# WAVEPRUNE: ONE PERIOD IS OFTEN ENOUGH FOR ROPE

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

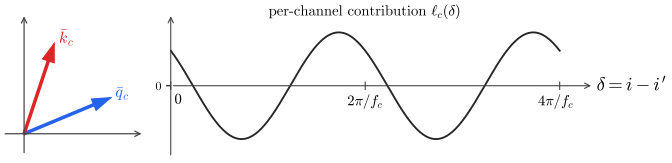
Rotary Position Embedding (RoPE) encodes token positions by rotating each two-dimensional channel of the query and key vectors at a channel-specific frequency, making the attention logits invariant to a common shift of positions. However, this rotation is periodic, and it leads to position aliasing where relative positions separated by a full rotation period become hard to tell apart. To address this, we propose *WavePrune*, which restricts each channel to its *first rotation period*. We show that it removes the distractions in attention maps created by position aliasing and improves overall long-context performance. Specifically, WavePrune raises the HELMET score on four of five models we test without any extra tuning (e.g., 35.7 $\rightarrow$ 40.0 on Qwen3-8B). When pretraining models from scratch, WavePrune also achieves lower validation loss at extrapolated lengths than pretraining without it. Because WavePrune restricts each channel to a sliding window, it induces a fine-grained sparsity that our hardware-aligned CUDA kernels exploit for 1.16× prefill and 1.24× decoding speedups over FlashAttention-2 at 32K context. Together, these results show that RoPE's periodic structure, widely regarded as essential, is largely redundant beyond the first rotation period.

## 1 INTRODUCTION

Rotary Position Embedding (RoPE) (Su et al., 2024) is commonly used in large language models (LLMs) to encode token positions by applying position-dependent rotations to the query and key vectors in attention layers. A key advantage of RoPE is that it makes the attention logit invariant under a common shift of the token positions, a property usually referred to as *translation invariance*.

More formally, let $q, k \in \mathbb{R}^{d_h}$ be the query and key vectors at positions $i$ and $i'$, respectively. To compute the attention logit $L_{i,i'}$ between positions $i$ and $i'$, RoPE first breaks $q, k$ into 2D channels $\bar{q}_c := (q_{2c+1}, q_{2c+2})$, $\bar{k}_c := (k_{2c+1}, k_{2c+2})$, where $c = 0, \ldots, d_h/2 - 1$, and then sums up the inner products between corresponding channels after rotating them by position-dependent angles:

$$L_{i,i'} = \sum_{c=0}^{d_h/2-1} \ell_c(i - i'), \qquad \ell_c(\delta) := \bar{k}_c^\top \begin{pmatrix} \cos(\delta f_c) & -\sin(\delta f_c) \\ \sin(\delta f_c) & \cos(\delta f_c) \end{pmatrix} \bar{q}_c. \tag{1}$$



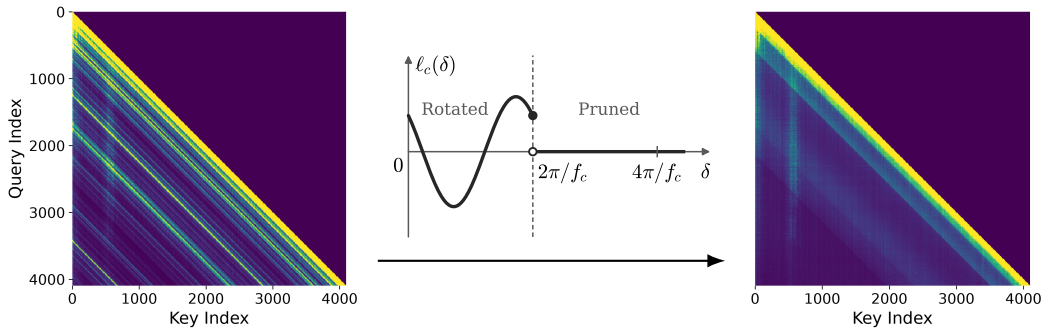

Here $f_c$ is a fixed value that is called the *rotation frequency* of channel $c$. One can see that each per-channel contribution in RoPE is a sinusoidal function of the relative displacement $\delta := i - i'$ with period $2\pi/f_c$ (for fixed $\bar{q}_c$ and $\bar{k}_c$), which is independent of the absolute positions $i$ and $i'$. Consequently, the full attention logit $L_{i,i'}$ remains unchanged when both positions are shifted by the same amount, thus leading to translation invariance.

While this translation invariance is natural for language modeling, RoPE is not the only way to achieve it. In fact, attention logits are translation invariant as long as each per-channel contribution is

*Equal contribution



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.




(a) Original

(b) WavePrune

Figure 1: In WavePrune, we restrict each channel to its first RoPE period, which removes several "ghost" sub-diagonals in the Qwen3-8B attention map. See Appendix H.1 for details.

a function of the relative displacement $\delta := i - i'$. If so, *why does the per-channel contribution in RoPE have to be periodic?*

There are two opposing views on the role of periodicity in RoPE. On the one hand, these periodic components provide a Fourier-like representation of relative positions, which LLMs can exploit to construct specific positional attention patterns (Barbero et al., 2025; Hua et al., 2025; Wang et al., 2026a). On the other hand, periodicity also introduces a fundamental ambiguity in positional encoding, as different relative positions can acquire similar rotary phases and consequently similar attention patterns (Du et al., 2026). This ambiguity can become especially problematic in long-context settings, where the model needs to distinguish relative positions over much longer distances. We refer to this phenomenon as *position aliasing*, following the terminology in Du et al. (2026).

In this paper, we support the second view by showing that repeated rotations beyond the first period are in fact unnecessary: **one period is often enough for RoPE**.

**Our Contributions.** We propose to restrict each RoPE channel to its first rotation period, a method we call **WavePrune**. Across several model families including Qwen, Gemma and Ministral, WavePrune does not harm long-context performance and often improves it (see Table 1). To understand why this is the case, we visualize the attention maps of the original and WavePrune models (Figure 1), and find that WavePrune eliminates many slash patterns that lie along sub-diagonals in the attention maps, which we refer to as *"ghost" sub-diagonals*. While slash patterns have often been interpreted as useful attention structures for long-context modeling (Yang et al., 2026; Cheng et al., 2026), we demonstrate that some of them are instead a byproduct of position aliasing in RoPE, and that WavePrune may succeed by removing these distractions. Our key findings are summarized as follows:

- We propose WavePrune, a training-free method that restricts each RoPE channel to one rotation period, and show that it removes "ghost" sub-diagonals and improves long-context performance, raising the HELMET score on four of five models (e.g., $35.7 \rightarrow 40.0$ on Qwen3-8B).
- To access the full potential of WavePrune, we also pretrain models with WavePrune from scratch, which consistently improves length extrapolation over pretraining without it. Controlled experiments on synthetic tasks suggest that WavePrune's success may be related to resolving two distinct failure modes of RoPE in handling short-range and long-range dependencies.
- We develop an efficient implementation of WavePrune, called **FlashWavePrune**, which reduces the computational cost and memory traffic of attention by leveraging the sparse attention pattern of WavePrune. It achieves $1.16\times$ prefill and $1.24\times$ decoding speedups over FlashAttention-2 at 32K context.

## 2 Preliminaries

We first review the standard Transformer architecture (Vaswani et al., 2017) and Rotary Position Embedding (RoPE) (Su et al., 2024), and introduce the notations used throughout the paper.

**Projecting tokens into QKV.** Consider a Transformer (Vaswani et al., 2017) operating on a sequence of $n$ tokens, represented by an input matrix $X \in \mathbb{R}^{n\times d}$. For clarity, we describe a single attention head of dimension $d_h$; a multi-head attention layer simply runs multiple such heads in parallel and concatenates their outputs. The head first maps $X$ through three learned projection matrices $W_Q, W_K, W_V \in \mathbb{R}^{d\times d_h}$ to obtain the query, key, and value matrices:

$$Q \coloneqq XW_Q, \qquad K \coloneqq XW_K, \qquad V \coloneqq XW_V \ \in \ \mathbb{R}^{n\times d_h}. \tag{2}$$

We write $q_i, k_i, v_i \in \mathbb{R}^{d_h}$ for the $i$-th rows of $Q, K, V$ (viewed as column vectors), and call them the *query*, *key*, and *value* of token $i$, respectively.

**Scaled Dot-Product Attention (SDPA).** To determine how much every token attends to another, each query is compared against keys via scaled dot products. Under causal (autoregressive) masking, a token attends only to itself and to preceding tokens, so for $j \le i$ the *attention logit* between query $i$ and key $j$ is defined as

$$L_{i,j} \coloneqq \frac{1}{\sqrt{d_h}}\, q_i^\top k_j, \tag{3}$$

and for $j > i$ the logit is defined as $L_{i,j} \coloneqq -\infty$. Applying a row-wise softmax to the *attention logit matrix* $L \in \mathbb{R}^{n\times n}$ yields the *attention score matrix* $A_{i,j} \coloneqq \frac{\exp(L_{i,j})}{\sum_{k=1}^{n}\exp(L_{i,k})}$. Consequently, every row of the resulting *attention map* $A \in \mathbb{R}^{n\times n}$ forms a probability distribution over the keys. The head output then aggregates the value vectors as $Y \coloneqq AV \in \mathbb{R}^{n\times d_h}$, a convex combination weighted by the attention scores.

**Injecting positions by rotating queries and keys.** While the order of tokens is crucial for language modeling, the standard dot product $q_i^\top k_j$ in Equation 3 is oblivious to token position. To inject positional information, Rotary Position Embedding (RoPE) (Su et al., 2024) rotates the coordinates of queries and keys according to their token positions prior to the dot product. More specifically, for a vector $x$ of dimension $d_h$, we split it into $d_h/2$ channels, where channel $c$ is the 2D vector formed by $(x_{2c+1}, x_{2c+2})$ $(c = 0, 1, \ldots, d_h/2 - 1)$. The rotated channel $c$ at position $i$ is defined as

$$\varphi_i^{(c)}(x) \coloneqq \begin{pmatrix} \cos(i\, f_c) & -\sin(i\, f_c) \\ \sin(i\, f_c) & \cos(i\, f_c) \end{pmatrix} \begin{pmatrix} x_{2c+1} \\ x_{2c+2} \end{pmatrix}, \quad c = 0, 1, \ldots, d_h/2 - 1, \tag{4}$$

where $f_c \in \mathbb{R}$ is a fixed frequency for channel $c$. We further define the position-dependent map $\varphi_i : \mathbb{R}^{d_h} \to \mathbb{R}^{d_h}$ as the vector formed by concatenating all the rotated channels, so that the channel $c$ of $\varphi_i(x)$ is $\varphi_i^{(c)}(x)$. For unmasked tokens ($j \le i$), the attention logit in Equation 3 becomes

$$L_{i,j} \coloneqq \frac{1}{\sqrt{d_h}}\, \varphi_i(q_i)^\top \varphi_j(k_j) = \frac{1}{\sqrt{d_h}} \sum_{c=0}^{d_h/2-1} \varphi_i^{(c)}(q_i)^\top \varphi_j^{(c)}(k_j). \tag{5}$$

**Frequency schedule.** The sequence $\{f_c\}_{c=0}^{d_h/2-1}$ is called the *frequency schedule*, which dictates how rapidly the queries and keys rotate with respect to their positions. A high frequency $f_c$ causes channel $c$ to rotate rapidly, making it ideal for distinguishing positions but unreliable for resolving long-range dependencies. Conversely, a low-frequency channel rotates slowly, making it better suited for tracking distant tokens (Barbero et al., 2025; Cheng et al., 2026). In the standard RoPE formulation, the frequency schedule follows a geometric progression $f_c \coloneqq \theta^{-2c/d_h}$ governed by a large base $\theta$ (e.g., $\theta = 10^6$ for Qwen3 (Yang et al., 2025)). Several subsequent works have proposed alternative frequency schedules (Chen et al., 2023; Peng et al., 2024; Barbero et al., 2025) to extend the model's effective context length.

## 3 WavePrune

### 3.1 WavePrune: Channel-wise Attention Truncation

In this paper, we propose WavePrune, which exploits RoPE's periodic structure to prune attention at *channel level*. Within each RoPE channel $c$, WavePrune restricts the attention computation to a local window whose size equals the wavelength $w_c \coloneqq 2\pi/f_c$. The attention logit $L_{i,j}$ is then computed as

$$L_{i,j} \coloneqq \frac{1}{\sqrt{d_h}} \sum_{c=0}^{d_h/2-1} \varphi_i^{(c)}(q_i)^\top \varphi_j^{(c)}(k_j) \cdot \mathbf{1}_{|i-j|\le w_c} \tag{6}$$

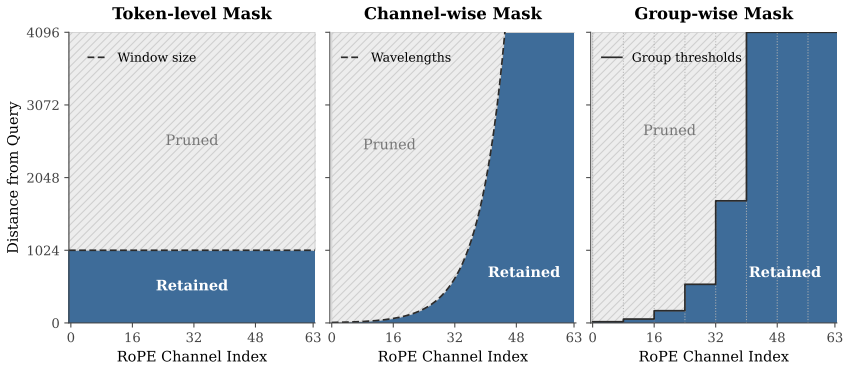



Figure 2: Conceptual comparison between the conventional token-level mask (e.g., sliding window mask), WavePrune's channel-wise mask, and its group-wise approximation. We use the group-wise approximation in our efficient implementation of WavePrune, FlashWavePrune (Section 3.3).

Conceptually, WavePrune acts as a fine-grained mask over both the sequence and head dimensions. Compared to conventional attention masks that operate on tokens, such as those in sliding window attention (SWA) (Beltagy et al., 2020), the key difference here is that WavePrune truncates the attention span in a channel-dependent manner (Figure 2).

**One period is often enough for RoPE.** Since WavePrune discards all but the first rotation period of each channel, a natural question is whether such truncation loses useful context. To examine this, we scale each channel's threshold by a coefficient $\alpha$ (i.e., $w_c := \alpha\pi/f_c$), where $\alpha = 2$ corresponds to one period, the default setting. As shown in Figure 3, the HELMET score (Qwen3-8B, 32K context) peaks slightly below $\alpha = 2$ and declines beyond it. This suggests that the first period already captures the required context information, while retaining too many periods hurts accuracy. Conversely, aggressive pruning at small $\alpha$ significantly degrades performance. Section 4 details additional experimental results demonstrating WavePrune's benefits over the original model.

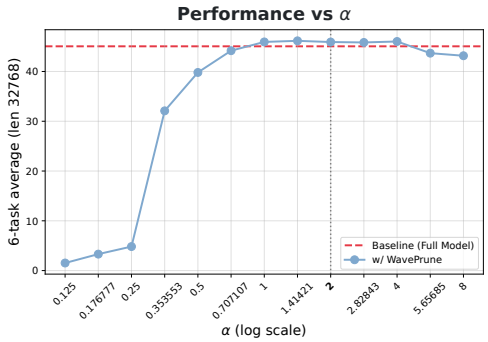



Figure 3: When tuning the pruning coefficient $\alpha$ in the exact version of WavePrune, the HELMET score peaks slightly below $\alpha = 2$ (one period) and declines beyond it. The experiment is conducted on Qwen3-8B with 32K context.

**Efficiency benefits of WavePrune.** If implemented properly, WavePrune can reduce the computational cost and memory traffic of attention. In Section 3.3, we develop an efficient implementation of WavePrune based on FlashAttention called **FlashWavePrune**, where the exact version of WavePrune is approximated by splitting the head dimensions into groups of size 16 and letting all dimensions in the same group share the same threshold for pruning (see Figure 2 for an illustration). Beyond these computational benefits, WavePrune could also reduce KV cache memory by evicting cached key components once their distance from the current query exceeds the corresponding pruning thresholds. We leave this optimization for future work.

### 3.2 Analysis: Periodicity of RoPE Distracts Attention

Having shown that WavePrune preserves model quality, a natural question is why attention beyond the first RoPE period can be safely discarded. We address this by examining LLMs' attention maps.

**"Ghost" sub-diagonals.** To reveal the underlying structural patterns of attention, we visualize the attention maps of LLMs, averaged over 100 sequences from the PG19 dataset[1] with fixed sequence length 4096. As a representative example, we plot the attention map of layer 0, head 20 of Qwen3-8B.

[1] https://huggingface.co/datasets/emozilla/pg19

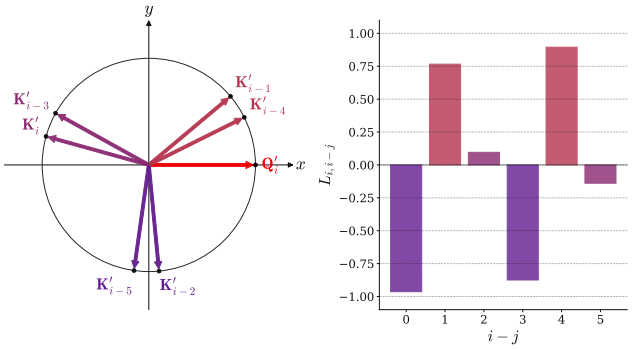



Figure 4: Illustration of position aliasing in a single RoPE channel. **Left:** rotated query $\varphi_i(q)$ and keys $\varphi_j(k)$ on the unit circle; keys three positions apart nearly coincide (e.g., $\varphi_{i-1}(k) \approx \varphi_{i-4}(k)$) as their rotation angles differ by almost $2\pi$. **Right:** the attention logits $L_{i,j}$ are therefore near-periodic with period 3 (e.g., $L_{i,i-1} \approx L_{i,i-4}$).

As shown in Figure 1a, attention concentrates near the main diagonal, specifically on the lag-1 sub-diagonal $\{A_{i,i-1}\}_{i=2}^{n}$ (Table 9), which is a typical pattern of a previous-token head (Elhage et al., 2021; Olsson et al., 2022a; Yedidia, 2023) where each query attends to the immediately preceding key. However, a considerable portion of attention also leaks onto many other sub-diagonals $\{A_{i,i-d}\}_{i=d+1}^{n}$ at larger lags $d > 1$. We refer to them as *"ghost" sub-diagonals*.

