River-LLM: Large Language Model Seamless Exit Based on KV Share

Yingtao Shen
Shanghai Jiao Tong University
Shanghai, China
doctorcoal@sjtu.edu.cn

An Zou
Shanghai Jiao Tong University
Shanghai, China
an.zou@sjtu.edu.cn

Abstract

Large Language Models (LLMs) have demonstrated exceptional performance across diverse domains but are increasingly constrained by high inference latency. Early Exit has emerged as a promising solution to accelerate inference by dynamically bypassing redundant layers. However, in decoder-only architectures, the efficiency of Early Exit is severely bottlenecked by the KV Cache Absence problem, where skipped layers fail to provide the necessary historical states for subsequent tokens. Existing solutions, such as recomputation or masking, either introduce significant latency overhead or incur severe precision loss, failing to bridge the gap between theoretical layer reduction and practical wall-clock speedup. In this paper, we propose River-LLM, a training-free framework that enables seamless token-level Early Exit. River-LLM introduces a lightweight KV-Shared Exit River that allows the backbone's missing KV cache to be naturally generated and preserved during the exit process, eliminating the need for costly recovery operations. Furthermore, we utilize state transition similarity within decoder blocks to predict cumulative KV errors and guide precise exit decisions. Extensive experiments on mathematical reasoning and code generation tasks demonstrate that River-LLM achieves 1.71× to 2.16× practical speedup while maintaining high generation quality.

1 Introduction

In recent years, Large Language Models (LLMs), particularly decoder-only language models, have driven transformative advancements globally, demonstrating expert-level performance in domains such as code generation (Rozière et al., 2023), complex reasoning (Wei et al., 2022), and creative writing (Bubeck et al., 2023). Despite their remarkable capabilities, the enormous parameter counts of LLMs impose significant challenges, including substantial inference latency and excessive hardware energy consumption (Pope et al., 2023).

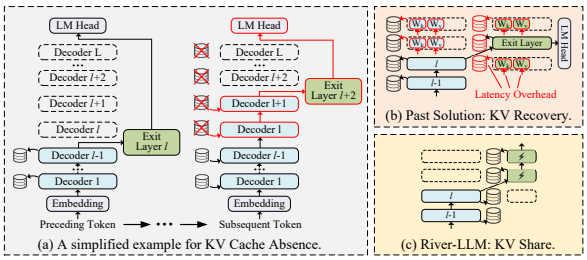


Figure 1: KV Cache Absence problem for Early Exit in decoder-only LLM.

To mitigate these costs, dynamic inference (Han et al., 2021) has emerged as a promising paradigm for optimizing token generation. Among various techniques, Early Exit, originally proven effective in classical CNNs (Teerapittayanon et al., 2016; Panda et al., 2016), is gaining traction as a dominant trend for LLM acceleration. By dynamically bypassing redundant layers based on input complexity, Early Exit enables sample-dependent computation reduction. Recent advancements encompass sophisticated skipping strategies (Del Corro et al., 2023; Luo et al., 2025), specialized exit layer designs (Chen et al., 2024; Jamialahmadi et al., 2025), and optimizations for specific reasoning modes such as Chain-of-Thought (Yang et al., 2025) or speculative drafting (Elhoushi et al., 2024).

However, a critical gap exists between the theoretical potential and practical efficiency of Early Exit in LLMs. Our empirical analysis reveals that while over 50% of tokens can theoretically exit at early layers without compromising output consistency, the actual wall-clock speedup during autoregressive inference remains marginal. This discrepancy is primarily attributed to the **KV Cache Absence** problem. As illustrated in Fig. 1, when a token exits early, it fails to compute the Key-Value (KV) pairs for the bypassed layers. Con-

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 37633–37645
July 2-7, 2026 ©2026 Association for Computational Linguistics

sequently, subsequent tokens lack the necessary history for their own self-attention computations at those layers. Existing remedies, such as Batching Recompute (Chen et al., 2024), State Propagation (Schuster et al., 2022), or KV Masking (Jiang et al., 2024), either introduce significant latency overhead or suffer from severe accuracy degradation. None of these approaches fully resolve the conflict between token-level exiting and KV cache integrity.

Inspired by recent explorations in KV cache redundancy (Liu et al., 2024), we propose **River-LLM**, a framework that achieves **seamless token-level exit** by constructing a "KV-Shared Exit River". River-LLM eliminates the need for costly recovery operations by ensuring that the skipped backbone layers' KV information is naturally preserved or approximated through a lightweight, shared structure. Our primary contributions are summarized as follows:

- We provide a systematic evaluation of existing methods for addressing KV Cache Absence. By establishing a unified testing environment, we demonstrate that current strategies varying from masking to recomputation fail to bridge the gap between theoretical layer reduction and practical latency savings.
- We introduce the first "seamless exit" mechanism for LLMs. By designing a lightweight KV-Shared Exit Layer, River-LLM allows the backbone's missing KV cache to be implicitly generated during the exit process. Furthermore, we utilize the similarity between decoder inputs and outputs to predict cumulative KV errors, guiding precise exit decisions.
- Experimental results across diverse long-sequence tasks, including mathematical reasoning and code generation, demonstrate that River-LLM achieves $1.71\times$ to $2.16\times$ wall-clock speedup without requiring any additional training or fine-tuning.

