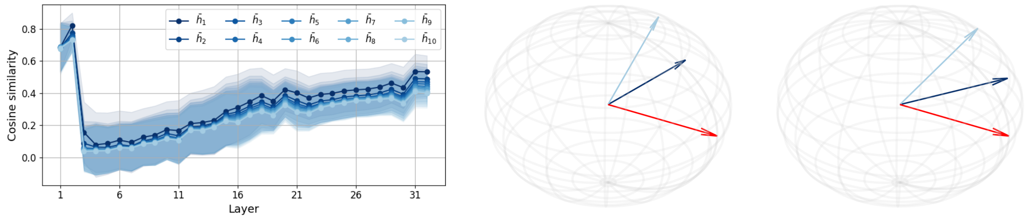
From this, we propose an importance ranking of tokens, `OrthoRank`, which leverages **Ortho**gonality to **Rank** tokens based on their relevance to the sink token. We confirm that selecting tokens with our orthogonal-based importance is effective, as it outperform the opposite method in language modeling performance (Section 3.1). To apply this across multiple layers in the LLM, we adopt the layer evaluation method from layer pruning. We then replace each layer with a token selection layer and evaluate them to identify the optimal token selection layers (Section 3.2). In Figure 1, we provide an overview of our proposed method, including the selection scheme. The main idea is to calculate each token's orthogonality to the sink token to select tokens. Selected tokens pass through all steps within the layer (e.g., query, key, value, feed forward network, etc.), while unselected tokens only participate in key and value calculations for the selected tokens without updating their own states, similar to early-exit mechanisms. Many studies (Sun et al., 2024; Son et al., 2024) suggest that in certain models, the attention sink phenomenon occurs not only with any token at the first position but also with specific delimiter tokens (e.g., ".", "\n"). However, for simplicity and consistency, we focus our calculations on the first token. That is, h_i^l represents the input hidden states of the sink token ($i = 0$) and other tokens ($i \geq 1$) at layer l.

In summary, the key contributions of our paper are as follows:

- We discover that after the layer where the attention sink occurs, the cosine similarity between the normalized hidden states of the sink token and those of other tokens increases, as the layers deepen. However, the normalized hidden states of the sink token across the layers remains largely unchanged. These mean that other tokens are heading toward the sink token.
- We propose a simple but effective token selection method `OrthoRank` based on token-sink orthogonality, prioritizing tokens closer to orthogonality for computation and bypassing others, without the need for additional modules or training.
- We conduct extensive evaluations demonstrating that our method has better performance compared to the existing layer pruning at the same sparsity with comparable throughput.



(a) Llama-2-13B. (b) Layer 5 of (a). (c) Layer 40 of (a).



(d) Mistral-7B. (e) Layer 3 of (d). (f) Layer 32 of (d).

Figure 2: (a, d) Cosine similarity between the normalized hidden states of the sink token ($\bar{h}_0$) and other tokens of Llama-2-13B and Mistral-7B. l_{sink} is layer 4 and layer 2, respectively. (b-c, e-f) Conceptual representation of the relationship between the sink token (red line) and other tokens (blue lines) at layer right after l_{sink} and the final layer. After the attention sink, as layers progress, the cosine similarity between the sink token and the other tokens increases, indicating that the tokens are gradually aligning more closely with the sink token.

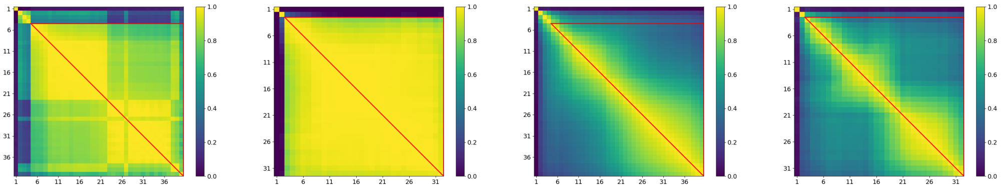
2 FURTHER ANALYSIS ON ATTENTION SINK BEYOND ATTENTION

In this section, we revisit the concept of the attention sink and introduce new insights based on further analysis. An attention sink refers to the phenomenon where a particular token receives a disproportionately high amount of attention from other tokens. This phenomenon is always observed after a certain early layer, l_{sink}, in the initial token (Xiao et al., 2024; Sun et al., 2024).