Importantly, these "ghost" sub-diagonals are not an isolated artifact but are prevalent across various heads and models (see Appendix H for other examples), and analogous patterns have been observed in prior work (Yang et al., 2026; Shen et al., 2026; Cheng et al., 2026; Du et al., 2026). As context length grows, the "ghost" sub-diagonals increasingly dilute attention away from the desired positions.

**Origin of the "ghost" sub-diagonals.** To identify their source, we decompose the attention logit matrix $L$ (Equation 5) into its per-channel contributions $L_{i,j}^{(c)} \coloneqq \varphi_i^{(c)}(q_i)^\top \varphi_j^{(c)}(k_j)$, the QK product restricted to individual 2D RoPE channels. As shown in Figure 19, we observe that the logits $L_{i,j}^{(c)}$ of high-frequency channels oscillate rapidly between positive and negative values as the lag $d \coloneqq i - j$ increases. When the per-channel logits are summed and softmax-normalized, such oscillatory structure manifests as the "ghost" sub-diagonals (Figure 1a).

**Why per-channel attention logits oscillate.** Consider a single RoPE channel with constant queries and keys (i.e., $q_i = q \in \mathbb{R}^2$ and $k_i = k \in \mathbb{R}^2$ for all $1 \le i \le n$) and rotation frequency $\theta$. Ignoring scaling factors, the channel's attention logit $L_{i,j}^{(c)}(i \ge j)$ is the dot product between the rotated query and key, which is proportional to $\cos(\theta(i-j))$. This leads to *position aliasing* (Du et al., 2026): different lags $i - j$ produce highly similar attention logits due to near-identical rotation angles $\theta(i-j)$ modulo $2\pi$. Figure 4 shows this on the unit circle, where keys several positions apart rotate to nearly the same angle, giving near-periodic attention logits and hence infinitely many "ghost" sub-diagonals.

The theorem below formalizes this intuition for multiple channels, extending the single-channel analysis in prior work (Yang et al., 2026). We consider the simplest case where all queries and all keys are both identical, so that the attention logit depends only on the lag. Even in this setting, attention is provably distracted by infinitely many "ghost" sub-diagonals; see Figure 12 for an illustration. Appendix B gives a stronger version of the theorem and its proof.

**Theorem 1 (Infinitely Many Ghost Sub-Diagonals).** *Assume $q_i = q$ and $k_i = k$ for all $i$, then $L_{i,j}$ depends only on the lag $i - j$. Let $L^\star \coloneqq \sup_{i \ge j \ge 1} L_{i,j}$ be the largest logit in the attention map. For any tolerance $\tau > 0$ and frequency schedule $\{f_c\}$, there exist infinitely many lags $\Delta \ge 0$ such that*

$$L_{i,i-\Delta} \ge L^\star - \tau \quad \forall i > \Delta. \tag{7}$$

*That is, no single sub-diagonal can dominate the attention map: the largest logit is approximately replicated onto infinitely many "ghost" sub-diagonals.*

**WavePrune eliminates "ghost" sub-diagonals.** By confining each channel to its primary period, WavePrune ensures the relative rotation angle between two positions never wraps around $2\pi$, which

Table 1: Results on the HELMET benchmark; metrics are averaged across the 64K and 128K context splits. We report cross-task averages and standard deviations (gray) to account for task-level variance. Despite pruning a significant portion of the attention computation, WavePrune maintains or improves performance on most models (Ministral3-3B, Qwen3-8B, Gemma3-12B, Qwen3-32B). Conversely, performance degrades when applying WavePrune to Llama3.1-8B.

| | | HELMET | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Model** | **Method** | **Recall** | **RAG** | **Re-rank** | **Cite** | **LongQA** | **ICL** | **Avg.** |
| Ministral3-3B | Baseline | 85.7 ± 1.2 | 58.0 ± 1.8 | 23.8 ± 1.7 | 9.7 ± 0.9 | 36.9 ± 1.7 | 20.5 ± 1.3 | 39.1 |
| | WavePrune | 86.1 ± 1.2 | 57.8 ± 1.8 | 23.2 ± 1.6 | 9.8 ± 0.9 | 36.7 ± 1.7 | 24.8 ± 1.4 | **39.7** |
| Qwen3-8B | Baseline | 63.7 ± 1.7 | 54.3 ± 1.8 | 14.7 ± 1.6 | 15.7 ± 1.2 | 42.7 ± 1.7 | 22.9 ± 1.3 | 35.7 |
| | WavePrune | 69.7 ± 1.6 | 55.6 ± 1.8 | 21.6 ± 1.8 | 15.2 ± 1.2 | 44.8 ± 1.7 | 32.9 ± 1.5 | **40.0** |
| Gemma3-12B | Baseline | 31.5 ± 1.5 | 54.1 ± 1.8 | 23.5 ± 1.8 | 0.8 ± 0.3 | 43.2 ± 1.8 | 76.4 ± 1.3 | 38.2 |
| | WavePrune | 33.1 ± 1.5 | 53.0 ± 1.8 | 23.7 ± 1.8 | 1.1 ± 0.4 | 44.5 ± 1.8 | 77.0 ± 1.3 | **38.7** |
| Qwen3-32B | Baseline | 79.1 ± 1.4 | 61.4 ± 1.7 | 34.0 ± 1.8 | 27.1 ± 1.7 | 49.5 ± 1.7 | 79.5 ± 1.3 | 55.1 |
| | WavePrune | 86.5 ± 1.2 | 61.3 ± 1.7 | 34.4 ± 1.9 | 23.9 ± 1.5 | 50.1 ± 1.7 | 81.6 ± 1.2 | **56.3** |
| Llama3.1-8B | Baseline | 84.9 ± 1.2 | 56.4 ± 1.8 | 20.0 ± 1.7 | 2.2 ± 0.0 | 39.3 ± 1.7 | 26.5 ± 1.4 | **38.2** |
| | WavePrune | 75.5 ± 1.5 | 55.6 ± 1.8 | 16.2 ± 1.7 | 2.2 ± 0.4 | 39.3 ± 1.7 | 22.5 ± 1.3 | 35.2 |

removes position aliasing at its source. As shown in Figure 20, WavePrune suppresses the oscillations in every channel, restoring a clean attention map without any "ghost" sub-diagonals (Figure 1b). Figure 8a further shows that applying WavePrune in this case indeed does not hurt model performance.

### 3.3 Efficient Implementation: FlashWavePrune

We now present **FlashWavePrune**, an efficient implementation of WavePrune that leverages its sparse attention pattern to reduce the cost of attention computation. To fully utilize Tensor Cores in modern GPUs, it uses the group-wise approximation of WavePrune (Figure 2).

**Overview.** FlashWavePrune builds on FlashAttention (FA) (Dao, 2024) and only differs in its QK multiplication path. A key challenge here is that WavePrune assigns a different pruning threshold $w_c$ to each channel $c$ (Equation (6)), so its logit matrix no longer corresponds to a standard General Matrix Multiplication (GEMM) and cannot fully utilize Tensor Cores. A straightforward implementation would instead run on CUDA cores, but then the throughput would be significantly lower. To solve this, we split the head dimension into groups of size 16 and let all dimensions in a group share one pruning threshold. Then, each group has a single mask for a given pair of $Q$ and $K$ blocks. We can therefore compute each group with a standard GEMM on Tensor Cores. For most groups, the mask is either all zeros or all ones: we skip fully masked groups altogether, saving FLOPs and, during decoding, key-cache loads. Fully retained groups follow the original FA path, and only the few remaining groups require an explicit mask after the GEMM. Appendix C details the kernel design.

## 4 Experiments

In this section, we evaluate WavePrune's effectiveness in two primary settings: training-free integration (Section 4.1) and training from scratch (Section 4.2). We then examine its system efficiency (Section 4.4) and test our design choices with ablation studies (Section 4.5). Unless specified otherwise, all experiments use FlashWavePrune (Section 3.3), our efficient implementation of WavePrune.

### 4.1 Training-Free Integration

We first apply WavePrune *training-free* to diverse LLMs, evaluating its performance on long-context and general benchmarks.

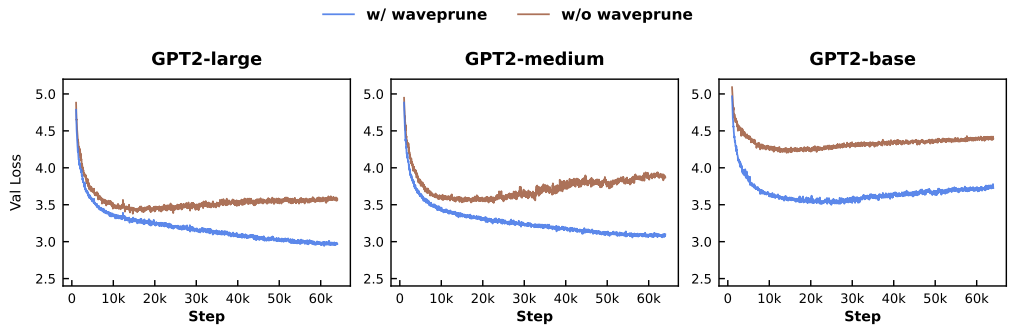



Figure 5: Length-extrapolated validation loss of GPT2-large (left), GPT2-medium (middle), and GPT2-base (right) on OpenWebText during pretraining. Training data has length 2048 and validation data has length 4096. Across all model scales, the models trained and evaluated with WavePrune consistently outperform the baselines at the extrapolated context length of 4096.

**Long-context evaluation.** We evaluate WavePrune on HELMET (Yen et al., 2025), a comprehensive benchmark suite extending to sequence lengths as long as 128K. To ensure broad applicability, we test a diverse range of long-context models: Qwen3-8B (Yang et al., 2025), Qwen3-32B, Gemma3-12B (Team et al., 2025), Ministral3-3B (Liu et al., 2026), and Llama3.1-8B (Grattafiori et al., 2024). These selections span multiple model scales, underlying architectures, and model families (Table 13). Note that these models rescale their RoPE frequencies at long contexts (e.g., via YaRN (Peng et al., 2024)); WavePrune derives each channel's wavelength from the updated frequencies. Appendix D provides additional details on the evaluation setup.

**Long-context results.** As shown in Table 1, WavePrune improves the HELMET score on four of the five models, all without any fine-tuning (length-specific 64K/128K results are in Tables 3 and 4). The gain is largest on Qwen3-8B ($35.7 \rightarrow 40.0$), where Recall and ICL improve by $6.0$ and $10.0$ points respectively; the negative log-likelihoods (NLL) are also reduced (Figure 8b). Ministral3-3B, Gemma3-12B, and Qwen3-32B improve by smaller yet consistent margins. Conversely, WavePrune fails on Llama3.1-8B; Appendix E describes our diagnosis of the failure and attempts to fix it.

**General evaluation.** We also evaluate the models on a comprehensive suite of general benchmarks (see Appendix D for details), using the lm-evaluation-harness framework (Gao et al., 2024) with default settings. As shown in Table 2, WavePrune maintains and often improves model performance across these diverse tasks, confirming no loss of general capability.

### 4.2 Pretraining Experiments

Beyond applying WavePrune to existing models, to assess its full potential, we explore training with WavePrune from scratch. We focus on the length extrapolation capability, i.e., how well the model generalizes zero-shot to sequences longer than those seen during training (Press et al., 2022).

**Experimental setup.** We train models using the GPT-2 architecture (Radford et al., 2019) at three scales: GPT2-base (117M), GPT2-medium (345M), and GPT2-large (774M). We replace the original absolute positional embedding with RoPE (Su et al., 2024) and set the RoPE base $\theta$ to $10^4$. All pretraining runs use the nanoGPT codebase (Karpathy, 2022) and the OpenWebText dataset[2]. To measure length extrapolation capabilities during training, we use *different context lengths for the training and validation sets*: the training data has a fixed sequence length of 2048, while the validation data has fixed length 4096. For each model scale, we conduct paired training runs, either enabling or disabling WavePrune. Appendix G summarizes all pretraining hyperparameters.

**Pretraining results.** Figure 5 shows the validation loss curves during pretraining. Across all model scales, WavePrune consistently improves length extrapolation by a large margin, as evidenced by the substantially lower loss on the length-extrapolated validation set, suggesting that WavePrune encourages the model to learn more length-generalizable patterns rather than overfit to the training

[2] https://huggingface.co/datasets/openwebtext

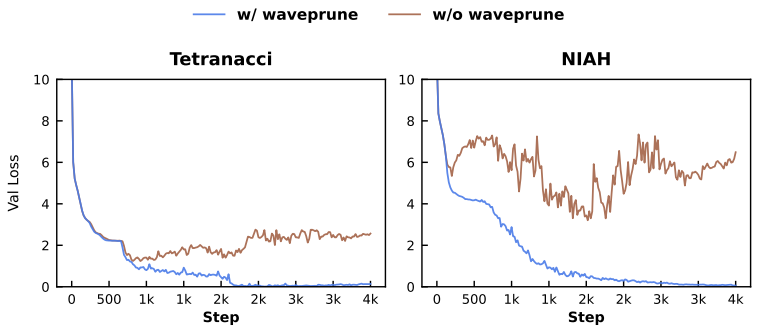



Figure 6: Length-extrapolated validation loss curves for the Tetranacci (left, for short-range dependencies) and NIAH (right, for long-range dependencies) tasks. Training set has length 2048 and validation set has length 4096. WavePrune significantly improves length extrapolation in both tasks.

sequence length. These gains over the baseline persist when increasing the RoPE base $\theta$ (Figure 9). Since the in-distribution validation loss (length 2048) with WavePrune is comparable to the baseline (Table 5a), WavePrune preserves the model's ability to fit in-distribution data.

**Comparison with training-free integration.** Since WavePrune can be independently enabled or disabled during pretraining and at inference, we evaluate all four resulting combinations on GPT2-medium and GPT2-large. As shown in Table 5b, training-free integration already improves over the baseline, yet pretraining with WavePrune consistently outperforms this plug-and-play setting: at the extrapolated length of 4096, WavePrune achieves 3.0953 vs. 3.3314 for GPT2-medium and 2.9536 vs. 3.1797 for GPT2-large. This indicates that WavePrune is most effective when incorporated into pretraining rather than applied only at inference.

### 4.3 Controlled Experiments

**Where does the improvement come from?** We hypothesize that WavePrune improves length extrapolation by resolving two distinct failure modes of standard RoPE, one affecting short-range dependencies and one affecting long-range dependencies. We state each hypothesis below and test it with a controlled experiment on GPT2-base, using the same training length of 2048 and validation length of 4096 as in Section 4.2.

**Hypothesis 1: Attention to nearby tokens is diluted by "ghost" sub-diagonals, and the dilution grows with context length.** A longer context contains more "ghost" sub-diagonals (Section 3.2), so a head attending to nearby tokens loses more attention mass at the extrapolated length. WavePrune removes them, so the attention mass should be preserved.

*Tetranacci experiment.* To isolate short-range dependencies, we train GPT2-base on the Tetranacci sequence, defined by $a_n \coloneqq (a_{n-1} + a_{n-2} + a_{n-3} + a_{n-4}) \bmod 1000$ from random initial values $a_0, a_1, a_2, a_3$ (construction details in Appendix F.1); each term depends only on the four preceding terms. Figure 6 shows that WavePrune significantly reduces the length-extrapolated validation loss. Furthermore, we find that WavePrune assigns higher attention mass to nearby tokens, and when extrapolating to longer sequences, this attention mass declines less (82.8% retained vs. 76.7% for the baseline; Table 7), which confirms Hypothesis 1.

**Hypothesis 2: Attention to distant tokens is perturbed by high-frequency channels.** Long-range retrieval relies on content matching in low-frequency channels (Barbero et al., 2025). However, under standard RoPE, the high-frequency channels have completed many rotations, so their contributions depend on the exact distance to the target, causing attention logits to fluctuate chaotically with distance (Figure 7). WavePrune removes this contribution beyond each channel's first rotation, giving a stable attention pattern that should support more robust long-range modeling (Xu et al., 2024).

*NIAH experiment.* To isolate long-range dependencies, we train GPT2-base on a needle-in-a-haystack (NIAH) task (Kamradt, 2023), in which a seven-digit "magic number" is randomly inserted within a long passage of irrelevant text and must be recalled at the end (construction details in Appendix F.2). WavePrune substantially reduces the length-extrapolated validation loss (Figure 6). Table 8 reports

the attention that each retrieval head assigns to the magic number across validation samples: while baseline heads attend to it on some samples but assign near-zero attention on others, WavePrune heads attend to it consistently, which confirms Hypothesis 2.

### 4.4 System Efficiency

**Experimental setup.** We benchmark our WavePrune kernel (Section 3.3) against FlashAttention-2 (FA2) (Dao, 2024) in terms of inference latency. Unless specified otherwise, we use `bf16` precision, a batch size of 32, Grouped-Query Attention (GQA) with 32 query heads and 8 KV heads, causal attention, head dimension $d_h = 128$, and sequence lengths $n \in \{2^{13}, 2^{14}, 2^{15}, 2^{16}, 2^{17}\}$. All measurements are collected on an NVIDIA A100 GPU as the median of 10 (prefill) / 50 (decode) timed iterations following 3 / 10 warmup runs, respectively. We report the speedup as the ratio of FA2 latency to WavePrune latency; values greater than 1 indicate WavePrune is faster.

**Kernel speed.** We compare FA2 and WavePrune across prefill and decode phases. As illustrated in Table 6, WavePrune consistently outperforms FA2 in terms of latency. This efficiency gain stems directly from our design: skipping QK multiplications decreases FLOPs crucial for prefilling, while bypassing key-cache loads reduces memory traffic crucial for decoding.

**System ablations.** In Table 11, we ablate each kernel optimization: suffix-only QK GEMM, suffix-only K loading, and split points specialization (Appendix C). The results demonstrate that removing any of these optimizations increases latency, confirming that each is necessary.