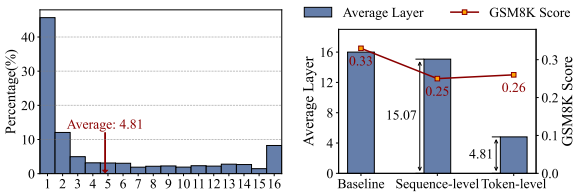
2 Related Works

Early Exit is a cornerstone of dynamic neural networks designed to reduce computational redundancy (Han et al., 2021). Initially, the Early Exit methods for Large Language Models (LLMs) was inherited from the previous Early Exit work on other neural network architectures. For instance, DAT (Elbayad et al., 2020) introduced a variable-layer Transformer reminiscent of the BranchyNet architecture (Teerapittayanon et al., 2016; Panda et al., 2016). Subsequent works like CALM (Schuster et al., 2022) explored confidence metrics tailored for LLMs, while AdaInfer (Fan et al., 2025) is similar to (Li et al., 2023), it trains an SVM to predict at which layer the LLM can achieve results consistent with those of the last layer.

As decoder-only architectures became dominant, research shifted toward leveraging the unique structural and inferencing properties of these models. One primary direction focuses on skipping strategies. SkipDecode (Del Corro et al., 2023) bypasses intermediate decoders based on a pre-defined computational budget, while DiffSkip (Luo et al., 2025) and AdaSkip (He et al., 2025b) utilize similarity between attention vectors to make on-the-fly skipping decisions. Notably, D-LLM (Jiang et al., 2024) incorporates MLPs to control layer skipping and introduces a mask mechanism to maintain KV Cache consistency. RAEE (Huang et al., 2024) further enhances control by retrieving exit distributions from similar tasks. Parallelly, another research thread focuses on exit layer design to mitigate accuracy loss. Balcony (Jamialahmadi et al., 2025) transfers the final layer of the backbone as fine-tuned exit layers, whereas EE-LLM (Chen et al., 2024) provides a diverse library of exit modules (MLP, Decoder, etc.) and implements parallel GPU scheduling to minimize overhead.

Beyond general autoregressive generation, Early Exit has recently been integrated into specialized applications. LayerSkip (Elhoushi et al., 2024) and SpecEE (Xu et al., 2025) combine Early Exit with Speculative Decoding to accelerate the drafting stage, pushing the Pareto frontier of inference efficiency. Furthermore, DEER (Yang et al., 2025) adapts the Early Exit concept to Chain-of-Thought (CoT) reasoning by truncating the reasoning chain based on confidence.

In this work, we distinguish Early Exit from other dynamic inference paradigms such as Router-Tuning (He et al., 2025a) and MoNE (Jain et al., 2024). While those methods treat dynamic routing as an intrinsic architectural property, Early Exit focuses on losslessly or near-losslessly bypassing computations within a fixed backbone. Consequently, we categorize "skip-based" and "dynamic depth" approaches under the broader umbrella of Early Exit.



(a) (b)

Figure 2: (a) Distribution of optimal Token-level Exit position for Llama3.2 1B on GSM8K. $Score = 0.26$. (b) Token-level Exit significantly outperforms Sequence-level Exit on GSM8K.

3 Motivation

In this section, we first investigate the full potential of the Early Exit technique in Large Language Models. A series of experiments conducted on the Llama3.2 1B model with pre-trained decoder-based exit layer will demonstrate that the finest-granularity Early Exit can achieve a theoretical inference speedup of up to 3.3×. Following this, we will delve into why existing Early Exit techniques have been largely confined to the academic domain so far. Experiments will elucidate the critical challenge posed by KV Cache Absence, which creates a dilemma between autoregressive inference quality and KV cache recovery overhead. Existing methods often compromise between these two factors, leading to a situation where the promised theoretical speedup often fails to materialize under practical inference frameworks.

3.1 Sequence-level Exit and Token-level Exit

Two distinct exit granularities exist when applying Early Exit to Large Language Models: Sequence-level Exit and Token-level Exit. For a single autoregressive generation step in a decoder-only model, Sequence-level Exit mandates a fixed exit position for every token within that sequence. For instance, LayerSkip (Elhoushi et al., 2024) directly employs a fixed layer E to specify the stopping layer for the Draft Model during the entire generation process. Balcony (Jamialahmadi et al., 2025) is even more straightforward, using Balcony-L/M/S to denote inference modes that exit at $\frac{3}{4}$, $\frac{1}{2}$, and $\frac{1}{4}$ of the model depth, respectively. The primary advantage of Sequence-level Exit is its simplicity of implementation and direct portability from previous Early Exit works (e.g., BranchyNet (Teerapittayanon et al., 2016)). However, the exit layer for the entire sequence must be determined by the latest exit token, which leads to significant lost opportunities for early termination during long sequence generation. While Sequence-level Exit still offers acceleration in specific contexts, such as when LayerSkip (Elhoushi et al., 2024) and SpecEE (Xu et al., 2025) integrate Early Exit with Speculative Decoding for early termination on a limited-length draft token sequence. These methods only demonstrate efficacy in constrained scenarios, and their scalability remains highly limited.