We begin by verifying whether the special relationship between the sink token and the other tokens appears in states other than attention. To the best of our knowledge, we are the first to analyze the behavior of the sink token and other tokens through similarity analysis. In fact, the attention map provides limited information regarding the layers because there is little difference between the layers after layer l_{sink}. Therefore, we focus on the hidden states after the pre-attention normalization layer (i.e., the normalized hidden states), because they are the direct inputs for an attention module in each layer.

We investigate the cosine similarity between the normalized hidden states of tokens throughout the layers, using randomly sampled 500 sentences. Figures 2(a) and 2(d) describe the cosine similarity between the normalized hidden states of the sink token (i.e., $\bar{h}_0$) and those of other tokens (i.e., $\bar{h}_i$ $(1 \leq i \leq 10)$) of Llama-2-13B and Mistral-7B, respectively. For each model, the attention sink occurs at layer 4 and layer 2. It is observed that the cosine similarity between the sink token and other tokens decreases drastically right after layer l_{sink}. However, after layer l_{sink}, the cosine similarity between the sink token and other tokens tends to increase as the layers progress, although the ranges of cosine similarity vary across models.

These findings are simply illustrated in 3D by Figures 2(b-c) and 2(e-f). We plot each state as a unit vector on the hypersphere to focus on angles at layer right after l_{sink} and the final layer. In the subfigures, the red line represents the normalized hidden states of the sink token (i.e., $\bar{h}_0$), while the blue lines represent those of other tokens (i.e., $\bar{h}_1$ and $\bar{h}_{10}$). Additionally, the cosine similarity between other tokens except for the sink token does not show any consistent trend, presented in Appendix B. In summary, our findings offer insights that go beyond the information derived from the attention map, revealing that as the layers deepen, the angles between the sink token and the other tokens gradually decrease, after layer l_{sink}.



(a) Llama-2-13B ($\bar{h}_0$). (b) Mistral-7B ($\bar{h}_0$). (c) Llama-2-13B ($\bar{h}_i$). (d) Mistral-7B ($\bar{h}_i$).

Figure 3: (a-b) Cosine similarity between the normalized hidden states of the sink token across layers. (c-d) Cosine similarity between the normalized hidden states of another token across layers. The red boundary represents the layers after layer l_{sink}. The sink token shows similar values not only with adjacent layers but also with distant layers, as confirmed through (a) and (b). In contrast, other tokens show similarity in adjacent layers, but differences accumulate, leading to dissimilarity in distant layers, as shown in (c) and (d). These results highlight the static nature unique to the sink token, in contrast to other tokens.

Obs. (1). For layer l_{sink} and the final layer L, $cos(\bar{h}_0^{l_{sink}+1}, \bar{h}_i^{l_{sink}+1}) \leq cos(\bar{h}_0^L, \bar{h}_i^L)$, $\forall i$. Moreover, when $l_{sink} < l_1 < l_2 \leq L$, it generally holds $cos(\bar{h}_0^{l_1}, \bar{h}_i^{l_1}) \leq cos(\bar{h}_0^{l_2}, \bar{h}_i^{l_2})$, $\forall i$.

Next, we explore the cosine similarity between the normalized hidden states of the same token across different layers. Through this analysis, we can determine whether the sink token and other tokens are converging towards each other, or if one remains relatively stationary while the other actively moves towards it.

Figures 3(a) and 3(b) illustrate the cosine similarity between the normalized hidden states of the sink token across all layers of Llama-2-13B and Mistral-7B, respectively. The red boundary highlights the layers ranging from l_{sink} to the final layer L. For Llama-2-13B, after passing through layer l_{sink}, the layers are grouped together, with each group exhibiting a significantly higher degree of similarity (close to 1). Despite this grouping, layers across different groups still maintain a relatively high level of similarity, approaching nearly 0.8. For Mistral-7B, all layers following layer l_{sink} form a single cohesive group, where the similarity between these different layers is consistently close to 1. These results suggest that the sink token experiences almost no change in its trajectory in the normalized hidden states space as it moves through the deeper layers. Therefore, the *fixed* sink token on the hypersphere in Figure 2, which simplifies our Observation (1), is nearly accurate. Furthermore, this observation can be linked to massive activations of the sink token in the hidden states, which appear in a small number of fixed feature dimensions and are delivered to the next layer via the residual connection, keeping high cosine similarity across layers.