### 4.5 Ablation Studies

**Ablation on pruning thresholds and approximation.** Recall that all experiments in this section are based on FlashWavePrune, which approximates WavePrune with its group-wise approximation. To see how this approximation affects the performance of WavePrune, we conduct experiments with Qwen3-8B on the HELMET benchmark at a 32K context length. Figure 11a shows four observations: (i) the gap between FlashWavePrune and the exact version is marginal for moderately large $\alpha$; (ii) aggressive pruning at small $\alpha$ significantly degrades performance; (iii) excessively large $\alpha$ brings no further gains ($\alpha \to \infty$ corresponds to no pruning); and (iv) FlashWavePrune peaks at $\alpha = 2$, where its gap to exact channel-wise pruning is smallest.

**Ablation on the number of pruned channels.** To isolate the effect of pruning different RoPE channels, we apply FlashWavePrune to only the first $n \in \{0, 8, 16, \ldots\}$ highest-frequency channels. Since WavePrune yields its largest gains on In-Context Learning (ICL) (Table 1), we use the HELMET ICL subset for a clearer signal. As shown in Figure 11b, performance generally improves as more channels are pruned, with notable jumps at $n = 8, 16, 24$, indicating that WavePrune's benefits extend across a broad range of high- and mid-frequency channels. Beyond $n = 40$, where wavelengths exceed the context length (Table 10), pruning offers no further benefit.

## 5 Conclusion and Future Work

In this work, we demonstrate that RoPE's periodicity makes attention leak onto "ghost" sub-diagonals, which causes distractions at long contexts. WavePrune removes it by restricting each channel to its first rotation period. Without any training, this simple change improves long context performance for most LLMs we evaluate. Its sparsity pattern also lets our CUDA kernel, FlashWavePrune, delivers practical speedups. Overall, our results suggest that one period is often enough for RoPE.

We outline the following limitations and directions for future work: (i) **Performance variations.** WavePrune's training-free performance varies across different models. Future work can investigate the underlying causes through mechanistic analysis, and explore whether advanced fine-tuning strategies can bridge the performance gap. (ii) **Head-specific thresholds.** WavePrune uses the same per-channel pruning thresholds for all attention heads, but different heads may play different roles and future work can design or even learn different thresholds for different heads. (iii) **Generalizing the fine-grained attention mask.** WavePrune's fine-grained attention mask over both the sequence and head dimensions is based on RoPE. Future work can generalize this masking principle to other positional encoding methods and model architectures (Wang et al., 2026a; Thinking Machines Lab, 2026; Lin et al., 2025), which can potentially improve long-context processing and accelerate inference.

## AI USE STATEMENT

We used generative AI tools to implement auxiliary code (plotting, data processing) and to polish writing and assist in retrieving related work. Our core CUDA kernels, training code, and all other tasks requiring disclosure under the ICLR 2027 AI Policy for Authors were performed without AI. All AI-assisted code, wording, and references were manually reviewed and verified by the authors, who take full responsibility for the final content.

## REPRODUCIBILITY STATEMENT

Complete proofs and assumptions for our theoretical results are in Appendix B. Our synthetic dataset construction, pretraining hyperparameters, kernel implementation, and evaluation protocol are detailed in Appendices F, G, C, and D, respectively. We will release our code, including the CUDA kernels and training and evaluation scripts, upon publication.

## REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. arXiv preprint arXiv:2108.07732, 2021.

Federico Barbero, Alex Vitvitskyi, Christos Perivolaropoulos, Razvan Pascanu, and Petar Veličković. Round and round we go! what makes rotary positional encodings useful? In The Thirteenth International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=GtvuNrk58a.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020. URL https://arxiv.org/abs/2004.05150.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation, 2023. URL https://arxiv.org/abs/2306.15595.

Yuan Cheng, Fengzhuo Zhang, Yunlong Hou, Cunxiao Du, Chao Du, Tianyu Pang, Aixin Sun, and Zhuoran Yang. Demystifying the slash pattern in attention: The role of roPE. In ICML 2026 Workshop on Foundations of Deep Generative Models: Understanding Memorization, Generalization, and Reasoning, 2026. URL https://openreview.net/forum?id=hJHQ2EQULj.

Ting-Rui Chiang and Dani Yogatama. The rotary position embedding may cause dimension inefficiency in attention heads for long-distance retrieval. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), Findings of the Association for Computational Linguistics: ACL 2025, pp. 13552–13562, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.697. URL https://aclanthology.org/2025.findings-acl.697/.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers, 2019. URL https://arxiv.org/abs/1904.10509.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In The Twelfth International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=mZn2Xyh9Ec.

Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. Longnet: Scaling transformers to 1,000,000,000 tokens, 2023. URL https://arxiv.org/abs/2307.02486.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. LongroPE: Extending LLM context window beyond 2 million tokens. In Forty-first International Conference on Machine Learning, 2024. URL https://openreview.net/forum?id=ONOtpXLqqw.

Yufeng Du, Phillip Harris, Minyang Tian, Eliu A Huerta, Srikanth Ronanki, Subendhu Rongali, Aram Galstyan, and Hao Peng. Rope distinguishes neither positions nor tokens in long contexts, provably, 2026. URL https://arxiv.org/abs/2605.15514.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs, 2019.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. Transformer Circuits Thread, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, et al. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. How to train long-context language models (effectively). In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 7376–7399, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.366. URL https://aclanthology.org/2025.acl-long.366/.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. LM-infinite: Zero-shot extreme length generalization for large language models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pp. 3991–4008, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.222. URL https://aclanthology.org/2024.naacl-long.222/.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. Proceedings of the International Conference on Learning Representations (ICLR), 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. Advances in neural information processing systems, 2021b.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In First Conference on Language Modeling, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

Ermo Hua, Che Jiang, Xingtai Lv, Kaiyan Zhang, Youbang Sun, Yuchen Fan, Xuekai Zhu, Biqing Qi, Ning Ding, and Bowen Zhou. Fourier position embedding: Enhancing attention's periodic extension for length generalization. In Forty-second International Conference on Machine Learning, 2025. URL https://openreview.net/forum?id=ZfDNDkg7Dh.

Huiqiang Jiang, YUCHENG LI, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, et al. MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention. In The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024. URL https://openreview.net/forum?id=fPBACAbqSN.

Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen, and Xia Hu. LLM maybe longLM: Selfextend LLM context window without tuning. In Forty-first International Conference on Machine Learning, 2024. URL https://openreview.net/forum?id=nkOMLBIiI7.

Mingyu Jin, Kai Mei, Wujiang Xu, Mingjie Sun, Ruixiang Tang, Mengnan Du, Zirui Liu, and Yongfeng Zhang. Massive values in self-attention modules are the key to contextual knowledge understanding. In Forty-second International Conference on Machine Learning, 2025. URL https://openreview.net/forum?id=1SMcxxQiSL.

Gregory Kamradt. Needle in a haystack - pressure testing llms, 2023. URL https://github.com/gkamradt/LLMTest_NeedleInAHaystack.

Andrej Karpathy. NanoGPT. https://github.com/karpathy/nanoGPT, 2022.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. Cmmlu: Measuring massive multitask language understanding in chinese, 2023.

Haoran Li, Sucheng Ren, Alan Yuille, and Feng Wang. Cope: Clipped rope as a scalable free lunch for long context llms, 2026. URL https://arxiv.org/abs/2602.05258.

Zhixuan Lin, Evgenii Nikishin, Xu He, and Aaron Courville. Forgetting transformer: Softmax attention with a forget gate. In The Thirteenth International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=q2Lnyegkr8.

Alexander H. Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, et al. Ministral 3, 2026. URL https://arxiv.org/abs/2601.08584.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. Transactions of the Association for Computational Linguistics, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In International Conference on Learning Representations, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. Transformer Circuits Thread, 2022a. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads, 2022b. URL https://arxiv.org/abs/2209.11895.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In The Twelfth International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=wHBfxhZu1u.

Ofir Press, Noah Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In International Conference on Learning Representations, 2022. URL https://openreview.net/forum?id=R8sQPpGCv0.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. OpenAI Blog, 2019.

Zhenyi Shen, Junru Lu, Lin Gui, Jiazheng Li, Yulan He, di yin, and Xing Sun. SSA: Sparse sparse attention by aligning full and sparse attention outputs in feature space. In Forty-third International Conference on Machine Learning, 2026. URL https://openreview.net/forum?id=mlT8I4ri30.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. Neurocomput., 568(C), February 2024. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Yutao Sun, Li Dong, Barun Patra, Shuming Ma, Shaohan Huang, Alon Benhaim, Vishrav Chaudhary, Xia Song, and Furu Wei. A length-extrapolatable transformer. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 14590–14604, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.816. URL https://aclanthology.org/2023.acl-long.816/.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. arXiv preprint arXiv:2210.09261, 2022.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Thinking Machines Lab. Inkling: Our open-weights model. https://thinkingmachines.ai/news/introducing-inkling/, July 2026. Accessed: 2026-09-21.

Qingyuan Tian, Wenhong Zhu, Xiaoran Liu, Xiaofeng Wang, and Rui Wang. MrroPE: Mixed-radix rotary position embedding. In The Fourteenth International Conference on Learning Representations, 2026. URL https://openreview.net/forum?id=1J63FJYJKg.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Shaowen Wang, Yuke Zheng, Tansheng Zhu, Shuang Chen, Shaofan Liu, Suncong Zheng, and Jian Li. AdaroPE: Not all attention heads should rotate and scale equally. In Forty-third International Conference on Machine Learning, 2026a. URL https://openreview.net/forum?id=xNuMtLaxo5.

Suyuchen Wang, Ivan Kobyzev, Peng Lu, Mehdi Rezagholizadeh, and Bang Liu. Resonance RoPE: Improving context length generalization of large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), Findings of the Association for Computational Linguistics: ACL 2024, pp. 586–598, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.32. URL https://aclanthology.org/2024.findings-acl.32/.

Yifei Wang, Yueqi Wang, Zhenrui Yue, Huimin Zeng, Yong Wang, Ismini Lourentzou, Zhengzhong Tu, Xiangxiang Chu, and Julian McAuley. FASA: FREQUENCY-AWARE SPARSE ATTENTION. In The Fourteenth International Conference on Learning Representations, 2026b. URL https://openreview.net/forum?id=FnSgaaCEwg.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. Retrieval head mechanistically explains long-context factuality. In The Thirteenth International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=EytBpUGB1Z.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. InfLLM: Training-free long-context extrapolation for LLMs with an efficient context memory. In The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024a. URL https://openreview.net/forum?id=bTHFrqhASY.

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, 2024b. URL https://openreview.net/forum?id=NG7sS51zVF.

Mingyu Xu, Xin Men, Bingning Wang, Qingyu Zhang, Hongyu Lin, Xianpei Han, and weipeng chen. Base of roPE bounds context length. In The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024. URL https://openreview.net/forum?id=EiIelh2t7S.

Yuhui Xu, Zhanming Jie, Hanze Dong, Lei Wang, Xudong Lu, Aojun Zhou, Amrita Saha, Caiming Xiong, and Doyen Sahoo. Think: Thinner key cache by query-driven pruning. In The Thirteenth International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=n0OtGl6VGb.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Qingyue Yang, Jie Wang, Xing Li, Yinqi Bai, Tong Xialiang, Hui-Ling Zhen, Jianye HAO, Mingxuan Yuan, and Bin Li. Why attention patterns exist: A unifying temporal perspective analysis. In The Fourteenth International Conference on Learning Representations, 2026. URL https://openreview.net/forum?id=XhqoDBouWS.

Adam Yedidia. The positional embedding matrix and previous-token heads: how do they actually work? https://www.alignmentforum.org/posts/zRA8B2FJLtTYRgie6/the-positional-embedding-matrix-and-previous-token-heads-how, August 2023. AI Alignment Forum, accessed 2026-07-03.

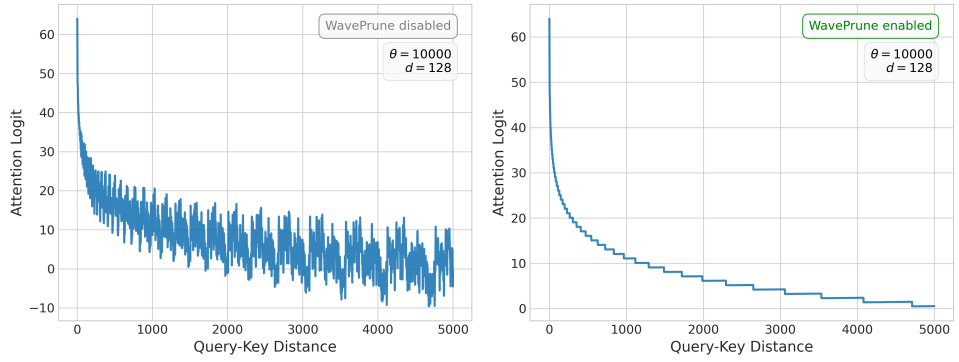
Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. HELMET: How to evaluate long-context models effectively and thoroughly. In The Thirteenth International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=293V3bJbmE.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 23078–23097, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1126. URL https://aclanthology.org/2025.acl-long.1126/.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), Advances in Neural Information Processing Systems, volume 33, pp. 17283–17297. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf.

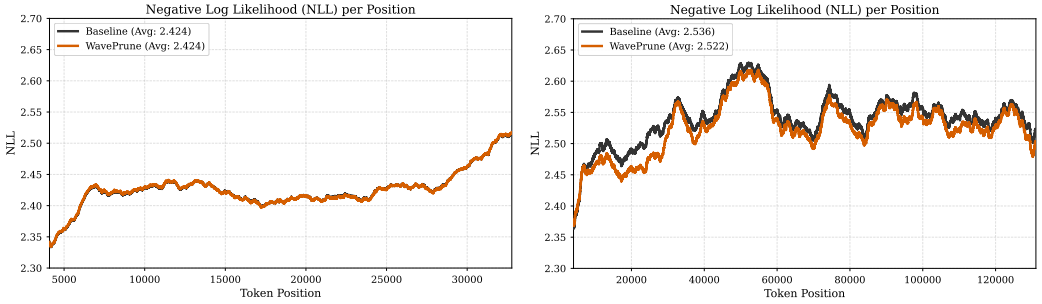
Yike Zhang, Zhiyuan He, Huiqiang Jiang, Chengruidong Zhang, Yuqing Yang, Jianyong Wang, and Lili Qiu. LeanK: Learnable k cache channel pruning for efficient decoding. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pp. 31122–31137, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1584. URL https://aclanthology.org/2025.emnlp-main.1584/.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Re, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In Thirty-seventh Conference on Neural Information Processing Systems, 2023. URL https://openreview.net/forum?id=RkRrPp7GKO.



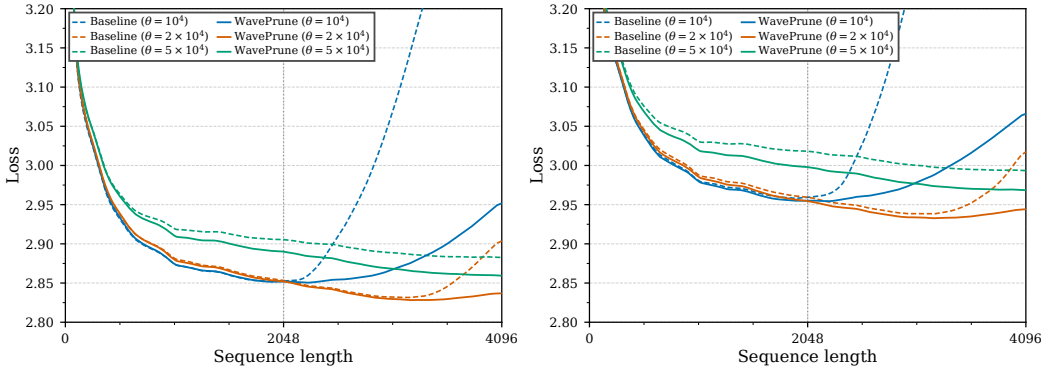

(a) Attention logit decay of standard RoPE. (b) Attention logit decay of WavePrune.

Figure 7: Attention logit decay over distance for standard RoPE (left) and WavePrune (right) under the setting of Su et al. (2024). Standard RoPE exhibits chaotic fluctuations that hinder long-range modeling, whereas WavePrune yields a stable, near-monotonic decay.




(a) NLL without YaRN. (b) NLL under YaRN.

Figure 8: Negative log-likelihoods (NLL) of Qwen3-8B, averaged over 100 PG19 samples **(a)** without YaRN and **(b)** under YaRN. WavePrune attains a lower NLL than the original model in both settings.




(a) Loss curves under GPT2-large. (b) Loss curves under GPT2-medium.

Figure 9: Validation loss versus sequence length under different RoPE base frequencies $\theta$. WavePrune attains lower validation loss than the baseline across all tested $\theta$.