In contrast, Token-level Exit has emerged as the prevailing paradigm for decoder-only LLMs due to its superior granularity. For instance, SkipDecode (Del Corro et al., 2023) explicitly highlights its "token-level" nature as a core feature. This approach allows each token in an autoregressive sequence to exit at its own optimal depth. To quantify the potential computational gains, we profile the optimal exit position for each token (defined as the the shallowest exit layer yielding the same prediction as the final layer) using Llama3.2 1B on the GSM8K benchmark with batch size of 1. Fig. 2a illustrates the distribution of these optimal exit layers, while Fig. 2b compares the average layers executed by Token-level and Sequence-level Exit at comparable accuracy levels. Despite the inherent difficulty of GSM8K for a model of this scale, Token-level Exit achieves a theoretical speedup of 3.3× with 6% accuracy drop, assuming zero overhead for KV Cache recovery and exit decision logic. Evaluations on the GSM8K benchmark (averaging over 100 tokens per response) underscore that **Token-level Exit possesses potential far exceeding that of Sequence-level Exit, especially for long-sequence generation**. However, the unique autoregression inference of decoder-only models introduces a critical bottleneck for Token-level Exit: KV Cache Absence.

3.2 Key Challenge: KV Cache Absence for Decoder-only Model

As illustrated in Figure 1, the KV Cache is one of the core mechanisms for autoregressive inference in decoder-only language models. While KV Cache prevents the redundant recomputation of keys and values for preceding tokens during each generation step, it consequently creates a data dependency between the generation process of the current token and that of previous tokens. An early exit of a preceding token directly leads to the absence of KV values from the skipped decoder layers (termed KV Cache Absence). This subsequently prevents the forward pass of later tokens that rely

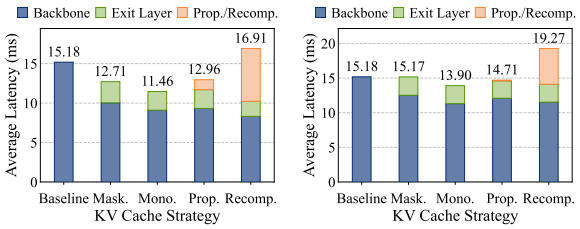


Figure 3: Average $ms/token$ of Token-level Exit using difference KV Cache Strategy on GSM8K. (a) Relaxed threshold, $Score \approx 0.15$. (b) Strict threshold, $Score \approx 0.25$.

on these layers from accessing the necessary prior Keys and Values. KV Cache Absence is the fundamental challenge distinguishing LLM Early Exit from other traditional neural networks. Almost all Token-level Exit works acknowledge this problem and attempt to address it. These existing strategies can be broadly categorized into four types:

- **Batching Recompute**: The principle of KV Recompute is straightforward: when the layer about to be executed lacks the KV Cache of a preceding token, the missing KV Cache is recalculated. Early Exit methods based on Batching Recompute typically cache the hidden state before exit. If the forward pass encounters KV Cache Absence, the previously saved hidden state is used to resume the inferencing until the necessary KV data is satisfied. While this approach reduces memory footprint of key and values, it offers negligible computational savings during long sequence generation and requires extra hardware batching support. For instance, EE-LLM (Chen et al., 2024) utilizes model parallelism and scheduling mechanisms to allow KV Recompute to run as a parallel cudastream, leveraging idle GPU resources to mitigate its overhead.

- **Mono-Decreasing Exit**: Mono-Decreasing Exit circumvents the KV Cache problem by regulating the token exit positions. For example, SkipDecode (Del Corro et al., 2023) strictly enforces that the exit position of every token within a single sequence adheres to a monotonically decreasing constraint to meet the KV Cache requirements of subsequently generated tokens. At the cost of restricting exit opportunities, Mono-Decreasing Exit offers a simple solution to avoid the KV Cache Absence issue.

- **State Propagation**: State Propagation is a classic Early Exit KV strategy, directly adopted by DAT (Elbayad et al., 2020), CALM (Schuster et al., 2022), and ELLM (Miao et al., 2024). In contrast to KV Recompute, State Propagation copies the hidden states from the current exit layer to serve as input for every subsequent decoder layer, directly computing the KV Cache for those skipped layers. This design represents a trade-off that balances the impact of KV Cache recovery on generation quality and inference speed.

- **KV Mask**: KV Mask is an aggressive strategy that modifies the original Causal Mask mechanism to make all tokens with missing KV Cache invisible to the current decoder layer. This naturally leads to a noticeable impact on generation quality. To compensate, D-LLM (Jiang et al., 2024) places a constraint on the minimum allowed exit layer to ensure limited quality degradation.

We evaluated the average per-token latency of these strategies on Llama3.2 1B at two consistent accuracy level, with results summarized in Fig. 3. The KV Mask approach exhibits the highest backbone latency, as it necessitates executing deeper layers to compensate for the precision degradation caused by missing KV values. While KV Recompute involves fewer backbone layers, it incurs substantial computational costs and disrupts memory access efficiency during long-sequence generation without specialized multi-GPU scheduling. State Propagation offers an approximation that trades off accuracy for overhead; however, its practical speedup remains inferior to the Mono-Decreasing Exit, which achieves better latency at the cost of restricting exit flexibility.

In summary, none of these strategies fully eliminates the penalty of KV Cache Absence. Neglecting KV integrity leads to severe performance degradation; recomputation significantly hampers the net acceleration; and imposing exit constraints severely limits the inherent potential of Token-level Exit. These overheads and limitations collectively obstruct the practical realization of early exit benefits.