Figures 3(c) and 3(d)illustrate the cosine similarity between the normalized hidden states of a token, excluding the sink token, across all layers of Llama-2-13B and Mistral-7B, respectively. As expected, due to the presence of residual skip connections, there is relatively high similarity between adjacent layers, especially along the diagonal. However, as the model processes more layers, differences between the layers begin to accumulate, and the normalized hidden states at the final layer eventually exhibit low cosine similarity compared to the normalized hidden states immediately following layer l_{sink}.

Obs. (2). When $l_{sink} < l_1 < l_2 \leq L$, $cos(\bar{h}_0^{l_1}, \bar{h}_0^{l_2})$ remains close to 1. However, $cos(\bar{h}_i^{l_1}, \bar{h}_i^{l_2})$ decreases as the gap between l_1 and l_2 widens, $\forall i \geq 1$.

From Obs. (1) and Obs. (2), it is concluded that:

As the layer deepens, other tokens gradually align with the sink token, which remains almost static.

3 ORTHORANK: DYNAMIC TOKEN SELECTION

In this section, we extend our observations as criteria for selecting tokens at layer l (Section 3.1). Then, we propose a dynamic token selection algorithm, called `OrthoRank` (Section 3). Our algorithm can be used in conjunction with the layer selection algorithm.

3.1 DYNAMIC TOKEN SELECTION CRITERIA

Attention scores are widely used to identify relationships between tokens and are often employed to determine token importance. However, we discover that the relationships between tokens can also be captured through normalized hidden states. Based on this observation, we propose using these states to define token importance.

Our findings suggest that tokens follow a discrete trajectory in which they align with the sink token (i.e., they move in a direction that increases cosine similarity). Building on this, we define the importance of token i in a certain layer after l_{sink} as the extent to which token i can influence its cosine similarity with the sink token[1]:

$$\left\| \frac{\partial}{\partial \bar{h}_i} \cos\left(\bar{h}_0, \bar{h}_i\right) \right\|. \tag{1}$$

Starting from the relation $\bar{h}_0^\top \bar{h}_i = \|\bar{h}_0\| \|\bar{h}_i\| \cos\left(\bar{h}_0, \bar{h}_i\right)$, we compute the gradient of $\cos\left(\bar{h}_0, \bar{h}_i\right)$ with respect to $\bar{h}_i$:

$$\frac{\partial}{\partial \bar{h}_i} \cos\left(\bar{h}_0, \bar{h}_i\right) = \frac{1}{\|\bar{h}_i\|} \left(\frac{\bar{h}_0}{\|\bar{h}_0\|} - \cos\left(\bar{h}_0, \bar{h}_i\right) \frac{\bar{h}_i}{\|\bar{h}_i\|} \right). \tag{2}$$

Assuming that normalized hidden states have approximately equal norms except for sink token, we can simplify the importance of token i based on the cosine similarity:

$$\left\| \frac{\partial}{\partial \bar{h}_i} \cos\left(\bar{h}_0, \bar{h}_i\right) \right\|^2 \approx 1 - \cos^2\left(\bar{h}_0, \bar{h}_i\right). \tag{3}$$

Thus, the importance of token i is directly related to how small $|\cos\left(\bar{h}_0, \bar{h}_i\right)|$ is. As $|\cos\left(\bar{h}_0, \bar{h}_i\right)|$ decreases, the importance increases because tokens that are **more orthogonal** to the sink token are more likely to be selected, as they have a greater potential to influence the overall cosine similarity.

For implementation convenience, since the norms are approximately equal, we use the absolute value of the inner product $|\bar{h}_0^\top \bar{h}_i|$ as a practical proxy for $|\cos\left(\bar{h}_0, \bar{h}_i\right)|$. Therefore, to select the top k important tokens, we rank them based on the smallest $|\bar{h}_0^\top \bar{h}_i|$, which corresponds to **selecting the tokens that are more orthogonal to the sink token**.

$$\text{Select top } k \text{ tokens with smallest } |\bar{h}_0^\top \bar{h}_i|. \tag{4}$$

To validate the effectiveness of our proposed selection criterion, we performed an experiment using the WikiText-2 dataset (Merity et al., 2022). In this experiment, we applied token selection one layer at a time, examining the impact on the model's language modeling performance. For each individual layer, we selected the top 33% of tokens for computation based on our proposed metric, which prioritizes tokens that are more orthogonal to the sink token (i.e., those with the smallest inner product). We then compared the resulting perplexity (ppl) scores to those obtained using an alternative method, where instead of selecting the most orthogonal tokens, we selected the bottom 33%—the tokens with the largest inner product—thereby evaluating the inverse of our approach.