Table 2: Results on general benchmarks.

| | | General Benchmarks | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **Method** | **MMLU** | **CMMLU** | **BBH** | **DROP** | **GSM8K** | **MATH** | **MBPP** | **HumanEval** | **Avg.** |
| Ministral3-3B | Baseline | 67.41 | 60.18 | 30.52 | 6.06 | 76.27 | 44.66 | 43.40 | 51.22 | 47.47 |
| | WavePrune | 67.55 | 60.31 | 31.15 | 6.02 | 77.10 | 45.22 | 45.00 | 52.44 | **48.10** |
| Qwen3-8B | Baseline | 72.92 | 76.90 | 79.47 | 10.91 | 88.32 | 52.84 | 65.40 | 63.41 | 63.77 |
| | WavePrune | 72.90 | 76.89 | 80.05 | 10.94 | 88.48 | 53.00 | 66.40 | 64.02 | **64.09** |
| Gemma3-12B | Baseline | 71.51 | 61.38 | 80.49 | 13.93 | 88.48 | 47.76 | 71.00 | 11.59 | 55.77 |
| | WavePrune | 71.54 | 61.34 | 80.22 | 13.97 | 88.70 | 47.60 | 70.60 | 15.24 | **56.15** |
| Qwen3-32B | Baseline | 80.79 | 84.16 | 47.38 | 12.06 | 63.68 | 54.14 | 35.80 | 38.41 | 52.05 |
| | WavePrune | 80.55 | 84.17 | 48.38 | 11.84 | 64.22 | 53.90 | 41.80 | 38.41 | **52.91** |

Table 3: HELMET results at the 64K context split.

| | | HELMET | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Model** | **Method** | **Recall** | **RAG** | **Re-rank** | **Cite** | **LongQA** | **ICL** | **Avg.** |
| Ministral3-3B | Baseline | 88.25 | 59.00 | 28.39 | 13.37 | 36.97 | 26.20 | 42.03 |
| | WavePrune | 89.06 | 58.75 | 28.10 | 13.76 | 37.05 | 32.40 | **43.19** |
| Qwen3-8B | Baseline | 66.19 | 57.00 | 18.93 | 19.27 | 41.63 | 23.60 | 37.77 |
| | WavePrune | 71.56 | 58.50 | 25.32 | 16.66 | 42.08 | 35.80 | **41.65** |
| Gemma3-12B | Baseline | 36.25 | 59.00 | 34.91 | 1.04 | 44.16 | 76.20 | 41.93 |
| | WavePrune | 37.69 | 58.00 | 34.72 | 1.99 | 44.97 | 76.20 | **42.26** |
| Qwen3-32B | Baseline | 76.38 | 63.25 | 37.54 | 30.95 | 46.74 | 78.00 | 55.48 |
| | WavePrune | 87.06 | 63.00 | 38.78 | 27.60 | 47.10 | 81.20 | **57.46** |
| Llama3.1-8B | Baseline | 79.56 | 58.75 | 31.42 | 4.32 | 39.83 | 31.40 | **40.88** |
| | WavePrune | 74.81 | 57.50 | 24.20 | 4.29 | 39.43 | 22.20 | 37.07 |

Table 4: HELMET results at the 128K context split.

| | | HELMET | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Model** | **Method** | **Recall** | **RAG** | **Re-rank** | **Cite** | **LongQA** | **ICL** | **Avg.** |
| Ministral3-3B | Baseline | 83.12 | 57.00 | 19.19 | 6.09 | 36.75 | 14.80 | 36.16 |
| | WavePrune | 83.19 | 56.75 | 18.33 | 5.79 | 36.37 | 17.20 | **36.27** |
| Qwen3-8B | Baseline | 61.19 | 51.50 | 10.42 | 12.17 | 43.67 | 22.20 | 33.52 |
| | WavePrune | 67.88 | 52.75 | 17.87 | 13.67 | 47.51 | 30.00 | **38.28** |
| Gemma3-12B | Baseline | 26.69 | 49.25 | 12.03 | 0.47 | 42.20 | 76.60 | 34.54 |
| | WavePrune | 28.44 | 48.00 | 12.57 | 0.11 | 43.95 | 77.80 | **35.15** |
| Qwen3-32B | Baseline | 81.81 | 59.50 | 30.39 | 23.18 | 52.24 | 81.00 | 54.69 |
| | WavePrune | 86.00 | 59.50 | 30.00 | 20.16 | 53.10 | 82.00 | **55.13** |
| Llama3.1-8B | Baseline | 90.31 | 54.00 | 8.63 | 0.04 | 38.67 | 21.60 | **35.54** |
| | WavePrune | 76.19 | 53.75 | 8.09 | 0.04 | 39.24 | 22.80 | 33.35 |

Table 5: Validation loss of GPT2-medium and GPT2-large on OpenWebText at lengths 2048 and 4096. "Pretraining" indicates whether WavePrune is enabled during pretraining, and "Eval" indicates whether WavePrune is enabled during evaluation.

(a) Eval @2048

| Model | Pretraining | Eval | |
|---|---|---|---|
| | | Baseline | WavePrune |
| GPT2-medium | Baseline | 3.0270 | 3.0261 |
| | WavePrune | 3.6210 | **3.0196** |
| GPT2-large | Baseline | 2.9009 | 2.9003 |
| | WavePrune | 3.7816 | **2.8974** |

(b) Eval @4096

| Model | Pretraining | Eval | |
|---|---|---|---|
| | | Baseline | WavePrune |
| GPT2-medium | Baseline | 3.8564 | 3.3314 |
| | WavePrune | 4.3849 | **3.0953** |
| GPT2-large | Baseline | 3.5618 | 3.1797 |
| | WavePrune | 4.5162 | **2.9536** |

Table 6: Full comparison between FA2 and WavePrune latency across prefill and decode. We use the same setting as Section 4.4, except that sequence lengths extend to 128K.

| **Setting** | **Length** | **FA2** | **WavePrune** | **Speedup** |
|---|---|---|---|---|
| Prefill, H64 | 8K | 46.96 | 46.07 | **1.02**× |
| | 16K | 183.54 | 174.37 | **1.05**× |
| | 32K | 731.00 | 673.71 | **1.09**× |
| | 64K | 2934.19 | 2659.50 | **1.10**× |
| | 128K | 12096.79 | 10775.51 | **1.12**× |
| Prefill, H128 | 8K | 82.17 | 75.03 | **1.10**× |
| | 16K | 325.02 | 287.66 | **1.13**× |
| | 32K | 1306.79 | 1132.27 | **1.15**× |
| | 64K | 5337.95 | 4490.98 | **1.19**× |
| | 128K | OOM | OOM | N/A |
| Decode, H64 | 8K | 0.448 | 0.432 | **1.04**× |
| | 16K | 0.803 | 0.767 | **1.05**× |
| | 32K | 1.567 | 1.490 | **1.05**× |
| | 64K | 3.095 | 2.938 | **1.05**× |
| | 128K | 6.155 | 5.838 | **1.05**× |
| Decode, H128 | 8K | 0.826 | 0.734 | **1.13**× |
| | 16K | 1.538 | 1.277 | **1.20**× |
| | 32K | 3.059 | 2.459 | **1.24**× |
| | 64K | 6.111 | 4.816 | **1.27**× |
| | 128K | 12.193 | 9.528 | **1.28**× |

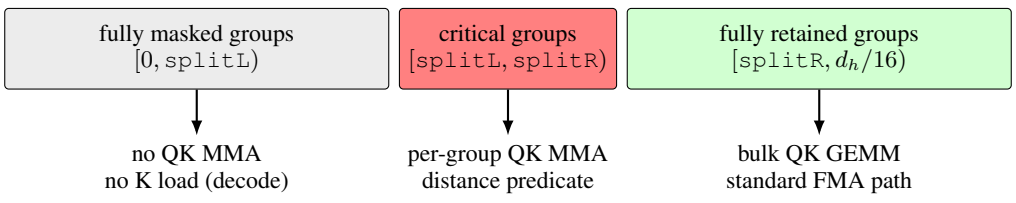


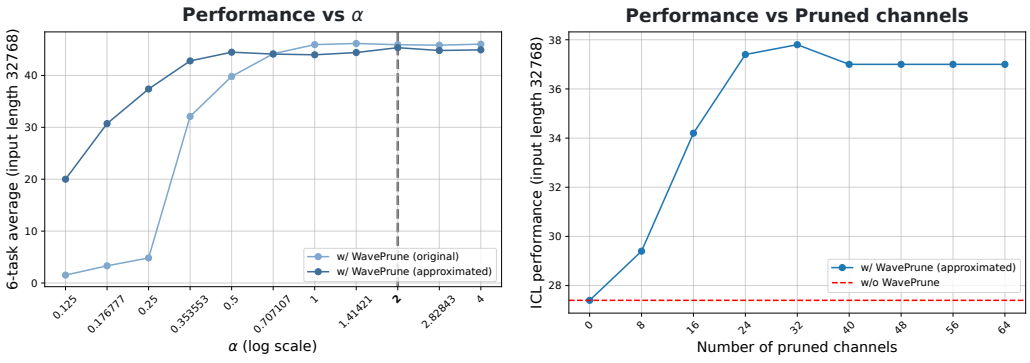

Figure 10: Head-dimension groups are partitioned into three regions with distinct compute paths.

Table 7: Average attention scores to neighboring tokens (distance $\leq 15$), averaged across all 12 GPT2-base layers, evaluated on the Tetranacci training (length 2048) and validation (length 4096) sets. WavePrune assigns more attention mass to neighboring tokens at both lengths and retains more of it at the extrapolated length, indicating it better preserves local attention under length extrapolation.

| Method | Train (len 2048) | Val (len 4096) | Retention (Val / Train) |
|---|---|---|---|
| WavePrune | **0.179** | **0.148** | **82.8%** |
| Baseline | 0.161 | 0.123 | 76.7% |

Table 8: Average attention scores to the "magic number" for each retrieval head (Wu et al., 2025), evaluated on samples from the NIAH validation set. We select the first eight retrieval heads of each model, sorted by layer and head index. For each validation set sample, we report the average attention allocated to tokens near the magic number (distance $\leq 5$) from tokens in the final answer span. Notably, the WavePrune model's retrieval heads are consistently activated across different samples, while the baseline model's heads frequently fail to attend to the magic number at all.

| Head | WavePrune | | | | Baseline | | | |
|---|---|---|---|---|---|---|---|---|
| | Sample 1 | Sample 2 | Sample 3 | Sample 4 | Sample 1 | Sample 2 | Sample 3 | Sample 4 |
| 1 | 0.750 | 0.750 | 0.798 | 0.751 | 0.245 | 0.000 | 0.000 | 0.000 |
| 2 | 0.749 | 0.742 | 0.523 | 0.516 | 0.750 | 0.000 | 0.000 | 0.000 |
| 3 | 0.748 | 0.412 | 0.800 | 0.250 | 0.000 | 0.000 | 0.000 | 0.373 |
| 4 | 0.413 | 0.516 | 0.603 | 0.502 | 0.000 | 0.250 | 0.600 | 0.000 |
| 5 | 0.944 | 0.000 | 0.000 | 0.986 | 0.021 | 1.000 | 1.000 | 0.003 |
| 6 | 0.750 | 0.750 | 0.800 | 0.750 | 0.000 | 0.500 | 0.400 | 0.500 |
| 7 | 0.751 | 0.750 | 0.800 | 0.750 | 0.001 | 0.750 | 0.984 | 0.500 |
| 8 | 0.394 | 0.500 | 0.800 | 0.250 | 0.000 | 1.000 | 0.008 | 0.000 |
| **Loss** | **0.030** | **0.005** | **0.003** | **0.001** | 6.328 | 4.734 | 10.840 | 5.767 |




(a) Ablation on the pruning coefficient $\alpha$.

(b) Ablation on the number of pruned channels.

Figure 11: Ablation studies of WavePrune on HELMET using Qwen3-8B at 32K context length. **(a)** Effect of the pruning coefficient $\alpha$ and comparison between the exact and approximated versions of WavePrune. **(b)** Effect of pruning different numbers of channels on ICL performance.

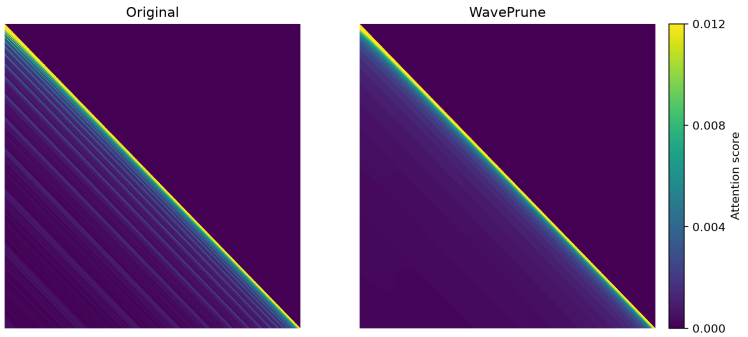



Figure 12: Attention maps under constant queries and keys ($q_i = k_i = \mathbf{1}$; $N = 1024$, $d_h = 64$, $\theta = 10^4$), where the intended lag is 0 (i.e., each token should attend only to itself). RoPE alone produces many ghost sub-diagonals (left); WavePrune suppresses these artifacts and recovers a clean attention map (right).

## A Related Work

**Understanding Rotary Position Embeddings.** RoPE (Su et al., 2024) encodes relative positions by rotating queries and keys at different frequencies along the head dimension. Mechanistic studies reveal that models partition these frequencies for distinct purposes. Barbero et al. (2025) show that low frequencies retrieve semantic context, whereas high frequencies produce position-based attention patterns with massive activation values (Jin et al., 2025). These patterns support various specialized attention mechanisms, such as retrieval heads (Wu et al., 2025) and previous-token heads (Elhage et al., 2021; Olsson et al., 2022a; Yedidia, 2023). Cheng et al. (2026) further identify "slash-dominant heads" (SDHs), which leverage RoPE's high-frequency channels to create strong sub-diagonal lines at specific lags $\Delta$ in the attention map. They categorize these lags as small-$\Delta$ and large-$\Delta$, the latter of which remain largely unexamined and are implicitly presumed necessary. Yang et al. (2026) likewise observe these periodically repeated attention peaks; Du et al. (2026) prove that such patterns are an inherent consequence of RoPE and can induce positional ambiguity and inversion. Building on these findings, WavePrune prunes such distractions.

**Rotary Position Embeddings and Length Extrapolation.** To enhance length extrapolation (Press et al., 2022), one line of work modifies RoPE frequencies. The central motivation is that RoPE produces out-of-distribution rotation angles under extended contexts, particularly for low-frequency channels that never complete a full $2\pi$ rotation within the training length (Chen et al., 2023). Accordingly, Peng et al. (2024); Ding et al. (2024); Wang et al. (2024); Tian et al. (2026); Li et al. (2026) rescale RoPE frequencies to keep rotation angles in-distribution while preserving high-frequency channels for position sensitivity. A second line of work (Sun et al., 2023; Xiao et al., 2024a; Jin et al., 2024; Han et al., 2024) instead modifies attention mechanics, e.g., via position-dependent query scaling or altered attention masks. Unlike these methods, WavePrune's attention mechanism is tied to RoPE: it eliminates RoPE's periodicity without altering the frequency spectrum, and can be combined with frequency-rescaling methods to further improve length extrapolation.

**Efficient Attention Mechanisms.** Efficient attention mechanisms reduce the $\mathcal{O}(N^2D)$ cost along sequence length $N$ and head dimension $D$. Along the sequence dimension, some approaches (Child et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020; Ding et al., 2023; Xiao et al., 2024b) design fixed sparse patterns to limit the evaluated token pairs, while others (Zhang et al., 2023; Jiang et al., 2024; Yuan et al., 2025) dynamically select tokens at inference. Recent works prune the sparsely utilized head dimension (Barbero et al., 2025; Wang et al., 2026a). FASA (Wang et al., 2026b) uses a small subset of dominant RoPE frequencies to guide token selection. ThinK (Xu et al., 2025) prunes the key cache in a query-dependent manner, while LeanK (Zhang et al., 2025) learns static channel sparsity. WavePrune unifies sequence- and head-wise pruning by tailoring local attention patterns to each RoPE frequency, opening an underexplored design space for attention mechanisms.

# B RoPE and Attention Periodicity: Theory and Implications

## B.1 Slash-Dominant Attention Pattern

After training, the model develops various structures in the attention map $A$ to implement diverse functionalities. In this section, we focus on a specific structure known as the **slash-dominant pattern** (Cheng et al., 2026).

We say that the attention score matrix $A \in \mathbb{R}^{n\times n}$ exhibits a slash-dominant pattern if there exists a constant $\Delta \in \mathbb{N}$ such that $A$ has large values along the $\Delta$-offset sub-diagonal $\{A_{i,i-\Delta}\}_{i=\Delta+1}^{n}$. Intuitively, this means each token $i (i > \Delta)$ attends strongly to the token at position $i - \Delta$, which enables information to flow in a position-wise "slash-like" manner. Such patterns have been empirically observed in various LLMs and play a critical role in tasks such as in-context learning (ICL) (Cheng et al., 2026; Elhage et al., 2021; Olsson et al., 2022b).

We formalize the notion of slash-dominant attention patterns as follows:

**Definition 1 (Slash-Dominant Pattern).** *For any lag $\Delta \in \mathbb{N}$ and $\epsilon, \eta > 0$, the attention score matrix $A \in \mathbb{R}^{n\times n}$ (or, equivalently, the attention head) is said to be $(\Delta, \epsilon, \eta)$-slash-dominant, if*

$$\sum_{i=\Delta+1}^{n} \mathbf{1}\left\{A_{i,i-\Delta} \geq 1-\epsilon\right\} \geq (1-\eta)(n-\Delta) \tag{8}$$

## B.2 Main Theoretical Result

### B.2.1 Assumptions and Definitions

Before stating our main theorem, we introduce the following assumptions and definitions.

**Assumption 1 (Symmetric i.i.d. Distributions of Queries and Keys).** *There exist distributions $\mathcal{Q}, \mathcal{K}$ over $\mathbb{R}^{d_h}$ such that the queries $q_i$ are i.i.d. samples from $\mathcal{Q}$ and the keys $k_i$ are i.i.d. samples from $\mathcal{K}$. Furthermore, $\mathcal{Q}$ and $\mathcal{K}$ are symmetric distributions centered at their respective means $q$ and $k$ (e.g., multivariate Gaussians); i.e., for any measurable set $S \subseteq \mathbb{R}^{d_h}$, let $\widetilde{S} \coloneqq \{2q - x \mid x \in S\}$ be the reflection of $S$ about $q$, then $\Pr(q_i \in S) = \Pr(q_i \in \widetilde{S})$. The same holds for $\mathcal{K}$ with mean $k$.*

**Definition 2 (Channel Sensitivity Measures and $\tau$-Truncation Sparsity).** *For $c = 0, 1, \ldots, d_h/2 - 1$, let $m_c^Q \coloneqq \sqrt{q_{2c+1}^2 + q_{2c+2}^2}$ and $m_c^K \coloneqq \sqrt{k_{2c+1}^2 + k_{2c+2}^2}$ denote the magnitudes of channel $c$ of the mean vectors $q$ and $k$, respectively. We define the **sensitivity measure** of channel $c$ as*

$$t_c \coloneqq \frac{2\pi m_c^Q m_c^K}{\sqrt{d_h}} \quad (c = 0, 1, \ldots, d_h/2 - 1) \tag{9}$$

*For any $\tau > 0$, the **$\tau$-truncation sparsity** of the sensitivity measures $t$ is defined as*

$$\sigma_\tau(t) \coloneqq \min\left\{\|z\|_0 \mid \|z - t\|_1 \leq \tau,\, z \in \mathbb{R}^{d_h/2}\right\} \tag{10}$$

### B.2.2 Main Theorem

With Assumption 1 and Definition 2 in place, we are now ready to state our main theorem. Intuitively, it shows that the attention logit matrix $L$ (Equation 3) is approximately periodic[3] across different position lags (see Figure 4 for an illustration). Consequently, the $\Delta$-offset sub-diagonal (the intended lag) competes with its periodic repetitions, causing attention weight to distribute across multiple ghost sub-diagonals (sub-diagonals with unexpectedly high attention scores) rather than concentrating on the desired lag $\Delta$.