To address these challenges, we propose River-LLM, a **seamless** exit framework. We define an

Table 1: Decoder-only Transformer Early Exit Techniques

Exit Granularity	KV Cache Strategy	Representative Methods	Seamless Exit	Train	Latency
Sequence Level	None (Independent)	Balcony	No	Train Exit Layer	High
	Speculation KV Reuse	Layerskip, SpecEE	No	Train Draft Model	Medium
Limited Token Level	Mono-Decreasing Exit	SkipDecode	No	Train Skip Predictor	Medium
Token Level	Batching Recompute	EE-LLM	No	Train Exit Layer	High
	State Propagation	ELLM, CALM	No	Train Exit Layer	Medium
	KV Mask	D-LLM	No	Train Skip Predictor	Medium
	KV Share	**River-LLM (Ours)**	**Yes**	**Train-Free**	**Low**

LLM Early Exit mechanism as "**seamless**" if it inherently achieves:

- **Granular Freedom**: Individual tokens can exit at arbitrary layers independently.
- **Intrinsic KV Integrity**: The KV cache for skipped layers is automatically populated as a byproduct of the exit path's execution, eliminating the need for post-exit recovery or recomputation.

Table 1 shows the core advantages of River-LLM over previous methods. Built upon the principle of KV Sharing, River-LLM enables tokens to traverse an "Exit River" rapidly regardless of their exit point. Crucially, the exit layers naturally substitute for the skipped decoder layers to generate a complete KV cache without incurring additional operational overhead.

4 Seamless Exit: River-LLM

River-LLM is designed as a scalable and architecture-agnostic framework for early exit, applicable to various decoder-only Transformer models. This section first introduces the KV-Shared Exit Layer, the core component responsible for maintaining Intrinsic KV Integrity. We then detail how these modules are orchestrated during inference to achieve maximum Granular Freedom and substantial acceleration.

4.1 KV-Shared Exit Layer

As illustrated in Fig. 4a, the exit layer of River-LLM is designed as a lightweight plug-in specifically for decoder-only Transformers. It adopts a river-like topology, where each exit layer serves as a direct mapping of its corresponding backbone decoder. To ensure Intrinsic KV Integrity, each exit layer is assigned the identical KV Cache addressing scheme as its backbone counterpart. Initially, the exit layers inherit the backbone's architecture and parameters. Subsequently, we apply Post-Training Quantization (PTQ) to the weights of the Attention and Feed-Forward Network blocks within these layers. Specifically, we utilize a 4-bit weight-only quantization (W4A16) scheme, while maintaining the KV Cache in FP16 format to preserve representation density. By leveraging quantization and specialized inference kernels optimized via partial graph compilation, the exit layer allows the hidden states to traverse the "Exit River" with a $2.4\times$ throughput enhancement over the full-precision backbone blocks, while the synthesized KV Cache remains highly consistent with the backbone's native output. According to Fig. 5, regardless of the exit point, the average cosine similarity between the exit KV Cache and the backbone's native KV Cache remains above 0.97. Notably, the KV-Shared mechanism combined with PTQ maintains exceptional Intrinsic KV Integrity without requiring any training; the entire weight transfer process typically concludes within one minute.

4.2 Inference with Seamless Exit

To maximize Granular Freedom, River-LLM's exit layers are interconnected in a serial topology starting from a predefined entry layer L_s. Each exit layer is logically coupled with its corresponding backbone decoder block. During the autoregressive generation at each decoding step t, the l^{th} layer's hidden state $\mathbf{h}_t^{(l)}$ undergoes an exit evaluation prior to entering the next decoder block. As illustrated in Fig. 4b, if the exit criteria are met, the remaining computation is offloaded to the accelerated exit layers, eventually reaching the original LM Head to generate token logits. Exit decision is paramount for maintaining KV Integrity. As evidenced by our analysis in Fig. 5b, we observe a cumulative quantization discrepancy: hidden states passing through multiple 4-bit exit layers exhibit a slight increase in error relative to the backbone. For instance, when exiting at layer 1, the Value vec-

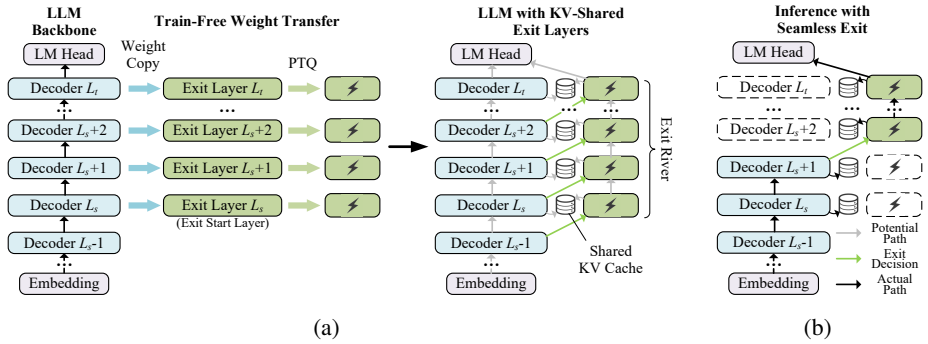


Figure 4: Seamless exit architecture and inference paradigm: River-LLM. (a) KV-shared exit layer. (b) Inference with seamless exit.