---

[1] For simplicity, l is omitted in this section.

Figure 4, drawn for layers after the attention sink (layer $>$ 4), shows the ppl difference between the two approaches, where a negative value indicates better performance (lower ppl) using our selection criterion. The results demonstrate that, except for the final layer, selecting orthogonal tokens consistently outperforms the opposite approach.

In summary, our orthogonal token selection criterion leads to better performance across most layers, confirming its effectiveness in reducing computation while maintaining accuracy.

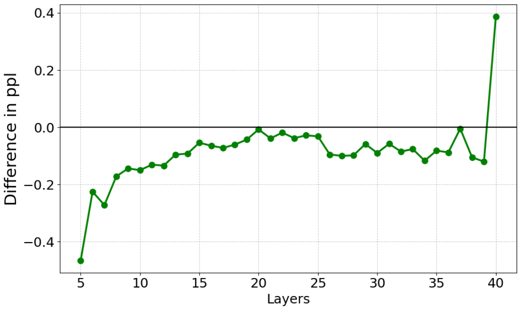
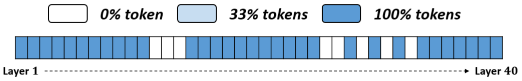


Figure 4: Layer-wise token selection.

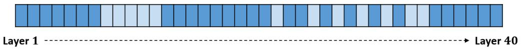
3.2 DYNAMIC TOKEN SELECTION WITH SELECTIVE LAYER

In Section 3.1, we demonstrated that selecting tokens closer to orthogonality at each layer improves effectiveness while preserving model performance. However, challenges arise when applying this selection across all layers. First, our selection criteria are less valid before the attention sink occurs. Second, layers near the output are crucial for maintaining model reliability and require computation for most tokens. Additionally, inter-layer dependencies must be considered.

Therefore, instead of applying our selection criteria (`OrthoRank`) across all layers, we propose selectively applying it to specific layers. To implement this, we combine our selection criteria with existing layer pruning methods. While traditional layer pruning approaches measure performance by removing layers one by one, we measure performance by applying token selection to layers incrementally. This strategy enables efficient computation across both tokens and layers while preserving model fidelity.



(a) Layer pruning.



(b) `OrthoRank` with selective layer.

Figure 5: Comparison under the same sparsity.

Figure 5 compares layer pruning (Song et al., 2024) and `OrthoRank` with selective layers. In Figure 5(a), Layer pruning is applied to a Llama2-13b model with 40 layers and 20% sparsity, showing the pruned layers. In Figure 5(b), to maintain the same sparsity, 30% of the layers are modified to compute only 33% of the tokens, where the top 33% most orthogonal tokens to the sink token are selected for computation.

Algorithm 1: OrthoRank Layer in PyTorch

```
def select_token_in_orthorank_layer(block, pruning_ratio):
    importance_score = abs(torch.matmul(hidden_states[:, [0], :],
        hidden_states.transpose(1, 2)).squeeze(1))

    # exception handing for sink token
    importance_score[:,0] = float('inf')

    lowk_indices = importance_score.topk(k=int(pruning_ratio *
        normalized_hidden_state.size(1), largest=False), dim=-1).indices

    #Sorting for attention module
    lowk_indices_sorted = torch.sort(lowk_indices).values

    # using except key & value states
    selected_hidden_states = torch.gather(hidden_states, 1,
        lowk_indices_sorted.unsqueeze(-1).expand(-1, -1, hidden_states.
        size(-1)))
```

Table 1: Perplexity results on C4 dataset for various models.

Method	Sparsity	Throughtput	Llama2			Llama3	Mistral	Mixtral
			7B	13B	70B	8B	7B	8x7B
Dense	0%	1.00x	7.26	6.73	5.71	9.45	8.38	7.41
SLEB	10%	1.11x	8.71	7.80	6.32	12.47	9.74	8.28
+OrthoRank	10%	1.10x	**8.06**	**7.39**	**6.13**	**11.27**	**9.31**	**8.05**
SLEB	20%	1.25x	10.90	9.42	7.31	16.49	12.38	**9.46**
+OrthoRank	20%	1.23x	**10.04**	**8.74**	**7.21**	**14.95**	**11.54**	9.56
Shortened LLaMa	10%	1.11x	8.79	7.93	6.34	13.28	9.99	**8.37**
+OrthoRank	10%	1.10x	**8.04**	**7.60**	**6.29**	**11.22**	**9.43**	8.47

Table 2: Mean accuracies (%) on zero-shot tasks for various models.