[3]Here, we use the term "periodic" to loosely refer to the *similarity* of attention logits at positions separated by certain lags, rather than the strict mathematical property.

**Theorem 1 (Periodicity of Attention Logits, Formal Statement).** *Let the query and key distributions $\mathcal{Q}, \mathcal{K}$ satisfy Assumption 1. For any $\tau > 0$, there exist infinitely many periods $P \in \mathbb{N}^+$ such that*

$$\forall \Delta \geq 0, \forall i > \Delta + P, \quad \Pr\left(L_{i,i-\Delta} - L_{i,i-\Delta-P} \leq 2\tau\right) \geq \frac{1}{4} \tag{11}$$

*Let $t^s$ be the sensitivity measures (Definition 2) sorted in descending order. Then, the minimum period $P_{\min}$ satisfying the above condition is upper bounded by a sparsity-dependent quantity:*

$$P_{\min} \leq \prod_{i=1}^{\sigma_\tau(t)} \left\lceil \frac{\sigma_\tau(t) \cdot t_i^s}{\tau} \right\rceil \tag{12}$$

*If $t_i^s = \mathcal{O}(1)$ for all $i$[4] and $\sigma_\tau(t) \leq S$ for some constant $S \geq 0$, the above bound can be simplified to*

$$P_{\min} = \mathcal{O}\left(\left(\frac{1}{\tau}\right)^S\right) \tag{13}$$

### B.2.3 Implication on Attention Scores

According the definition of attention score matrix $A_{i,j} := \frac{\exp(L_{i,j})}{\sum_{k=1}^{n} \exp(L_{i,k})}$, which is a softmax of the attention logit matrix, the periodicity of attention logits established in Theorem 1 directly implies that attention is prevented from concentrating exclusively on the desired sub-diagonal. We formalize this implication in the following corollary.

**Corollary 1 (Upper Bound on Attention Scores).** *Under the same conditions as Theorem 1, for any $\epsilon \in \left(0, \frac{1}{2}\right)$, let $\tau_\epsilon = \ln\left(\frac{1-\epsilon}{\epsilon}\right)$ be the log-odds associated with $\epsilon$. Then, there exists $P_{\min} \in \mathbb{N}^+$ such that*

$$\forall i > \Delta + P, \quad \Pr\left(A_{i,i-\Delta} \leq 1 - \epsilon\right) \geq \frac{1}{4} \tag{14}$$

where $P_{\min}$ is upper bounded by the same quantity as in Equation 12 with $\tau = \frac{\tau_\epsilon}{2}$.

**Remark (Fragility of Slash-Dominant Patterns).** Corollary 1 indicates that for sufficiently small $\epsilon$ and $\eta$ in Definition 1, any slash-dominant pattern will eventually collapse as the sequence length increases. Note that this does not contradict the empirical observations in Cheng et al. (2026): their study identifies slash-dominant heads using loose thresholds (e.g., 0.1 instead of $1 - \epsilon$) and evaluates them on relatively short context lengths (up to 6000 tokens).

## B.3 Empirical Evidences and Analysis

**Sparsity of channel utilization and the periodicity issue.** As illustrated in Theorem 1, the sparsity of channel utilization is a dominant factor determining the period of attention logits. Extensive prior work has observed sparsity in real models (Xu et al., 2025; Barbero et al., 2025; Chiang & Yogatama, 2025; Wang et al., 2026b;a), and we also empirically verify this. For instance, as shown in Figure 19, only 7 out of 64 channels are actively utilized in Qwen3-8B's layer 0 head 20 (a slash-dominant head with $\Delta = 1$; as evidenced by the attention score statistics in Table 9). Another example is shown in Figure 21. This can indeed be understood as a low $\tau$-truncation sparsity $\sigma_\tau(t)$ (Theorem 1), which often leads to a small period $P_{\min}$ by Theorem 1.[5]

As evidenced in Figure 1a, Figure 15a and Figure 16a, such periodicity causes attention logits of various models to spike at multiple ghost sub-diagonals. We argue that this behavior is detrimental to model performance on non-periodic tasks (including almost all NLP tasks), since it prevents the model from losslessly passing signals along a single desired lag.

[4]This assumption is reasonable since $X$ is typically normalized by LayerNorm, which means $m_c^Q$ and $m_c^K$ are bounded by constants.

[5]While channel sparsity is empirically observed, the mathematically defined $\sigma_\tau(t)$ may not be always small because inactive channels can still have non-negligible sensitivity values (note that two vectors with large magnitudes can yield a near-zero dot product). This gap primarily arises from omitting the rotation frequencies $f_i$ in Definition B.2.1 for technical tractability; readers may intuitively think of these inactive channels as having zero query/key magnitudes. We leave tighter analysis to future work.

Table 9: Percentile-based attention score statistics for Qwen3-8B layer 0 head 20 across different position lags ($\Delta$). We compute attention scores for each of the 100 PG19 samples, extract the percentiles for each sub-diagonal, and then average these percentiles. The $\Delta = 1$ sub-diagonal exhibits significantly higher attention scores, confirming a slash-dominant pattern with $\Delta = 1$.

| Percentile | $\Delta = 0$ | $\Delta = 1$ | $\Delta = 2$ | $\Delta = 3$ | $\Delta = 4$ |
|---|---|---|---|---|---|
| 75th | 0.0126 | **0.0611** | 0.0178 | 0.0065 | 0.0037 |
| 90th | 0.0255 | **0.1420** | 0.0399 | 0.0182 | 0.0112 |
| 95th | 0.0392 | **0.1573** | 0.0634 | 0.0295 | 0.0185 |

**Eliminating ghost sub-diagonals through WavePrune.** The periodicity issue identified in Theorem 1 can be resolved via WavePrune by truncating attention beyond each channel's wavelength. As can be seen in Figures 15 and 16, the ghost sub-diagonals are effectively eliminated after applying WavePrune. This improvement is particularly evident in the channel-wise decomposition of attention logits (Figure 19 and Figure 20), where periodic patterns are suppressed across all channels.

**WavePrune breaks Theorem 1.** Beyond empirical observations, we also theoretically show that WavePrune breaks the attention logits periodicity established in Theorem 1. Consider a minimal 2D example comprising a single channel with rotation frequency $\theta \in (0, 2\pi)$ and wavelength $w \coloneqq 2\pi/\theta$. Suppose the query and key are unit vectors $q, k \in \mathbb{R}^2$ with an initial relative angle $\omega = -\theta$. Ignoring the $1/\sqrt{d_h}$ scaling, the attention logit under WavePrune (Equation 6) at lag $\Delta$ is given by $L_\Delta = \cos(\Delta\theta + \omega) \cdot \mathbf{1}_{\Delta \le w} = \cos(\theta(\Delta - 1)) \cdot \mathbf{1}_{\Delta \le w}$. At $\Delta = 1$, the RoPE rotation exactly cancels the initial relative angle, yielding a maximum logit $L_1 = \cos(0) = 1$. For $1 < \Delta \le w$, $\theta(\Delta - 1)$ cannot reach any non-zero multiple of $2\pi$, so $L_\Delta < 1$; for $\Delta > w$, the indicator evaluates to zero, yielding a constant logit $L_\Delta = 0$. By permanently forcing the logit to zero beyond the wavelength, WavePrune prevents any periodic spikes, invalidating Theorem 1.

Finally, note that the periodicity issue is not exclusive to slash-dominant heads but is observable across numerous other types of attention heads (Figure 17 and 18), as Theorem 1 relies on no assumptions regarding the attention pattern. Nonetheless, since the elimination of ghost sub-diagonals is particularly beneficial for slash-dominant heads, we use them as a representative example.

### B.4 Proof of Theorem 1

**Proof roadmap.** The proof consists of two main steps. First, we assume $\mathcal{Q}$ and $\mathcal{K}$ are point mass distributions concentrated at $q$ and $k$, respectively (i.e., $q_i = q$ and $k_i = k$ for all $i$), and prove the deterministic version of Theorem 1 Then, we extend the result to general symmetric distributions, leveraging Lemma 2 to establish the $\frac{1}{4}$ probability bound.

#### B.4.1 Step 1: Periodicity under Point Mass Distributions

**Lemma 1 (Periodicity under Point Mass Distributions).** *Let $\mathcal{Q} = \delta_q$ and $\mathcal{K} = \delta_k$ be point mass distributions concentrated at $q$ and $k$, respectively. Then, there exists infinitely many periods $P \in \mathbb{N}^+$ such that*

$$\forall \Delta \ge 0,\ \forall i > \Delta + P, \quad L_{i,i-\Delta} - L_{i,i-\Delta-P} \le 2\tau. \tag{15}$$

*Let $t^s$ be the sensitivity measures (Definition 2) sorted in descending order. Then, the minimum period $P_{\min}$ satisfying the above condition is upper bounded by a sparsity-dependent quantity:*

$$P_{\min} \le \prod_{i=1}^{\sigma_\tau(t)} \left\lceil \frac{\sigma_\tau(t) \cdot t_i^s}{\tau} \right\rceil. \tag{16}$$

**Proof.** Let $m_c^Q, m_c^K, \omega_c^Q, \omega_c^K$ denote the magnitudes and angles of channel $c$ of $q$ and $k$ for $0 \le c \le \frac{d_h}{2} - 1$, respectively. i.e.,

$$\begin{aligned} q_{2c+1} &= m_c^Q \cos(\omega_c^Q),\ \ q_{2c+2} = m_c^Q \sin(\omega_c^Q) \\ k_{2c+1} &= m_c^K \cos(\omega_c^K),\ k_{2c+2} = m_c^K \sin(\omega_c^K). \end{aligned} \tag{17}$$

Denote $\omega_c \coloneqq \omega_c^Q - \omega_c^K$ as the angle difference between channel $c$ of $q$ and $k$, then the attention logit $L_{i,j}(i \geq j)$ can be expressed as:

$$\begin{aligned} L_{i,j} &= \frac{1}{\sqrt{d_h}} \varphi_i(q_i)^\top \varphi_j(k_j) \\ &= \frac{1}{\sqrt{d_h}} \sum_{c=0}^{d_h/2-1} m_c^Q m_c^K \cos\left((i-j)\, f_c + \omega_c\right). \end{aligned} \tag{18}$$

Let $\text{wrap}(x)$ denote the unique value $v \in [-\pi, \pi)$ that satisfies $\frac{x-v}{2\pi} \in \mathbb{Z}$. Then, the attention logit difference $L_{i,j_1} - L_{i,j_2}(i \geq j_1 > j_2)$ can be bounded as follows:

$$\begin{aligned} L_{i,j_1} - L_{i,j_2} &= \frac{1}{\sqrt{d_h}} \sum_{c=0}^{d_h/2-1} m_c^Q m_c^K \Big( \cos\left((i-j_1)\, f_c + \omega_c\right) - \cos\left((i-j_2)\, f_c + \omega_c\right) \Big) \\ &\leq \frac{1}{\sqrt{d_h}} \sum_{c=0}^{d_h/2-1} m_c^Q m_c^K \left| \cos\left((i-j_1)\, f_c + \omega_c\right) - \cos\left((i-j_2)\, f_c + \omega_c\right) \right| \\ &\leq \frac{1}{\sqrt{d_h}} \sum_{c=0}^{d_h/2-1} m_c^Q m_c^K \left| \text{wrap}\left(j_1\, f_c - j_2\, f_c\right) \right|, \end{aligned} \tag{19}$$

where the last inequality holds because $\cos$ is $2\pi$-periodic and 1-Lipschitz.

For any $1 \leq j \leq i$, let $x_j \in [-\pi, \pi)^{d_h/2}$ be the vector defined as $x_{j,c} \coloneqq \text{wrap}(j\, f_c)$, then

$$\begin{aligned} L_{i,j_1} - L_{i,j_2} &\leq \frac{1}{\sqrt{d_h}} \sum_{c=0}^{d_h/2-1} m_c^Q m_c^K \left| \text{wrap}(x_{j_1,c} - x_{j_2,c}) \right| \\ &\leq \frac{1}{\sqrt{d_h}} \sum_{c=0}^{d_h/2-1} m_c^Q m_c^K \left| x_{j_1,c} - x_{j_2,c} \right|. \end{aligned} \tag{20}$$

We define the metrics $\rho_c, \rho$ on $V \coloneqq [-\pi, \pi)^{d_h/2}$ as follows:

$$\rho_c(x, y) \coloneqq \frac{1}{\sqrt{d_h}} \sum_{c=0}^{d_h/2-1} m_c^Q m_c^K \left| \text{wrap}(x_c - y_c) \right|, \tag{21}$$

$$\rho(x, y) \coloneqq \frac{1}{\sqrt{d_h}} \sum_{c=0}^{d_h/2-1} m_c^Q m_c^K \left| x_c - y_c \right|. \tag{22}$$

Then

$$L_{i,j_1} - L_{i,j_2} \leq \rho_c(x_{j_1}, x_{j_2}) \leq \rho(x_{j_1}, x_{j_2}). \tag{23}$$

Suppose $V$ can be covered by $M$ sets $C_1, C_2, \ldots, C_M$ with $\rho$-diameter at most $2\tau$, i.e.,

$$\bigcup_{m=1}^{M} C_m = V, \quad \forall m,\, \forall x, y \in C_m,\, \rho(x, y) \leq 2\tau. \tag{24}$$

Then, for any $\Delta \geq 0$ and $i > \Delta + M$, there must exist $i - \Delta - M \leq j_2 < j_1 \leq i - \Delta$ such that $x_{j_1}, x_{j_2} \in C_m$ for some $m$ because of the pigeonhole principle. Consequently, we have $\rho_c(x_{j_1}, x_{j_2}) \leq \rho(x_{j_1}, x_{j_2}) \leq 2\tau$. Note that $\rho_c(x_{j_1}, x_{j_2})$ only depends on the position difference $j_1 - j_2$ rather than the specific values of $j_1$ and $j_2$, Thus, we can set $P \coloneqq j_1 - j_2$ and yield the desired periodicity (Equation 15):

$$L_{i,i-\Delta} - L_{i,i-\Delta-P} \leq \rho_c(x_{i-\Delta}, x_{i-\Delta-P}) = \rho_c(x_{j_1}, x_{j_2}) \leq 2\tau, \tag{25}$$

which holds for all $i > \Delta + P$ since $L_{i,i-\Delta} - L_{i,i-\Delta-P}$ is independent of $i$.

If $\rho_c(x_{i-\Delta}, x_{i-\Delta-P}) = 0$, then $\rho_c(x_{i-\Delta}, x_{i-\Delta-kP}) = 0$ for all $k \in \mathbb{N}^+$, which means any integer multiple of $P$ can serve as a period. Otherwise, we can set $\tau' \coloneqq \frac{1}{3}\rho_c(x_{i-\Delta}, x_{i-\Delta-P}) > 0$ as the new threshold and repeat the above argument to find another period $P'$ such that $\forall \Delta \geq 0$, $\forall i > \Delta + P'$, $L_{i,i-\Delta} - L_{i,i-\Delta-P'} \leq 2\tau' < 2\tau$. By iterating this process, we can find infinitely many periods that satisfy Equation 15.

Now, it suffices to show that the covering number $M$ is upper bounded by the quantity in Equation 16. Note that the metric space $(V, \rho)$ is isometric to $V_0 \coloneqq [0, t_0] \times [0, t_1] \times \ldots \times [0, t_{d_h/2-1}]$ equipped with the $\ell_1$ metric, where $t_c$ is the sensitivity measure defined in Equation 9. This reduces the problem to covering $V_0$ with sets of $\ell_1$-diameter at most $2\tau$.

We use a simple grid-based approach to cover $V_0$. Specifically, let $g_0, g_1, \ldots, g_{d_h/2-1} > 0$ denote the grid sizes for each dimension, then we partition $V_0$ into axis-aligned hyperrectangles by placing hyperplanes at intervals of $g_c$, ensuring each hyperrectangle has $\ell_1$-diameter at most $\sum_c g_c \leq 2\tau$. The total number of hyperrectangles is then

$$M = \prod_{c=0}^{d_h/2-1} \left\lceil \frac{t_c}{g_c} \right\rceil, \tag{26}$$

which leads to the following optimization problem for minimizing $M$:

$$\begin{aligned} \min \quad & \prod_{c=0}^{d_h/2-1} \left\lceil \frac{t_c}{g_c} \right\rceil \\ \text{s.t.} \quad & \sum_{c=0}^{d_h/2-1} g_c \leq 2\tau \\ & g_c > 0 \quad (c = 0, 1, \ldots, d_h/2 - 1). \end{aligned} \tag{27}$$

Denote $\kappa \coloneqq \sigma_\tau(t)$ as the $\tau$-truncation sparsity of $t$ (Definition 2), and $T \coloneqq \{c_1, c_2, \ldots, c_\kappa\}$ as the indices of the top $\kappa$ sensitivity measures. By setting $g_c \coloneqq \frac{\tau}{\kappa}$ for $c \in T$ and $g_c \coloneqq t_c$ for $c \notin T$, we have

$$\sum_c g_c = \sum_{c \in T} \frac{\tau}{\kappa} + \sum_{c \notin T} t_c \leq 2\tau, \tag{28}$$

$$M = \prod_{c \in T} \left\lceil \frac{t_c}{\frac{\tau}{\kappa}} \right\rceil \cdot \prod_{c \notin T} \left\lceil \frac{t_c}{t_c} \right\rceil = \prod_{i=1}^{\kappa} \left\lceil \frac{\kappa \cdot t_i^s}{\tau} \right\rceil, \tag{29}$$

which completes the proof.

### B.4.2 Step 2: Extension to General Symmetric Distributions

**Lemma 2 (Symmetric Distributions and Probability Bounds).** *Let the query and key distributions $\mathcal{Q}, \mathcal{K}$ satisfy Assumption 1. Denote $L^0$ as the attention logit matrix when $q_i = q$ and $k_i = k$ for all $i$. Then, for any $j_1 < j_2 \leq i$, we have*

$$\Pr\left(L_{i,j_1} - L_{i,j_2} \leq L^0_{i,j_1} - L^0_{i,j_2}\right) \geq \frac{1}{4}. \tag{30}$$

To prove Lemma 2, we need the following lemmas regarding symmetric distributions.