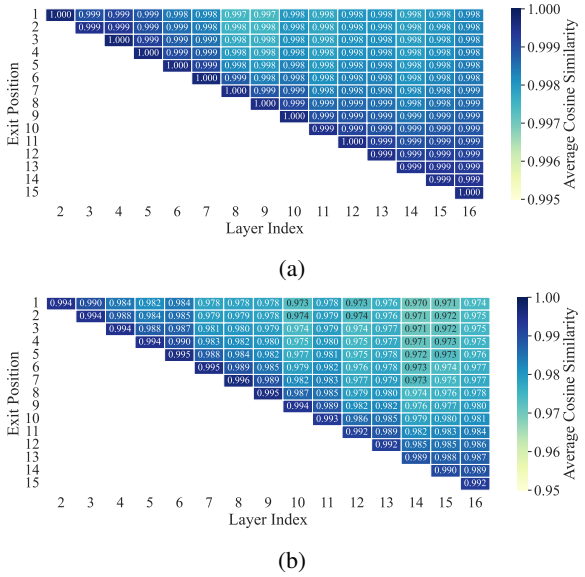


Figure 5: KV Cache similarity between exit layer and backbone decoder of Llama3.2 1B. (a) Key. (b) Value.

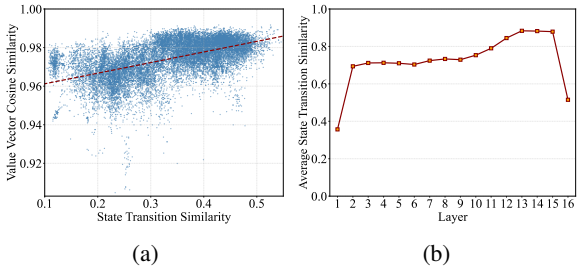


Figure 6: (a) Relationship between first layer state transition similarity and last layer backbone-exit value vector similarity ($r = 0.5536, p < 0.001$). (b) Trend of state transition similarity with the change of layer number.

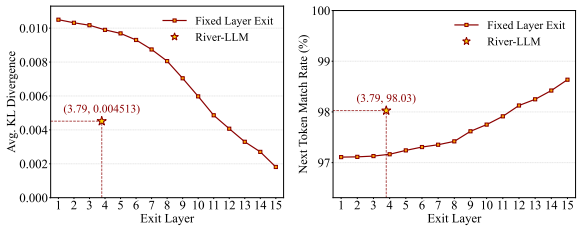


Figure 7: (a) Average KL Divergence and (b) next token match rate between backbone output and exit prediction, $\tau = 0.5$.

tor similarity begins to fluctuate between 0.97 and 0.98 after the 7th exit layer. Notably, we found that the state transition similarity (i.e., the input-output cosine similarity) of early backbone layers serves as a reliable proxy for predicting this cumulative discrepancy. As shown in Fig. 6a, a moderate positive correlation ($r = 0.5536$) exists between the first layer's state transition similarity and the final layer's backbone-exit value similarity. Leveraging this observation, we define the exit decision $\mathcal{D}^{(l)}$ at layer l as:

$$\mathcal{D}^{(l)} = \mathbb{I}\left(\min_{b\in\mathcal{B}} s_{t,b}^{(l)} > \tau\right), s_{t,b}^{(l)} = \frac{\mathbf{h}_{t,b}^{(l-1)\top}\mathbf{h}_{t,b}^{(l)}}{\|\mathbf{h}_{t,b}^{(l-1)}\|\|\mathbf{h}_{t,b}^{(l)}\|} \quad (1)$$

where $s_{t,b}^{(l)}$ is the state transition similarity of generation step t at layer l, τ is the exit threshold and $\mathcal{B}$ is the current batch. Based on Fig. 6b, cosine similarity generally follows an upward trend across layers, indicating that early and terminal layers exert a more pronounced influence on the hidden states. Except for the last layer, the state transition similarity shows a roughly monotonically increasing trend, which conforms to the objective law of early exit, that is, most of the layers after the exit layer also meet the conditions for early exit.

Building upon this observation, River-LLM facilitates a resource-efficient deployment strategy termed Backbone Offloading. Since the vast majority of tokens terminate their backbone traversal at early stages, the framework can autonomously des-

ignate the subsequent, sparsely activated backbone blocks for eviction from primary VRAM. This strategy allows the model to operate within a memory footprint comparable to a fully quantized baseline while ensuring that the Exit River remains resident to provide continuous semantic completion for the few tokens that may require deeper processing.

To harmonize with existing inference frameworks, River-LLM adaptively switches its granularity based on the inference phase:

- Prefill Phase: The prompt is processed via Sequence-level Exit, where all tokens exit at a unified depth to maintain the efficiency of parallelized attention kernels.
- Generation Phase: Inference switches to Token-level Exit, allowing individual tokens to terminate at their optimal depths to maximize speedup.

A significant advantage of River-LLM over full-model quantization is its selective computational fidelity. By allowing "difficult" or high-entropy tokens to traverse the backbone in full precision while offloading "easy" tokens to the Exit River, River-LLM preserves the model's representational robustness in complex scenarios. As illustrated in Figure 7, the quantization discrepancy between the Exit River and the full backbone accumulates as the exit position moves earlier. This accumulation leads to an upward trend in KL divergence between their logit distributions and a corresponding decrease in the next-token match rate. River-LLM leverages this relationship by utilizing the estimated quantization discrepancy to guide precise exit decisions. On the GSM8K benchmark, River-LLM maintains a next-token match rate of 98.03% and a negligible KL divergence of 0.0045 while achieving an average exit depth of 3.79 layers. This error-aware mechanism ensures that the framework preserves both semantic and KV integrity even when substantial backbone layers are bypassed. Furthermore, by tuning τ, River-LLM facilitates a flexible accuracy-speed trade-off, which we compare against fully quantized baselines in Section 5.3.