Method	Sparsity	Llama2			Llama3	Mistral	Mixtral
		7B	13B	70B	8B	7B	8x7B
Dense	0%	63.13	66.74	73.13	72.87	74.14	74.41
SLEB	10%	63.13	66.74	73.13	66.94	68.82	**77.23**
+OrthoRank	10%	**65.06**	**69.71**	**74.56**	**69.55**	**69.78**	75.55
SLEB	20%	58.68	62.97	70.82	58.10	61.59	70.84
+OrthoRank	20%	**60.35**	**66.99**	**71.25**	**60.84**	**63.38**	**72.52**
Shortened LLaMa	10%	62.07	69.72	**74.22**	69.87	66.63	71.97
+OrthoRank	10%	**64.79**	**70.78**	73.67	**70.77**	**68.33**	**73.70**

4 EXPERIMENTS

4.1 IMPLEMENTATION DETAILS

We conducted experiments comparing layer pruning and OrthoRank with selective layer approaches. Following the evaluation protocol in (Song et al., 2024), we set target sparsities at 10% and 20%. To ensure the same sparsity ratio across methods, our algorithm applied 15% and 30 % layer selection, with only 33% of tokens computed in the selected layers. We compared our method against two baseline algorithms: the iterative layer pruning method SLEB (Song et al., 2024) and the one-shot pruning method Shortened LLaMA (Kim et al., 2024) without finetuning. Since the one-shot method suffers a significant performance drop at 20% without fine-tuning, we limited its comparison to 10% sparsity. We measured throughput on a single A6000 GPU with a batch size of 32 and prompts of length 2048. To validate the robustness of ours across a wide range of models, we conducted experiments on various models, including Llama2 (7B, 13B, 70B) (Touvron et al., 2023), Llama3 (8B) (Dubey et al., 2024), Mistral (7B) (Jiang et al., 2023), Mixtral (8×7B) (Jiang et al., 2024).

4.2 RESULTS ON PERPLEXITY

Table 1 compares the performance of various models on the language modeling task. Since the layers were pruned (selected) using the Wikitext-2 dataset, we used the the C4 validation set (Raffel et al., 2020) for the performance comparison. Our proposed method, OrthoRank, outperformed other layer pruning approaches in terms of perplexity in most cases except for some case of Mixtral-8x7B model. This indicates that by focusing on token orthogonality to the sink token, OrthoRank efficiently reduces computational complexity, outperforming layer pruning at the same sparsity ratio with nearly equivalent throughput.

4.3 RESULTS ON ZERO-SHOT TASK

We further evaluated OrthoRank's performance on several zero-shot tasks, including PIQA (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2021), HellaSwag (Zellers et al., 2019), ARC-easy, and ARC-challenge (Clark et al., 2018), using the LM Evaluation Harness. As shown in Table 2, OrthoRank demonstrated better performance compared to layer pruning in most cases, except for Mixtral 8x7B with 10% SLEB, and Llama2-70B.

Table 3: Ablation study for selection criteria, stage, and KV for unselected token.

Criteria	Stage	KV	Llama2-13B		Llama3-8B		Mistral-7B		Mixtral-8x7B	
			ppl$_{\downarrow}$	acc$_{\uparrow}$	ppl$_{\downarrow}$	acc$_{\uparrow}$	ppl$_{\downarrow}$	acc$_{\uparrow}$	ppl$_{\downarrow}$	acc$_{\uparrow}$
Random	$\bar{\boldsymbol{h}}_{\boldsymbol{i}}$	✓	10.83	61.17	16.23	59.15	12.05	62.88	9.85	69.45
Orthogonal ↓	$\bar{\boldsymbol{h}}_{\boldsymbol{i}}$	✓	11.85	58.81	18.06	58.37	13.03	59.47	9.76	66.27
Orthogonal ↑	h_i	✓	9.64	64.30	15.72	60.82	**11.53**	63.87	**9.55**	71.49
Orthogonal ↑	$\bar{\boldsymbol{h}}_{\boldsymbol{i}}$	✗	9.77	64.70	17.72	58.76	14.21	61.34	10.94	67.55
Orthogonal ↑	$\bar{\boldsymbol{h}}_{\boldsymbol{i}}$	✓	**8.74**	**66.99**	**14.95**	**60.84**	11.54	**63.88**	9.56	**72.85**

Table 4: Ablation study according to the layer and token selection ratios under the sparsity of 20%.