**Lemma 3.** *Let $(\Omega, \mathscr{F}, \mathrm{P})$ be a probability space, $V$ be a vector space, and $X, Y : \Omega \to V$ be random variables. Assume $X$ and $Y$ are independent and symmetric about their respective means $q$ and $k$. Then, $X - Y$ is symmetric about $q - k$.*

**Proof.** For any measurable set $\mathcal{S} \subseteq V$, we denote the translation of $\mathcal{S}$ by $x$ as $\mathcal{S}+x \coloneqq \{s+x \mid s \in \mathcal{S}\}$ and other set operations similarly. Then

$$\begin{aligned}
\Pr(X - Y \in \mathcal{S}) &= \int_Y \Pr(X \in \mathcal{S} + Y)\, d\mathrm{P} \\
&= \int_Y \Pr(X \in 2q - \mathcal{S} - Y)\, d\mathrm{P} \\
&= \int_X \Pr(Y \in 2q - \mathcal{S} - X)\, d\mathrm{P} \\
&= \int_X \Pr(Y \in 2k - 2q + \mathcal{S} + X)\, d\mathrm{P} \\
&= \int_X \Pr(X - Y \in 2(q-k) - \mathcal{S})\, d\mathrm{P} \\
&= \Pr(X - Y \in 2(q-k) - \mathcal{S}).
\end{aligned} \tag{31}$$

**Lemma 4.** *Let $(\Omega, \mathscr{F}, \mathrm{P})$ be a probability space, $L : V \to W$ be a linear operator, and $X : \Omega \to V$ be a random variable. Assume $X$ is symmetric about its mean $q$. Then, $L(X)$ is symmetric about $L(q)$.*

**Proof.** For any measurable set $\mathcal{S} \subseteq W$, we have

$$\begin{aligned}
\Pr(L(X) \in \mathcal{S}) &= \Pr(X \in L^{-1}(\mathcal{S})) \\
&= \Pr(2q - X \in L^{-1}(\mathcal{S})) \\
&= \Pr(L(2q - X) \in \mathcal{S}) \\
&= \Pr(L(X) \in 2L(q) - \mathcal{S}).
\end{aligned} \tag{32}$$

**Lemma 5.** *Let $(\Omega, \mathscr{F}, \mathrm{P})$ be a probability space, $V$ be an inner product space, and $X, Y : \Omega \to V$ be random variables. Assume $X$ and $Y$ are independent and symmetric about their respective means $q$ and $k$. Then, $\Pr(\langle X, Y\rangle \le \langle q, k\rangle) \ge \frac{1}{4}$.*

**Proof.** We define four sets based on reflections about $q$ and $k$:

$$\begin{aligned}
T_0 &\coloneqq \{(x,y) \in V \times V \mid \langle x, y\rangle \le \langle q, k\rangle\} \\
T_1 &\coloneqq \{(x,y) \in V \times V \mid \langle x, 2k - y\rangle \le \langle q, k\rangle\} \\
T_2 &\coloneqq \{(x,y) \in V \times V \mid \langle 2q - x, y\rangle \le \langle q, k\rangle\} \\
T_3 &\coloneqq \{(x,y) \in V \times V \mid \langle 2q - x, 2k - y\rangle \le \langle q, k\rangle\}.
\end{aligned} \tag{33}$$

By symmetry, these four sets have equal probability.

Observe that for any $(x, y) \in V \times V$, we have

$$\langle x, y\rangle + \langle x, 2k - y\rangle + \langle 2q - x, y\rangle + \langle 2q - x, 2k - y\rangle = 4\langle q, k\rangle, \tag{34}$$

which implies at least one of the four pairs must belong to $T_0$.

Therefore, $T_0 \cup T_1 \cup T_2 \cup T_3 = V \times V$. Hence $\Pr((X,Y) \in T_0) + \Pr((X,Y) \in T_1) + \Pr((X,Y) \in T_2) + \Pr((X,Y) \in T_3) \ge 1$ and $\Pr((X,Y) \in T_0) \ge \frac{1}{4}$ follows immediately.

**Proof of Lemma 2.** We express the attention logit difference $L_{i,j_1} - L_{i,j_2}$ in terms of $Q$ and $K$:

$$L_{i,j_1} - L_{i,j_2} = \frac{1}{\sqrt{d_h}} \left\langle q_i, \varphi_{j_1}(k_{j_1}) - \varphi_{j_2}(k_{j_2}) \right\rangle.$$

By Lemma 4, $\varphi_i(q_i), \varphi_{j_1}(k_{j_1}), \varphi_{j_2}(k_{j_2})$ are symmetric about $\varphi_i(q), \varphi_{j_1}(k), \varphi_{j_2}(k)$, respectively. By Lemma 3, the difference $\varphi_{j_1}(k_{j_1}) - \varphi_{j_2}(k_{j_2})$ is symmetric about $\varphi_{j_1}(k) - \varphi_{j_2}(k)$. Finally, by Lemma 5, we have

$$\Pr\left(\langle \varphi_i(q_i), \varphi_{j_1}(k_{j_1}) - \varphi_{j_2}(k_{j_2})\rangle \le \langle \varphi_i(q), \varphi_{j_1}(k) - \varphi_{j_2}(k)\rangle\right) \ge \frac{1}{4}. \tag{35}$$

Hence

$$\Pr\left(L_{i,j_1} - L_{i,j_2} \le L^0_{i,j_1} - L^0_{i,j_2}\right) \ge \frac{1}{4}, \tag{36}$$

which completes the proof.

Table 10: Channel IDs and their wavelengths in Qwen3-8B. These wavelengths serve as the pruning thresholds in FlashWavePrune.

| **Channel ID** | 7 | 15 | 23 | 31 | 39 | 47 | 55 | 63 |
|---|---|---|---|---|---|---|---|---|
| **Wavelength (rounded down)** | 28 | 160 | 900 | 5063 | 28472 | 160114 | 900388 | 5063255 |

# C Kernel Implementation Details

This section details the kernel design of FlashWavePrune (Section 3.3).

## C.1 Forward Kernel

**Preliminaries: MMA atoms and head-dimension groups.** Efficient GEMM implementations tile the query ($M$), key ($N$), and head ($K$) dimensions into small, fixed-size *MMA atoms* (e.g., `M16N8K16`), where each atom consumes contiguous fragments along these three dimensions and executes a FMA (Fused-Multiply-Add) operation on Tensor Cores. Note that an MMA atom is indivisible: masking can only be applied to an entire group, not to individual elements within it. Since the head dimension $d_h$ typically exceeds the atom's $K$ size (typically 16), the kernel splits it into $d_h/16$ *head-dimension groups*, accumulating each group's partial GEMM result to form the final logits.

**Approximating WavePrune for Tensor Core MMA.** Because an MMA atom cannot be split, we propose an approximation for WavePrune to align the dimension-wise pruning thresholds $\{d_k\}$ derived from channel-wise thresholds $\{w_c\}$ with the head-dimension groups. Specifically, for each group $g$, we assign all head dimensions in the group a uniform threshold equal to the group-wise maximum $\max_{k\in\text{group}g}\{d_k\}$ (Figure 2). Under this approximation, we can perform a standard GEMM for the group, and apply the distance predicate $\mathbf{1}_{|i-j|\le d_k}$ directly to the group's GEMM result. Formally, the GEMM result $L^g_{i,j}$ for the $g$-th head-dimension group is added to the logit $L_{i,j}$ iff

$$|i-j| \le d'_g \coloneqq \max_{k\in\text{group } g}\{d_k\} = d_{\text{last dimension in group } g}. \tag{37}$$

In summary, by coalescing per-channel distance predicates into a single predicate per head-dimension group, we preserve the integrity of each MMA atom, enabling efficient execution on Tensor Cores. Empirically, this approximation has minimal effect on model performance (Section 4.5).

**Reducing FLOPs and memory traffic.** FA partitions $Q, K \in \mathbb{R}^{n\times d_h}$ into blocks along the sequence dimension and multiplies each pair of $Q$- and $K$-blocks. Let $I_Q, I_K \subseteq \{1,\dots,n\}$ denote the sequence indices of a $Q$-block and a $K$-block, respectively, and let $D(I_Q, I_K) \coloneqq \{|i-j| \mid i \in I_Q, j \in I_K\}$ denote their pairwise distances. Since block indices are contiguous, $D(I_Q, I_K)$ is also contiguous and forms an integer interval $[l_D, r_D]$. For each head-dimension group $g$, we compare its group-wise threshold $d'_g$ (Equation 37) with $l_D$ and $r_D$ to classify it into one of the three categories:

- *Fully masked groups:* $d'_g < l_D$. The entire head-dimension group is masked.
- *Fully retained groups:* $d'_g \ge r_D$. The entire head-dimension group is retained.
- *Critical groups:* $l_D \le d'_g < r_D$. The head-dimension group is partially masked.

Since $d_k$ increases monotonically with $k$, head-dimension groups are partitioned by two split points: `splitL`, the first critical group, and `splitR`, the first fully retained group.

Since groups before `splitL` are fully masked, we skip all their QK multiplications and, during decode, bypass their key-cache loads. This reduces both FLOPs and memory traffic. For groups at or after `splitR`, we use FA2's standard FMA path; only the narrow critical region [`splitL`, `splitR`) requires group-wise GEMM and masking. Figure 10 summarizes this design.

**Improving ILP via compile-time specialization.** If `splitL` and `splitR` are purely treated as runtime variables, the compiler has to preserve the data-dependent branches and cannot fully unroll the loop over head-dimension groups. This reduces Instruction-Level Parallelism (ILP) and prevents the efficient scheduling of shared-memory loads and tensor-core MMA instructions. To resolve this, we enumerate (`splitL`, `splitR`) pairs and pass them as template arguments, elevating these split

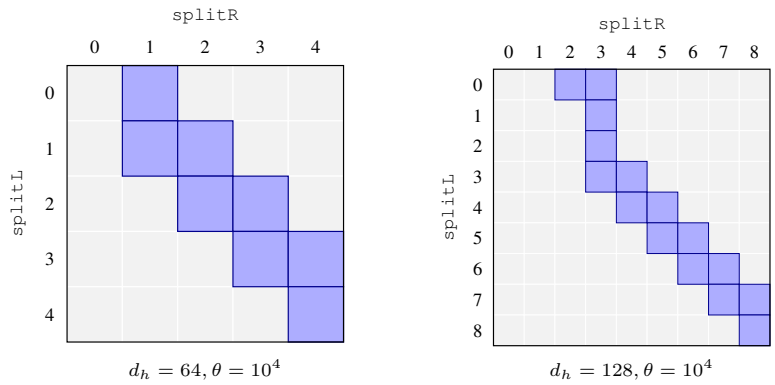



Figure 13: Two example grids of (`splitL, splitR`) branches. Blue squares denote reachable branches; gray squares denote unreachable ones. Note that the reachable set depends on $\{d'_g\}$; we set the model's RoPE base $\theta$ to $10^4$ here for illustration.

points to compile-time constants. Consequently, the retained suffix is compiled into a highly-optimized static GEMM, and the masking overhead is strictly confined to the boundary region.

**Controlling branching and code size.** Naively specializing each (`splitL, splitR`) branch creates separate template instances that scale quadratically with the number of head-dimension groups (e.g., 8 for $d_h = 128$ and 16 for $d_h = 256$). This bloats the SASS code size and exhausts the compiler's inlining budget, forcing helper functions into out-of-line device calls (`CALL.FUNC`). We also observe that this hinders the compiler's ability to conduct interprocedural optimizations across the entire kernel. To avoid these issues, we specialize only the *reachable* (`splitL, splitR`) branches for a given model, which are far fewer than the full Cartesian product (Figure 13). Although this necessitates recompiling the kernel for each distinct model, it is an acceptable tradeoff, as the inlining of hot paths and the simplified control flow outweigh the one-time compilation cost.

### C.2 Backward Kernel

**Asymmetry with the forward kernel.** We first name the computation performed by the forward kernel. In the standard MMA notation, the $M$ and $N$ axes index the rows and columns of the output matrix, and the $K$ axis is the contraction axis that is summed over to produce the output. For the forward attention computation, these axes correspond to the query dimension ($i$), the key dimension ($j$), and the head dimension ($k$), respectively. In WavePrune, The head dimension is grouped into 16-dimensional head-dimension groups, where each head-dimension group is assigned a pruning threshold $d'_g$ (Equation 37).

Let $Q \in \mathbb{R}^{m\times k}, K \in \mathbb{R}^{n\times k}$ denote the RoPE-encoded query and key matrices. The core computation of the forward kernel is $P \coloneqq \mathrm{WMM}_P(Q, K)$ (WMM denotes WavePrune matrix multiplication):

$$P = [\mathrm{WMM}_P(X, Y)]_{i,j} \coloneqq \sum_{k=1}^{d_h} X_{i,k} Y_{j,k} \cdot \mathbf{1}_{|i-j|\le d'_{g(k)}}. \tag{38}$$

Crucially, the backward pass has a different axis arrangement. Given $\mathrm{d}P$ and $K$, $\mathrm{d}Q$ is computed as

$$\mathrm{d}Q = [\mathrm{WMM}_Q(\mathrm{d}P, K)]_{i,k} \coloneqq \sum_{j=1}^{n} \mathrm{d}P_{i,j} K_{j,k} \cdot \mathbf{1}_{|i-j|\le d'_{g(k)}},$$

where the $M, N, K$ axes correspond to the query dimension ($i$), the head dimension ($k$), and the key dimension ($j$), respectively. The computation of $\mathrm{d}K$ is similar:

$$\mathrm{d}K = [\mathrm{WMM}_K(\mathrm{d}P, Q)]_{j,k} \coloneqq \sum_{i=1}^{m} \mathrm{d}P_{i,j} Q_{i,k} \cdot \mathbf{1}_{|i-j|\le d'_{g(k)}},$$

where the $M, N, K$ axes correspond to the key dimension ($j$), the head dimension ($k$), and the query dimension ($i$), respectively.

Table 11: Ablation of WavePrune kernel optimizations. Each entry reports the runtime ratio relative to the final optimized kernel at sequence length 32768; larger values indicate slower execution. Since our prefill implementation does not use suffix-only K loading, these entries are marked as N/A.

| Removed Optimizations | Prefill H64 ↓ | Prefill H128 ↓ | Decode H64 ↓ | Decode H128 ↓ |
|---|---|---|---|---|
| w/o splits specialization | 1.26× | 1.47× | 1.09× | 1.04× |
| w/o splitR specialization | 1.25× | 1.54× | 1.04× | 1.08× |
| w/o suffix-only K load | N/A | N/A | 0.99× | 1.14× |
| w/o suffix-only QK GEMM | 1.38× | 1.46× | 1.01× | 1.08× |

Unlike forward computation ($\mathrm{WMM}_P$), the head dimension for $\mathrm{WMM}_K$ and $\mathrm{WMM}_Q$ is the $N$ axis rather than the shared $K$ axis.

**Implementation overview.** We modify only the $\mathrm{d}Q$ and $\mathrm{d}K$ computations as $\mathrm{d}V$ and $\mathrm{d}P$ follow FA2's standard path. Despite the axis mismatch above, both gradients can still be computed using the same idea as the forward kernel. Under the Ampere MMA atom `M16N8K16`, an atom spans $8$ elements along the $N$ axis (head dimension), which divides the group size of 16. No atom therefore straddles two head-dimension groups, so the pruning threshold $d'_g$ can be applied at the atom level, and the three-way split of the forward kernel carries over (Figure 10). The primary difference is that we must mask the operand $\mathrm{d}P$ *before* the MMA rather than the accumulator afterwards.

**Joint $\mathrm{d}\boldsymbol{Q}$ and $\mathrm{d}\boldsymbol{K}$ computation.** For a given head-dimension group $g$, the two gradients $\mathrm{d}Q$ and $\mathrm{d}K$ apply the same predicate $\mathbf{1}_{|i-j|\le d'_g}$ on $\mathrm{d}P$, so we can mask $\mathrm{d}P$ once and use the result twice by jointly computing $\mathbf{d}\boldsymbol{Q}$ and $\mathbf{d}\boldsymbol{K}$. Specifically, we enumerate all critical groups (Section 3.3) *at runtime*, load a short-lived $\mathrm{d}P$ fragment from shared memory, applies the predicate on the fragment, and then issues MMAs for both gradients. This joint computation halves the masking work and addresses the ALU bottleneck. For the fully retained suffix $[\texttt{splitR}, d_h/16)$, we directly consume the unmasked tiles and template-specialize `splitR` in the same way as the forward kernel.

**Why per-pair specialization fails.** Surprisingly, the (`splitL`, `splitR`) compile-time specialization that accelerates the forward kernel is counterproductive here. The backward kernel body is massive: it contains the score recomputation, the $\mathrm{d}V$ and $\mathrm{d}P$ GEMM, and the joint $\mathrm{d}Q, \mathrm{d}K$ GEMM. Cloning the joint GEMM per (`splitL`, `splitR`) pair replicates a far larger body, which prevent the compiler from generating efficient machine code. The cost does not grow gradually but appears as a discrete cliff: with $m_{\text{block}} = 64, n_{\text{block}} = 128, d_h = 128$, specializing even a single (`splitL`, `splitR`) pair grows the per-thread stack frame from 120 to 512 bytes, the static local load/store counts from 34/38 to 275/239, and the kernel from 89,856 to 122,752 bytes of SASS. Once `ptxas` crosses its inlining and allocation budgets, it holds program state in local memory and moves code out of line, adding control overhead that outweighs the pruning savings.

**Results.** Table 12 reports backward kernel latencies under the same setup as Section 4.4. The overhead of WavePrune backward kernel is $8.58\%$ at 8K tokens, which narrows to $3.81\%$ at 16K and $0.18\%$ at 32K. Comparing with standard FA2, the overall training-time gap is negligible.

Table 12: Backward kernel latencies. Overhead refers to the latency increase of WavePrune over FA2.

| Length | FA2 (ms) | WavePrune (ms) | Overhead (%) |
|---|---|---|---|
| 8K | 93.30 | 101.31 | 8.58% |
| 16K | 363.24 | 377.08 | 3.81% |
| 32K | 1434.34 | 1436.88 | 0.18% |

## D Training-Free Integration

**HELMET benchmark.** HELMET (Yen et al., 2025) is a comprehensive benchmark comprising diverse long-context tasks and extending to sequence lengths as long as 128K. Specifically, it includes the following *tasks*, each of which consists of multiple *subtasks*:

- **Recall:** Retrieving hidden information from a large context (inherited from RULER (Hsieh et al., 2024) and JSON KV (Liu et al., 2024)).
- **RAG:** Answering open-domain questions by retrieving relevant information from a provided noisy corpus.
- **Re-rank:** Ranking a set of passages based on their relevance to a given query.
- **Cite:** Generating long-form answers to multi-faceted questions while providing correct inline passage citations.
- **LongQA:** Answering questions based on long documents like books or movie scripts.
- **ICL:** Solving new classification tasks using thousands of in-context examples.
- **Summ:** Synthesizing and summarizing information from extensive contexts.