5 Evaluation

5.1 Experimental Setup

Models and Benchmarks. We evaluate the proposed River-LLM using following representative backbones: Llama3.2 1B, Llama3.1 8B, Phi4-mini and Ministral3 8B. Our evaluation covers a diverse suite of eight benchmarks:

- Common Sense Reasoning: We include BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), ARC-Challenge (Clark et al., 2018), ARC-Easy, and MMLU (Hendrycks et al., 2020). For these tasks, we report accuracy based on loglikelihood ranking.
- Long Sequence Generation: We evaluate mathematical reasoning on GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), alongside code generation on HumanEval (Chen et al., 2021). These benchmarks serve as the primary tasks for measuring practical wall-clock speedup.

Evaluation Protocol. All experiments are conducted on an NVIDIA A40 GPU. To ensure robust assessment, we adopt a 5-shot setting for GSM8K and a 4-shot setting for MATH, while all other benchmarks are evaluated in a 0-shot configuration.

5.2 Benchmark Accuracy

We first conduct a comprehensive evaluation of River-LLM's accuracy on Llama3.2 1B and Llama3.1 8B across various benchmarks. As demonstrated in Table 2, under the default threshold ($\tau = 0.5$), River-LLM achieves performance competitive with the baseline while executing only 3 to 4 backbone layers on average. This remarkable efficiency is attributed to the KV-Shared Exit River, which maintains KV integrity and enables tokens to exit at extremely early stages without significant semantic degradation. By increasing the threshold to $\tau = 0.7$, River-LLM achieves nearly lossless inference. For the 32-layer Llama3.1 8B, the majority of commonsense reasoning and even complex generation tasks can terminate well before the median layer, significantly reducing computational overhead. Interestingly, we observe that River-LLM can even outperform the full-model baseline in specific scenarios, such as HumanEval on Llama-3.1-8B (57.3 vs. 55.5). This suggests that by bypassing redundant deeper layers, the Exit River may effectively mitigate cumulative noise or overthinking, thereby enhancing the model's calibration on certain tasks. Consequently, River-LLM proves to be a robust framework that preserves generation quality for complex problems while fully unlocking acceleration potential on simpler tokens.

Table 2: Evaluation results of River-LLM across diverse benchmarks. The exit position indicates the average number of executed backbone layers. τ denotes the threshold for the exit decision.

	Llama3.2 1B (16 layers)					Llama3.1 8B (32 layers)				
Benchmark	Accuracy			Exit Position		Accuracy			Exit Position	
	Backbone	$\tau = 0.5$	$\tau = 0.7$	$\tau = 0.5$	$\tau = 0.7$	Backbone	$\tau = 0.5$	$\tau = 0.7$	$\tau = 0.5$	$\tau = 0.7$
BoolQ	69.4	**67.5**	69.2	**3.90**	15.02	84.1	**83.4**	83.4	**3.07**	9.74
HellaSwag	45.1	**44.3**	44.9	**3.71**	10.94	59.1	**58.5**	58.6	**3.09**	8.80
ARC-c	35.7	**35.2**	35.9	**3.24**	10.68	51.5	**50.3**	51.2	**3.01**	8.44
ARC-e	68.4	**67.8**	67.1	**3.24**	10.68	81.7	**82.0**	81.4	**3.01**	8.44
MMLU	46.1	**44.3**	46.0	**4.05**	12.70	68.0	**66.1**	67.4	**3.04**	14.01
GSM8K	33.5	**29.3**	33.5	**3.79**	15.05	78.2	**74.4**	75.6	**2.96**	26.98
MATH	17.8	**14.6**	17.0	**3.56**	14.67	27.0	**26.6**	25.7	**2.81**	13.10
HumanEval	25.8	**23.2**	25.7	**2.30**	10.05	57.3	**55.5**	57.3	**2.16**	5.84

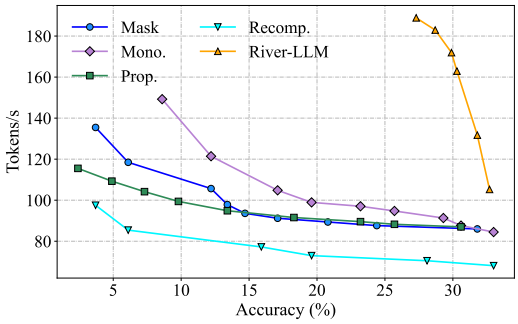


Figure 8: Trade-off between generation throughput and GSM8K accuracy on Llama3.2 1B. River-LLM has exceptional accuracy retention capability.

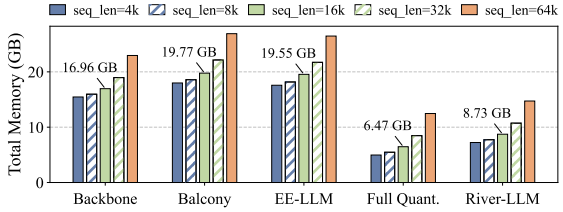


Figure 9: Peak GPU memory usage of Llama3.1 8B with different methods, `batch_size` $= 1$.