Layer ratio	**Token ratio**	**Acc. (%)**
0.2	0	62.97
0.3	0.333	66.99
0.4	0.5	65.14

4.3.1 Token selection Criteria

Row 1, 2, and 5 in Table 3 compare the performance of different token selection strategies: random selection, Orthogonal ↓, and Orthogonal ↑ (Ours). Ours consistently achieves the best results, with lower perplexity and higher accuracy across all models, followed by random selection, while the opposite strategy (Orthogonal ↓) performs the worst. These results confirm that selecting tokens further from the sink token (Orthogonal ↑) leads to more efficient selection and improved performance for same computation complexity, validating the token selection criterion introduced in Section 3.

4.3.2 Similarity Measurement Stage

Table 3 compares the use of hidden states (Row 3) and normalized hidden states (Row 5) for token selection. The results show that using normalized hidden states leads to better performance, with lower perplexity and higher accuracy. As discussed in Section 2, our findings are based on normalized hidden states, making this result consistent with our expectations and further confirming the importance of normalization in improving token selection. However, it performs better than some other components of our approach, suggesting that while the weights within the normalization process do affect the cosine similarity, the hidden state similarity still operates in a somewhat similar manner.

4.3.3 KV calculation for unselected tokens

In Table 3, we compare Row 4, where Key and Value (KV) computations for unselected tokens are skipped, with Row 5, where KV values are computed even for unselected tokens. The results show that calculating KV values for all tokens, regardless of whether they are selected for updates, leads to better performance. This is because our token selection strategy focuses on how quickly a token's state updates, without considering the influence these tokens exert on others through KV interactions. When KV calculations for unselected tokens are skipped, the reduced interaction among tokens significantly degrades overall performance. Therefore, while unselected tokens are not updated, it is essential to compute their KV values to maintain model performance. This approach resembles calculating key-value pairs for tokens that have exited in early exit methods.

4.4 Sparsity Trade-offs in Token and Layer Selction

We conducted experiments by varying the ratio of selected layers and selected tokens while maintaining the same effective sparsity. Based on Table 4, we recommend using a token selection ratio between 0 and 0.5.

5 RELATED WORK

Layer Pruning. Layer pruning has been a prominent approach for reducing the computational complexity of large language models (LLMs), particularly in transformer architectures (Siddiqui et al., 2024; Men et al., 2024). Approaches like SLEB (Song et al., 2024) and Shortened LLaMA (Kim et al., 2024) aim to remove entire layers that are deemed less critical for downstream tasks. These methods often rely on performance metrics or sensitivity analysis to determine which layers contribute less to overall model accuracy and can be pruned without significant loss of performance. However, layer pruning may result in abrupt performance degradation, particularly when layers responsible for essential token transformations are removed. While these methods effectively reduce model depth, they do not account for token-level variations within layers. In contrast, our approach integrates token selection within specific layers, maintaining layer depth but reducing the number of tokens processed in each layer, thus offering a more fine-grained control over computational efficiency.

Attention Sink. The concept of the attention sink, where certain tokens receive disproportionately high attention across layers, has gained attention in recent studies. Xiao et al. (2024) first introduced the term “attention sink” to describe how the initial token in a sequence tends to dominate attention scores in autoregressive models. This is attributed to its visibility to all subsequent tokens, causing it to act as a “sink” for attention. Sun et al. (2024) further investigated the attention sink phenomenon, showing that this behavior persists across multiple layers, leading to a static role for the sink token, while other tokens move toward it in hidden state space. Building on these observations, our work explores token-sink orthogonality and uses this metric to inform token selection. By selecting tokens that are more orthogonal to the sink token, we prioritize tokens with greater potential to contribute to meaningful computations, leveraging the inherent token dynamics to optimize inference efficiency.