Due to the substantial cost of LLM-as-a-Judge (which requires proprietary models such as GPT-4o in the original HELMET evaluation), we switch to ROUGE-1 F1-score for NarrativeQA evaluation (a subtask of LongQA). In Summ, where the ground-truth and model-generated summaries are often more than 1,000 tokens long, we find F1 score fails to capture the quality of summaries, so we drop Summ from our evaluation and report only the remaining six tasks. For all subtasks where LLM-as-a-Judge is not required, we use the same evaluation protocol as HELMET.

During evaluation, we find the maximum output tokens are insufficient for RULER MV and JSON KV (subtasks of Recall), which leads to extremely unstable results due to early truncation of model responses. Therefore, we increase the maximum output tokens of RULER MV and JSON KV to 200 and 300 respectively.

**General benchmarks.** To ensure a comprehensive evaluation, we select benchmarks from four core capabilities:

- **Language modeling**: MMLU (Hendrycks et al., 2021a), CMMLU (Li et al., 2023)
- **General reasoning**: BBH (Suzgun et al., 2022), DROP (Dua et al., 2019)
- **Code**: MBPP (Austin et al., 2021), HumanEval (Chen et al., 2021)
- **Math**: GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021b)

All evaluations are conducted using the `lm-evaluation-harness` framework (Gao et al., 2024) under default configurations.

**Evaluated LLMs.** Table 13 summarizes the architectural properties of the LLMs evaluated in Section 4.1. Following Yang et al. (2025), we use YaRN (Peng et al., 2024) for Qwen3-8B and Qwen3-32B at context lengths above 32K, and native RoPE otherwise. To reduce evaluation cost, we disable the thinking mode of Qwen3-8B and Qwen3-32B.

| Model | Context length | RoPE $\theta$ | RoPE scaling | Attention type | Release date |
|---|---|---|---|---|---|
| Qwen3-8B | 32K (128K w/ YaRN) | $10^6$ | None (YaRN optional) | Standard full attention | Apr 2025 |
| Qwen3-32B | 32K (128K w/ YaRN) | $10^6$ | None (YaRN optional) | Standard full attention | Apr 2025 |
| Gemma3-12B | 128K | $10^4$ / $10^6$ | Linear ($\times 8$) | Sliding window + full attention | Mar 2025 |
| Ministral3-3B | 256K | $10^6$ | YaRN ($\times 16$) | Standard full attention | Dec 2025 |
| Llama3.1-8B | 128K | $5 \times 10^5$ | Llama 3 ($\times 8$) | Standard full attention | Jul 2024 |

Table 13: Properties of LLMs used in long-context evaluation.

## E Failed Attempts on Llama3.1-8B

**Diagnosis: high-frequency channels carry long-range retrieval signals.** Because Llama3.1-8B's degradation (Table 1) is concentrated on retrieval tasks, we inspect a 32K-context NIAH example in which the needle is placed near the middle of the context. In the original model, layer 16, head 21 attends strongly to the needle, which is driven by an RoPE channel whose wavelength is approximately 571. Surprisingly, this channel stably transmits the needle's information to the final tokens despite a

query–key distance far exceeding its wavelength. This behavior contradicts the conventional view that high-frequency channels encode local positional information (Barbero et al., 2025). WavePrune truncates exactly this signal, causing the model to fail at retrieving the answer. Similar patterns are observed in other heads of Llama3.1-8B, sometimes mediated by different RoPE channels.

**Failure attempts.** We document the following failed remedies; none of them recovers the performance of the unpruned baseline.

- Increasing the pruning coefficient $\alpha$ (Section 3.1) to postpone truncation.
- Truncating only the first few highest-frequency channels.
- Applying WavePrune only to the first or last few layers.
- Continuing long-context training with WavePrune under the ProLong recipe (Gao et al., 2025), on either the base or the instruct models.

# F Controlled Experiment Details

## F.1 Tetranacci Data Construction

**Data Construction.** In this task, each sample encodes a Tetranacci sequence, which is defined by $a_n \coloneqq (a_{n-1}+a_{n-2}+a_{n-3}+a_{n-4}) \bmod 1000$. The initial values $a_0, a_1, a_2, a_3$ are drawn uniformly at random from $\{0, \ldots, 999\}$. Values are written in decimal, tokenized with the GPT-2 tokenizer (Radford et al., 2019), and each is followed by an EOT token that separates consecutive values. The four initial values, the EOT tokens between them, and all padding are excluded from the training loss; every subsequent value and EOT token are supervised. Since the recurrence is deterministic given the four most recent values, a sequence becomes periodic as soon as any block of four consecutive values repeats; we discard and regenerate such samples to guarantee that no training or validation sequence is periodic. The dataset comprises $500{,}000$ samples, split $99{:}1$ into training and validation sets, with sequence lengths of $2{,}048$ and $4{,}096$ tokens, respectively.

**Example.** With initial values $a_0 = 123$, $a_1 = 456$, $a_2 = 789$, and $a_3 = 12$, the next values are $a_4 = (123+456+789+12) \bmod 1000 = 380$ and $a_5 = (456+789+12+380) \bmod 1000 = 637$. The resulting token sequence is

$$\texttt{123 <eot> 456 <eot> 789 <eot> 12 <eot>} \underbrace{\texttt{380 <eot> 637 <eot> } \cdots}_{\text{supervised}},$$

where $\langle\texttt{eot}\rangle$ denotes the EOT token.

## F.2 NIAH Data Construction

**Data Construction.** In this task, each sample hides a "magic number" (a seven-digit integer drawn uniformly at random from $[10^6, 10^7 - 1]$) inside a haystack of irrelevant text sampled from OpenWebText. A sample concatenates (i) a fixed instruction stating that a magic number exists, (ii) a segment of background tokens containing the sentence "The special magic number is $\langle$number$\rangle$" inserted at a uniformly random position, (iii) the fixed instruction "The special magic number mentioned in the text above is:", and (iv) the target number followed by a final EOT token. Only the target number and the final EOT token contribute to the training loss. The dataset comprises $500{,}000$ samples, split $99{:}1$ into training and validation sets, with sequence lengths of $2{,}048$ and $4{,}096$ tokens, respectively.

**Example.** With the magic number $5{,}432{,}167$, a sample has the layout

```
[instruction] [background tokens]
\n The special magic number is 5432167.\n [background tokens]
\n The special magic number mentioned in the text above is: 5432167 <eot>.
```

(`5432167 <eot>` supervised)

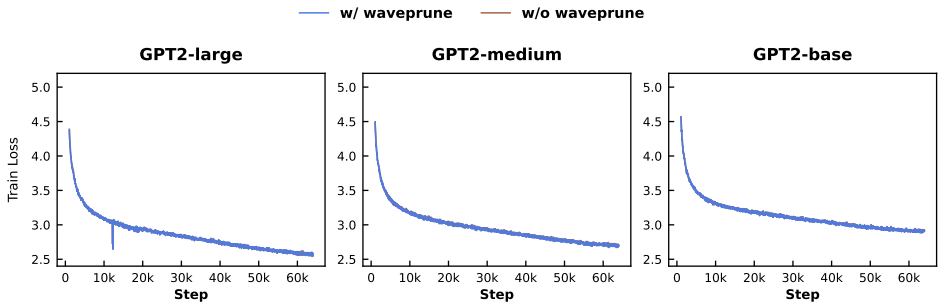



Figure 14: Training loss of GPT2-large (left), GPT2-medium (middle), and GPT2-base (right).

## G PRETRAINING HYPERPARAMETERS

| **Hyperparameter** | **GPT2-base** | **GPT2-medium** | **GPT2-large** |
|---|---|---|---|
| Total Params | 124M | 350M | 774M |
| Layers | 12 | 24 | 36 |
| Heads | 12 | 16 | 20 |
| Dimension | 768 | 1024 | 1280 |
| Head Dimension | 64 | 64 | 64 |
| Attention Module | Multi-Head Attention (Vaswani et al., 2017) | | |
| Positional Encoding | RoPE (Su et al., 2024) | | |
| RoPE $\theta$ | 10,000 | | |
| Dataset | OpenWebText | | |
| Vocab Size | 50,304 | | |
| Train Sequence Length | 2,048 | | |
| Val Sequence Length | 4,096 | | |
| Micro Batch Size | 3 | | |
| Gradient Accum. Steps | 40 | | |
| Effective Batch Size (seqs) | 120 | | |
| Training Steps | 64,000 | | |
| Total Train Tokens | $\sim$15.7B | | |
| Optimizer | AdamW (Loshchilov & Hutter, 2019) | | |
| Adam $(\beta_1, \beta_2)$ | $(0.9, 0.95)$ | | |
| Base Learning Rate | $6 \times 10^{-4}$ | | |
| Min Learning Rate | $6 \times 10^{-5}$ | | |
| LR Scheduler | Cosine decay w/ linear warmup | | |
| Warmup Steps | 2,000 | | |
| Weight Decay | 0.1 | | |
| Gradient Clipping | 1.0 | | |
| Dropout | 0.0 | | |
| Precision | `bfloat16` | | |

Table 14: Training hyperparameters for the GPT-2 series pretraining experiments.

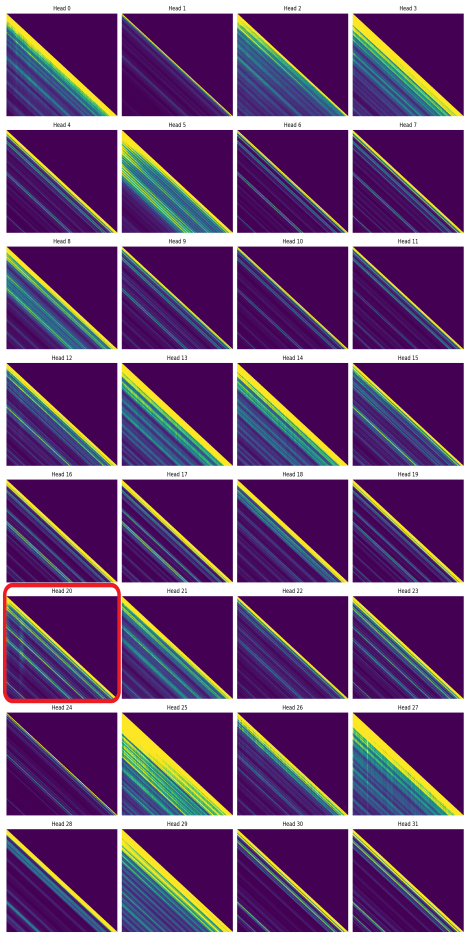
# H ATTENTION VISUALIZATIONS

## H.1 EXPERIMENTAL DETAILS

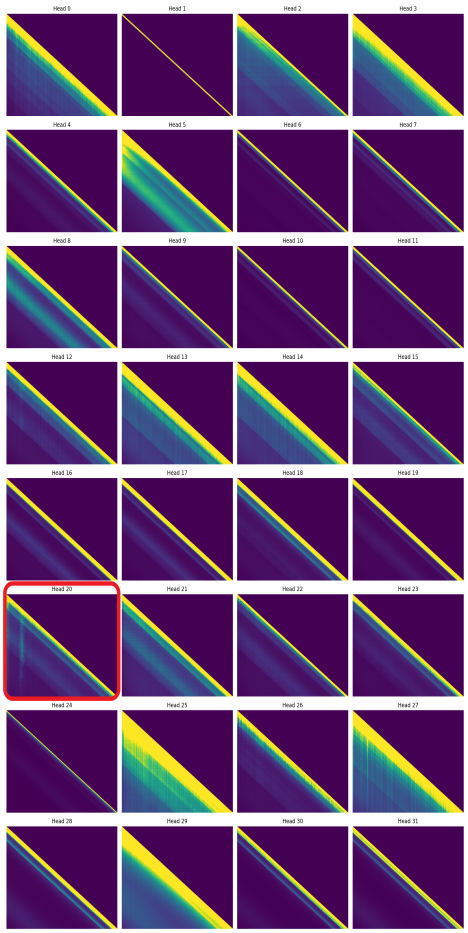
We produce the attention maps by running Qwen3-8B and Ministral3-3B on 100 sequences sampled from PG19. Each sequence is truncated to 4096 tokens. For each model, we extract the attention scores and average these maps across the 100 sequences.

## H.2 LAYER-WIDE AND HEAD-LEVEL COMPARISONS

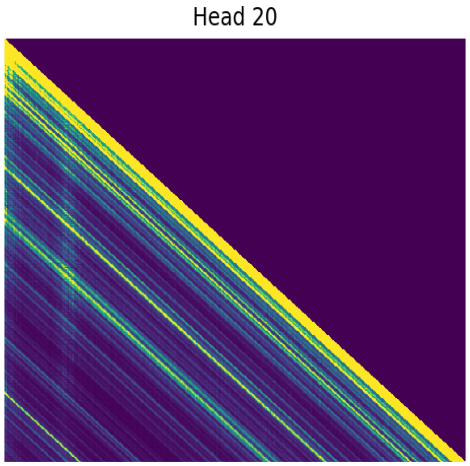
Our visualization of attention maps includes four layer-wide attention score and two head-level per-channel logit comparisons. The example in the main text is Qwen3-8B layer 0, head 20 (Figure 15); additional layer-wise visualizations below use Qwen3-8B layer 32 head 12 (Figure 17), Ministral3-3B layer 0 head 23 (Figure 16), and Ministral3-3B layer 1 head 17 (Figure 18). In each visualization, "original" denotes the unmodified attention computation and "after WavePrune" denotes attention with each channel truncated at its wavelength. The full-map panels show layer-wide scores, while the cropped panels show the indicated head in greater detail. The per-channel figures show centered logit contributions for Qwen3-8B layer 0, heads 20 (Figure 19 vs. Figure 20) and 6 (Figure 21 vs. Figure 22), each before and after applying WavePrune.



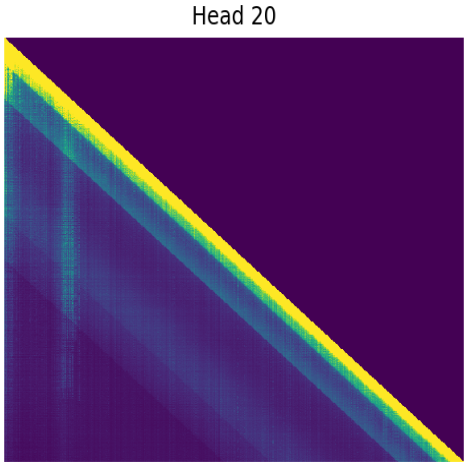
(a) Attention scores (original)



(b) Attention scores (after WavePrune)

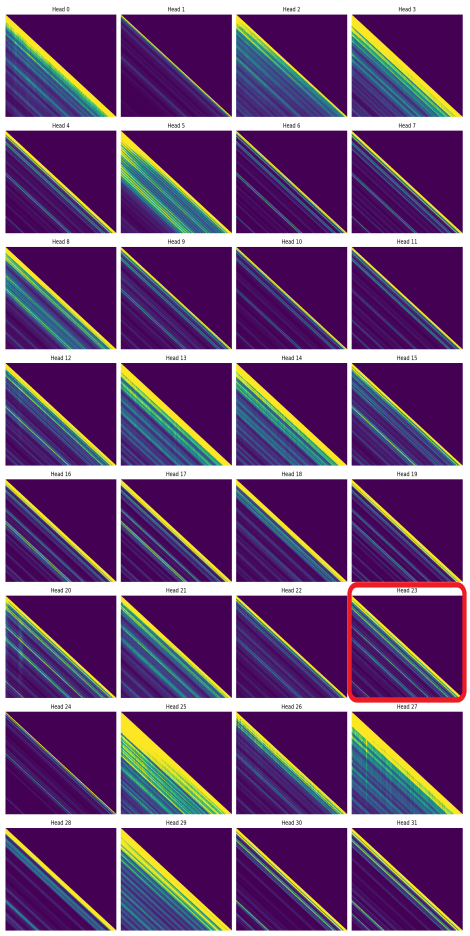



(c) Attention scores (original, head 20)

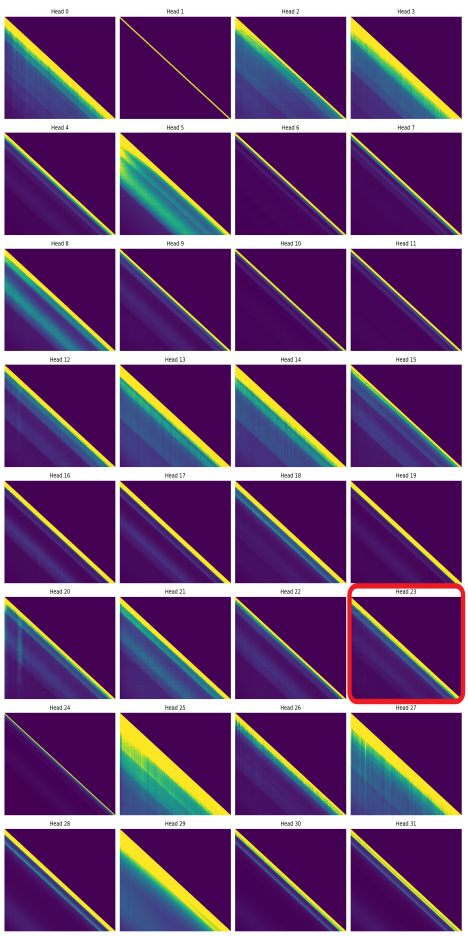



(d) Attention scores (after WavePrune, head 20)

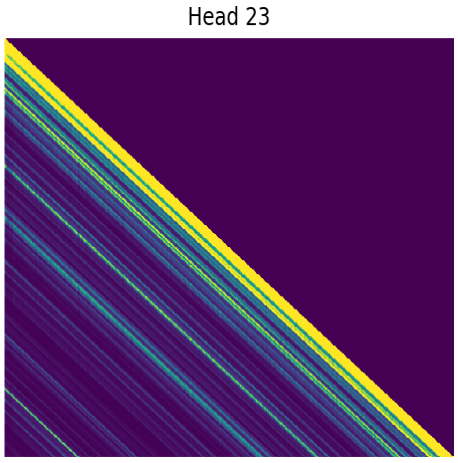
Figure 15: Attention scores in Qwen3-8B's layer 0. (a-b) Layer-wide attention scores. In the original model (a), attention is scattered across multiple ghost sub-diagonals. WavePrune (b) effectively eliminates these artifacts, allowing attention to concentrate on the intended lags. (c-d) We use head 20 as an example to illustrate the elimination of ghost sub-diagonals after applying WavePrune.