5.3 Generation Speedup

Table 3 reports the practical wall-clock speedup of River-LLM during autoregressive generation on long-sequence benchmarks, including GSM8K, MATH, and HumanEval, with a batch size of 1. We report the average value of 10 tests. For Llama3.2 1B and Llama3.1 8B, we adopt a default exit threshold of 0.5; for Phi4-mini and Ministral3 8B, the threshold is set to 0.9. As a robust baseline for ablation, we evaluate the backbone under Full Quantization (Full Quant.) using the same HQQ framework (Badri and Shaji, 2023) combined with compilation optimization. While static quantization achieves marginally higher peak throughput by applying uniform optimization across all tokens, it incurs significant accuracy degradation due to accumulated precision loss, especially on "difficult" tokens, without the benefit of QAT fine-tuning. In contrast, River-LLM delivers comparable speedups ($\approx 10\%$ lower than Full Quant.) while maintaining near-lossless fidelity to the original backbone's performance.

The versatility of River-LLM is further demonstrated by its ability to navigate the speedup-accuracy trade-off via the exit threshold τ. As illustrated in Fig. 8, River-LLM consistently outperforms methods with alternative KV strategies, such as masking, recomputation, and state propagation, by a substantial margin. Notably, the performance curve of River-LLM defines a superior Pareto frontier. Unlike traditional methods that suffer from a "cliff-like" drop in accuracy when seeking higher throughput, River-LLM effectively harmonizes the concepts of dynamic depth and KV integrity. This synergy allows the model to selectively skip computation for simpler tokens while preserving the full capacity of the backbone for complex reasoning, thereby achieving an optimal balance between inference efficiency and generation quality.

5.4 Memory and Latency Overhead

To evaluate the deployment efficiency of River-LLM, we analyze its GPU memory consumption across varying sequence lengths, ranging from 4K to 64K tokens. As illustrated in Fig. 9, prior early exit frameworks such as Balcony and EE-LLM incur substantial memory overhead that regularly exceeds the original backbone, primarily due to the retention of additional exit layer parameters or duplicated KV cache states. In contrast, by leveraging the single shared KV mechanism alongside the backbone offloading strategy, River-LLM dramatically reduces memory requirements across

Table 3: Practical generation speedup and accuracy of River-LLM compared with the backbone and full quantization baselines. Throughput is measured in tokens per second (Tokens/s) on an NVIDIA A40 GPU.

Llama3.2 1B									
	GSM8K			**MATH**			**HumanEval**		
	Backbone	Full Quant.	**River-LLM**	Backbone	Full Quant.	**River-LLM**	Backbone	Full Quant.	**River-LLM**
Accuracy	33.2	25.1	29.3	17.8	12.2	14.6	25.8	20.4	23.2
Tokens/s	84.5	195.5	182.9	100.5	208.8	189.2	100.4	190.6	171.8
Speedup	1.00×	2.31×	**2.16**×	1.00×	2.08×	**1.88**×	1.00×	1.90×	**1.71**×
Llama3.1 8B									
	GSM8K			**MATH**			**HumanEval**		
	Backbone	Full Quant.	**River-LLM**	Backbone	Full Quant.	**River-LLM**	Backbone	Full Quant.	**River-LLM**
Accuracy	78.2	69.8	74.4	27.0	23.1	26.6	57.3	50.2	55.5
Tokens/s	25.3	47.5	45.0	25.1	47.6	43.1	25.2	47.7	44.7
Speedup	1.00×	1.88×	**1.78**×	1.00×	1.89×	**1.72**×	1.00×	1.89×	**1.77**×
Phi4-mini									
	GSM8K			**MATH**			**HumanEval**		
	Backbone	Full Quant.	**River-LLM**	Backbone	Full Quant.	**River-LLM**	Backbone	Full Quant.	**River-LLM**
Accuracy	82.1	75.9	81.0	37.4	28.8	35.1	63.4	57.9	63.1
Tokens/s	71.4	136.8	115.0	62.41	110.7	95.2	69.5	132.1	118.6
Speedup	1.00×	1.92×	**1.61**×	1.00×	1.77×	**1.53**×	1.00×	1.90×	**1.71**×
Ministral3 8B									
	GSM8K			**MATH**			**HumanEval**		
	Backbone	Full Quant.	**River-LLM**	Backbone	Full Quant.	**River-LLM**	Backbone	Full Quant.	**River-LLM**
Accuracy	84.5	84.1	84.3	48.1	46.0	46.6	29.3	20.7	26.2
Tokens/s	34.9	65.0	61.8	32.0	62.7	59.1	33.7	71.3	66.6
Speedup	1.00×	1.86×	**1.77**×	1.00×	1.96×	**1.85**×	1.00×	2.12×	**1.98**×

all context lengths. Its total memory footprint remains consistently and significantly lower than both the backbone and existing early exit baselines, approaching the memory efficiency of a fully quantized model while still preserving selective high-precision inference capabilities.