Token Pruning. Token pruning methods have been widely explored as a way to reduce the number of tokens processed across layers, thus decreasing computational load. Techniques like dynamic token selection (Lou et al., 2024) and early exit mechanisms (Chen et al., 2024; Del Corro et al., 2023; Elhoushi et al., 2024) progressively drop tokens deemed uninformative as they pass through layers. These methods rely on criteria such as attention scores or token contribution measures to decide which tokens to prune. However, one potential downside of token pruning is the loss of potentially relevant information as tokens are eliminated layer by layer, especially in deeper models where remaining tokens may not fully capture the complexity of the input sequence. Our approach differs significantly in that we do not progressively drop tokens across layers. Instead, we selectively compute a subset of tokens at specific layers based on their orthogonality to the sink token. This ensures that we preserve the flexibility to compute tokens based on their relevance without completely discarding them, thus mitigating the risk of information loss while still reducing computational costs.

6 CONCLUSION

In this paper, we introduced OrthoRank, a dynamic token selection strategy based on the orthogonality between tokens and the sink token. Our approach was motivated by the observation that as layers deepen, tokens increasingly align with the sink token in the normalized hidden state space. By analyzing token-sink similarity, we found that tokens more orthogonal to the sink token play a greater role in computation. Leveraging this insight, we developed a token selection mechanism that prioritizes such tokens at specific layers, leading to more efficient computation. By applying this token selection approach to selective layers, we achieved superior performance compared to traditional layer pruning methods at the same sparsity level with comparable throughput. Extensive experiments demonstrated significant improvements, and ablation studies confirmed that our selection scheme is optimized both theoretically and empirically. Furthermore, our findings on token-sink similarity offer valuable insights for future research in Efficient LLM inference and Interpretable LLMs, providing a foundation for further optimizing and understanding large language models.

REPRODUCIBILITY

We provide pseudo code through Algorithm 1 and implementation details in Section 4.1.

REFERENCES

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Nicola Cancedda. Spectral filters, dark signals, and attention sinks. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4792–4808, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.263.

Yanxi Chen, Xuchen Pan, Yaliang Li, Bolin Ding, and Jingren Zhou. Ee-llm: Large-scale training and inference of early-exit large language models with 3d parallelism. In *Forty-first International Conference on Machine Learning*, 2024.

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 preprint arXiv:1803.05457*, 2018.

Luciano Del Corro, Allie Del Giorno, Sahaj Agarwal, Bin Yu, Ahmed Awadallah, and Subhabrata Mukherjee. Skipdecode: Autoregressive skip decoding with batching and caching for efficient llm inference. *arXiv preprint arXiv:2307.02628*, 2023.

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 preprint arXiv:2407.21783*, 2024.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layer skip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*, 2024.

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 *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=nZeVKeeFYf9`.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. Shortened llama: A simple depth pruning for large language models. *arXiv preprint arXiv:2402.02834*, 2024.

Yanis Labrak, Adrien Bazoge, Emmanuel Morin, Pierre-Antoine Gourraud, Mickael Rouvier, and Richard Dufour. BioMistral: A collection of open-source pretrained large language models for medical domains. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics ACL 2024*, pp. 5848–5864, Bangkok, Thailand and virtual meeting, August 2024. Association for Computational Linguistics.

Chao Lou, Zixia Jia, Zilong Zheng, and Kewei Tu. Sparser is faster and less is more: Efficient sparse attention for long-range transformers. *arXiv preprint arXiv:2406.16747*, 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 preprint arXiv:2403.03853*, 2024.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2022.

Daye Nam, Andrew Macvean, Vincent Hellendoorn, Bogdan Vasilescu, and Brad Myers. Using an llm to help with code understanding. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pp. 1–13, 2024.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

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 preprint arXiv:2404.02258*, 2024.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems*, 2022.

Shoaib Ahmed Siddiqui, Xin Dong, Greg Heinrich, Thomas Breuel, Jan Kautz, David Krueger, and Pavlo Molchanov. A deeper look at depth pruning of llms. *arXiv preprint arXiv:2407.16286*, 2024.

Seungwoo Son, Wonpyo Park, Woohyun Han, Kyuyeun Kim, and Jaeho Lee. Prefixing attention sinks can mitigate activation outliers for large language model quantization. *arXiv preprint arXiv:2406.12016*, 2024.

Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, et al. Sleb: Streamlining llms through redundancy verification and elimination of transformer blocks. In *Forty-first International Conference on Machine Learning*, 2024.

Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024.

Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. Large language models in medicine. *Nature medicine*, 29(8):1930–1940, 2023.