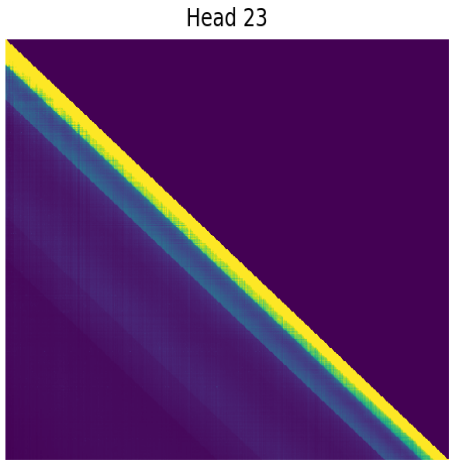
(a) Attention scores (original)



(b) Attention scores (after WavePrune)

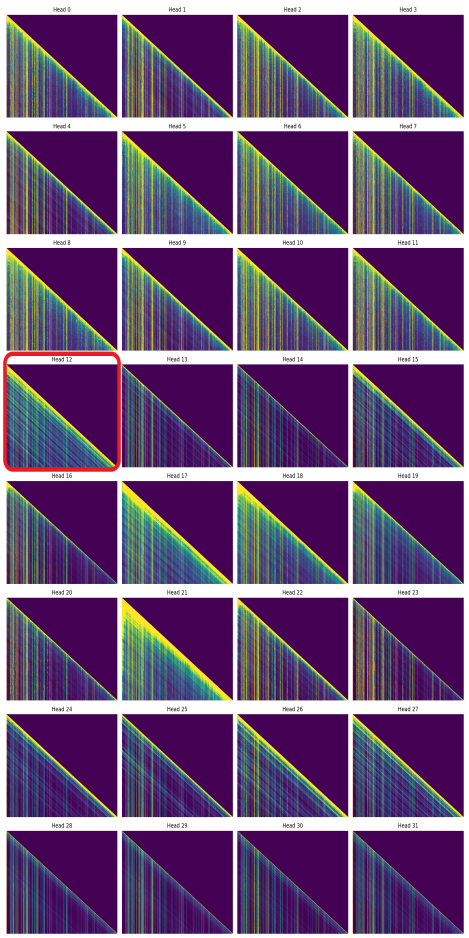



(c) Attention scores (original, head 23)

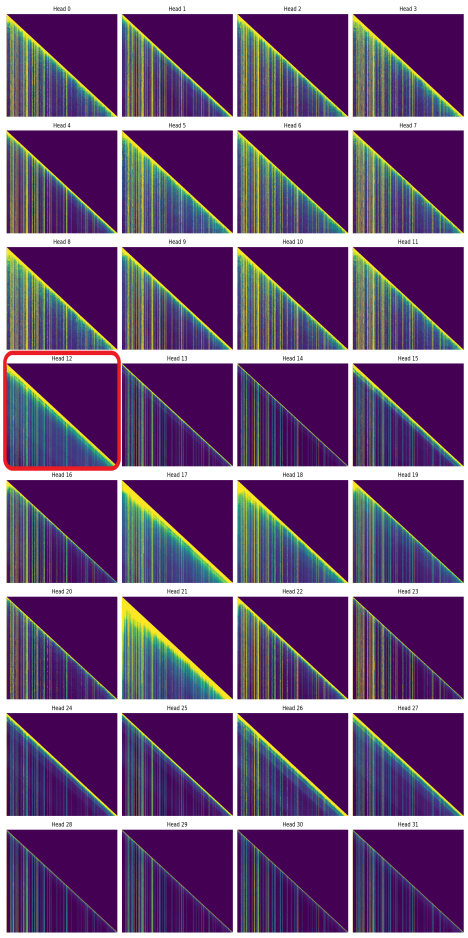



(d) Attention scores (after WavePrune, head 23)

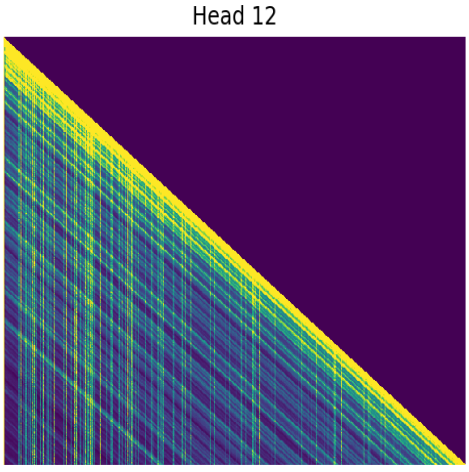
Figure 16: Attention scores in Ministral3-3B's layer 0. (a-b) Layer-wide attention scores. In the original model (a), attention is scattered across multiple ghost sub-diagonals. WavePrune (b) effectively eliminates these artifacts, allowing attention to concentrate on the intended lags. (c-d) We use head 23 as an example to illustrate the elimination of ghost sub-diagonals after applying WavePrune.



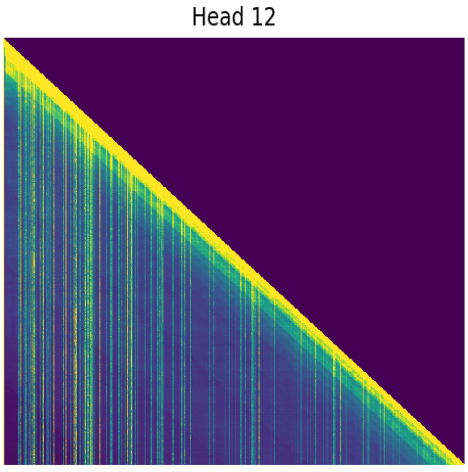
(a) Attention scores (original)



(b) Attention scores (after WavePrune)

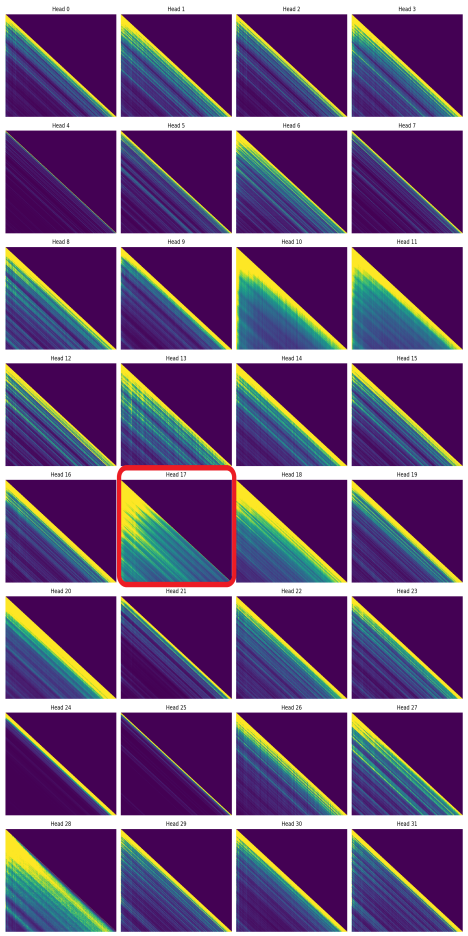



(c) Attention scores (original, head 12)

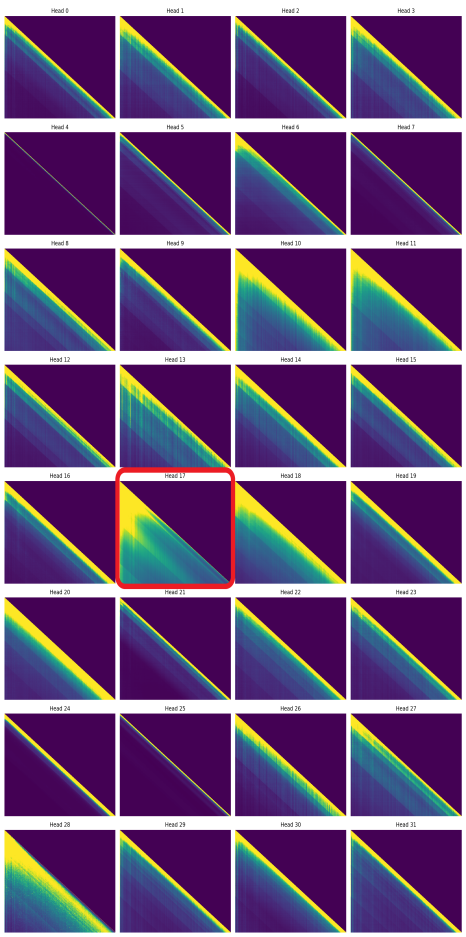



(d) Attention scores (after WavePrune, head 12)

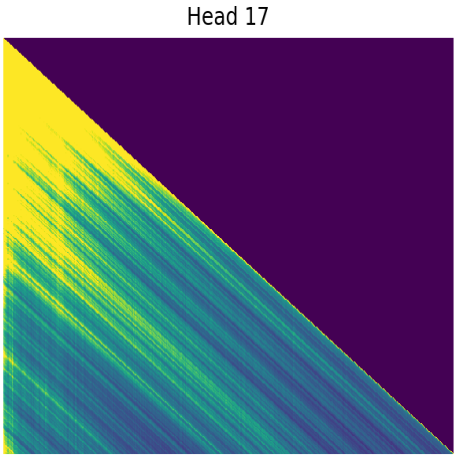
Figure 17: Attention scores in Qwen3-8B's layer 32. (a-b) Layer-wide attention scores. In the original model (a), attention is scattered across multiple ghost sub-diagonals. WavePrune (b) effectively eliminates these artifacts, allowing attention to concentrate on the intended lags. (c-d) We use head 12 as an example to illustrate the elimination of ghost sub-diagonals after applying WavePrune.



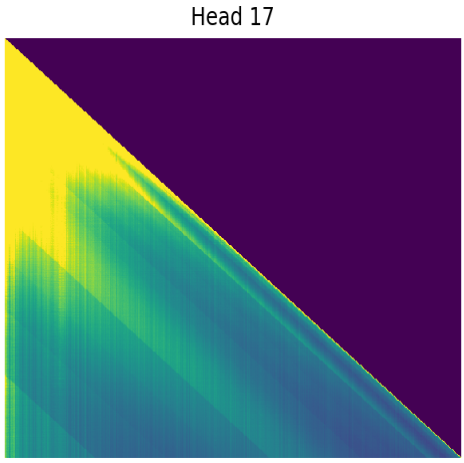
(a) Attention scores (original)



(b) Attention scores (after WavePrune)




(c) Attention scores (original, head 17)




(d) Attention scores (after WavePrune, head 17)

Figure 18: Attention scores in Ministral3-3B's layer 1. (a-b) Layer-wide attention scores. In the original model (a), attention is scattered across multiple ghost sub-diagonals. WavePrune (b) effectively eliminates these artifacts, allowing attention to concentrate on the intended lags. (c-d) We use head 17 as an example to illustrate the elimination of ghost sub-diagonals after applying WavePrune.

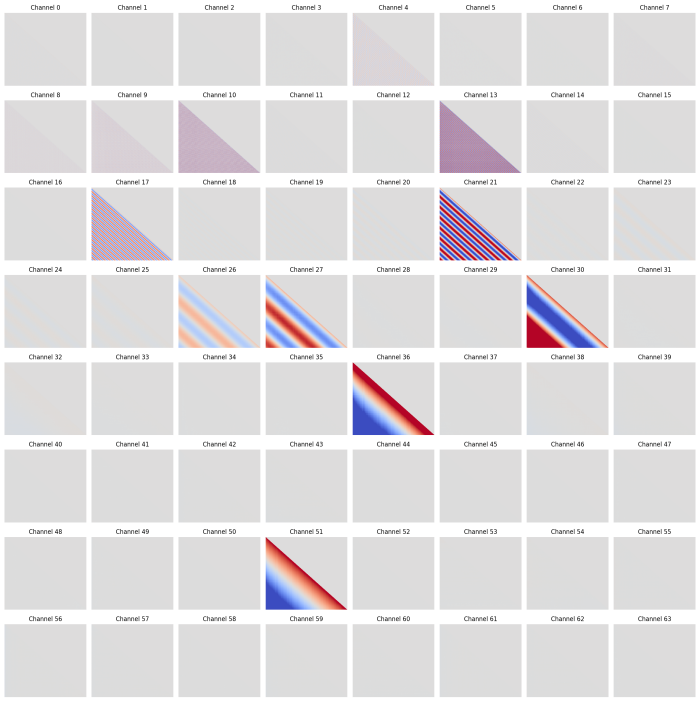


Figure 19: Channel-wise decomposition of attention logits for Qwen3-8B, layer 0 head 20 (centered by subtracting the mean logit value).

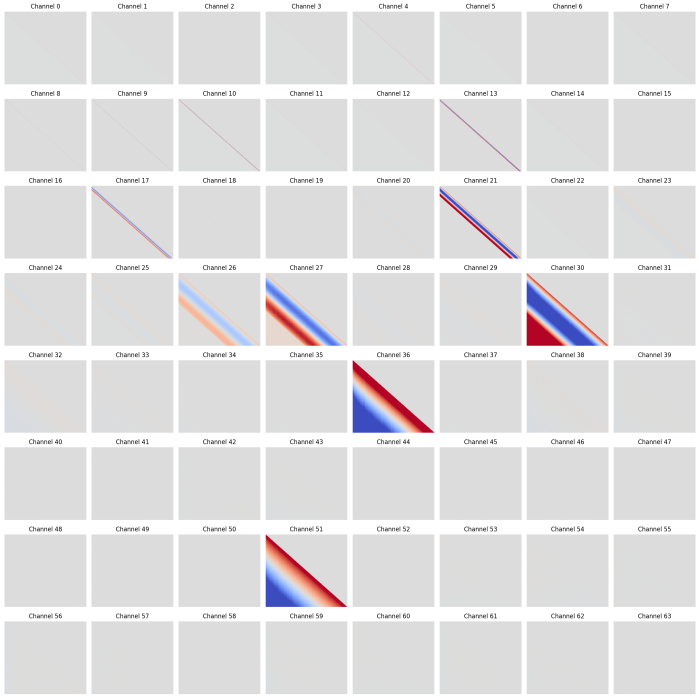


Figure 20: Channel-wise decomposition of attention logits for Qwen3-8B, layer 0 head 20 after applying WavePrune (centered by subtracting the mean logit value).

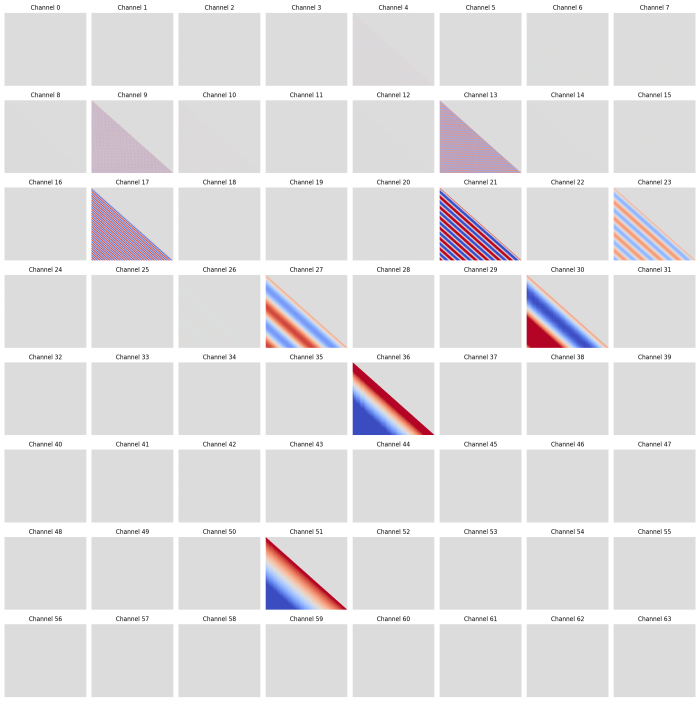



Figure 21: Channel-wise decomposition of attention logits for Qwen3-8B, layer 0 head 6 (centered by subtracting the mean logit value).

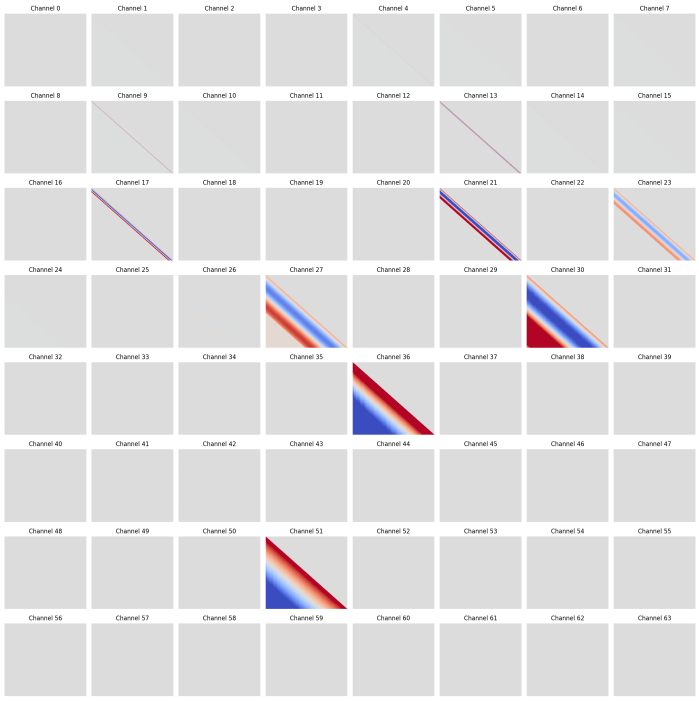



Figure 22: Channel-wise decomposition of attention logits for Qwen3-8B, layer 0 head 6 after applying WavePrune (centered by subtracting the mean logit value).