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

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*, 2023.

Yiran Wu, Feiran Jia, Shaokun Zhang, Hangyu Li, Erkang Zhu, Yue Wang, Yin Tat Lee, Richard Peng, Qingyun Wu, and Chi Wang. Mathchat: Converse to tackle challenging math problems with llm agents. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024.

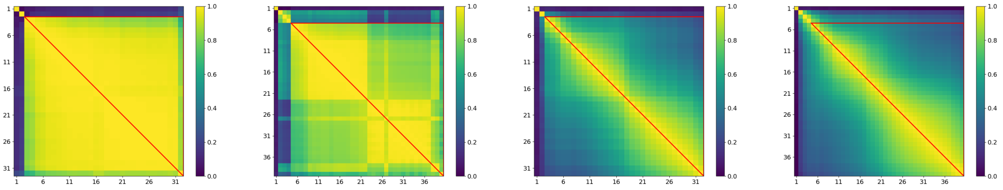
Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024.

Zhongzhi Yu, Zheng Wang, Yonggan Fu, Huihong Shi, Khalid Shaikh, and Yingyan Celine Lin. Unveiling and harnessing hidden attention sinks: Enhancing large language models without training through attention calibration. *arXiv preprint arXiv:2406.15765*, 2024.

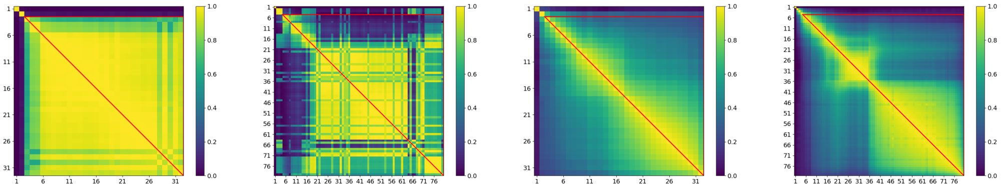
Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019.

Yichi Zhang, Bofei Gao, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, Wen Xiao, et al. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

A COSINE SIMILARITY CHANGES OF THE SAME TOKEN ACROSS LAYERS



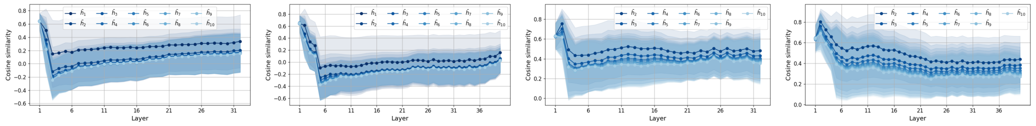
(a) Llama-2-7B ($\bar{h}_0$). (b) Llama-2-13B ($\bar{h}_0$). (c) Llama-2-7B ($\bar{h}_i$). (d) Llama-2-13B ($\bar{h}_i$).



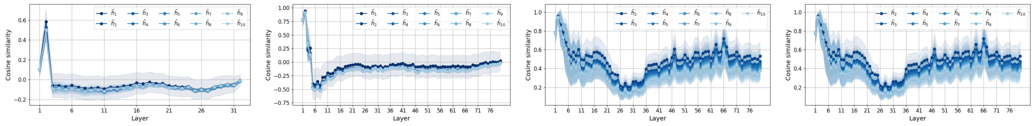
(e) Llama-3-8B ($\bar{h}_0$). (f) Llama-3-70B ($\bar{h}_0$). (g) Llama-3-8B ($\bar{h}_i$). (h) Llama-3-70B ($\bar{h}_i$).

Figure 6: (a-b,e-f) Cosine similarity between the normalized hidden states of the sink token across layers. (c-d, g-h) Cosine similarity between the normalized hidden states of another token across layers. The red boundary represents the layers after layer l_{sink}.

B CHANGES IN COSINE SIMILARITY BETWEEN TOKENS ACROSS LAYERS



(a) Llama-2-7B. (b) Llama-2-13B. (c) Llama-2-7B. (d) Llama-2-13B.



(e) Llama-3-8B. (f) Llama-3-70B. (g) Llama-3-8B. (h) Llama-3-70B.

Figure 7: (a-b,e-f) cosine similarity between the normalized hidden states of the sink token ($\bar{h}_0$) and other tokens. (c-d, g-h) cosine similarity between the normalized hidden states of the token at position i (where $i \neq 0$) and other tokens.