Beyond memory efficiency, River-LLM introduces negligible latency overhead during inference. The exit criterion relies on computing the state transition similarity between consecutive hidden states, which, for a hidden dimension d, incurs a time complexity of $\mathcal{O}(d)$. Empirical profiling corroborates this efficiency: the exit decision logic executes in approximately 100 microseconds on Llama3.1 8B, accounting for a mere 0.0688% of the total per-token inference time. This confirms that River-LLM's seamless exit mechanism delivers substantial speedup with minimal overhead.

6 Conclusion

In this paper, we address the KV Cache Absence problem, a fundamental bottleneck that hinders the practical efficiency of Early Exit in decoder-only Large Language Models. Our empirical analysis demonstrates that existing remedies, ranging from recomputation to masking, fail to bridge the gap between theoretical layer reduction and actual wall-clock speedup due to their substantial latency overhead or precision degradation. To overcome these limitations, we introduce River-LLM, a training-free framework designed for seamless token-level Early Exit. By constructing a lightweight KV-Shared Exit River, River-LLM enables the backbone's missing KV cache to be naturally generated and preserved as an intrinsic byproduct of the exit process. Furthermore, we utilize state transition similarity within decoder blocks to guide precise exit decisions, ensuring high fidelity to the original backbone's output. Extensive evaluations on diverse benchmarks demonstrate that River-LLM achieves 1.71× to 2.16× practical speedup while maintaining near-lossless generation quality. Compared to prior dynamic inference methods, River-LLM defines a superior Pareto frontier, offering a flexible accuracy-speed trade-off without the need for additional training or fine-tuning. In summary, River-LLM provides a robust and scalable solution for efficient LLM inference, proving that maintaining KV integrity is the key to fully unlocking the token-level exit potential.

Limitations

- Our current evaluation focuses on representative models up to 8B parameters. While the framework is designed to be architecture-agnostic, further validation on larger scales, such as 24B and 70B models, is necessary to confirm its performance and scalability at extreme parameter counts.

- As River-LLM is primarily optimized for token-level autoregressive decoding, its efficiency gains are most significant during the generation phase. Consequently, the speedup is less pronounced for prefill-dominant tasks, such as those within the MMLU benchmark, where a sequence-level exit strategy is currently applied.

Acknowledgments

This work was supported in part by the the Open Research Fund of Peng Cheng Laboratory under Grant 2025KF1B0010. An Zou is the corresponding author.

The authors acknowledge the use of Gemini 3.0 Pro for polishing assistance. This assistance was limited to polishing, and all research findings and final content were independently verified by the authors.

References

Mehmet Emre Akbulut, Hazem Hesham Yousef Shalby, Fabrizio Pittorino, and Manuel Roveri. 2026. Infoq: Mixed-precision quantization via global information flow. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 19598–19606.

Hicham Badri and Appu Shaji. 2023. Half-quadratic quantization of large machine learning models.

Sébastien Bubeck and 1 others. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Hao Mark Chen, Fuwen Tan, Alexandros Kouris, Royson Lee, Hongxiang Fan, and Stylianos I. Venieris. 2025. Progressive mixed-precision decoding for efficient LLM inference. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Mark Chen and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Yanxi Chen, Xuchen Pan, Yaliang Li, Bolin Ding, and Jingren Zhou. 2024. Ee-llm: Large-scale training and inference of early-exit large language models with 3d parallelism. In *International Conference on Machine Learning*, pages 7163–7189. PMLR.

Christopher Clark and 1 others. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Peter Clark and 1 others. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

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

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. 2020. Depth-adaptive transformer. In *ICLR 2020-Eighth International Conference on Learning Representations*, pages 1–14.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, and 1 others. 2024. Layerskip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12622–12642.

Siqi Fan, Xin Jiang, Xiang Li, Xuying Meng, Peng Han, Shuo Shang, Aixin Sun, and Yequan Wang. 2025. Not all layers of llms are necessary during inference. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pages 5083–5091. International Joint Conferences on Artificial Intelligence Organization.

Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. 2021. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7436–7456.

Shwai He, Tao Ge, Guoheng Sun, Bowei Tian, Xiaoyang Wang, and Dong Yu. 2025a. Router-tuning: A simple and effective approach for dynamic depth. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 1925–1938.

Zhuomin He, Yizhen Yao, Pengfei Zuo, Bin Gao, Qinya Li, Zhenzhe Zheng, and Fan Wu. 2025b. Adaskip: Adaptive sublayer skipping for accelerating long-context llm inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24050–24058.

Dan Hendrycks and 1 others. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Dan Hendrycks and 1 others. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Lianming Huang, Shangyu Wu, Yufei Cui, Ying Xiong, Xue Liu, Tei-Wei Kuo, Nan Guan, and Chun Jason Xue. 2024. Raee: A robust retrieval-augmented early exiting framework for efficient inference. *arXiv preprint arXiv:2405.15198*.

Gagan Jain, Nidhi Hegde, Aditya Kusupati, Arsha Nagrani, Shyamal Buch, Prateek Jain, Anurag Arnab, and Sujoy Paul. 2024. Mixture of nested experts: Adaptive processing of visual tokens. *Advances in Neural Information Processing Systems*, 37:58480–58497.

Benyamin Jamialahmadi, Parsa Kavehzadeh, Mehdi Rezagholizadeh, Parsa Farinneya, Hossein Rajabzadeh, Aref Jafari, Boxing Chen, and Marzieh S. Tahaei. 2025. Balcony: A lightweight approach to dynamic inference of generative language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 24853–24867, Suzhou, China. Association for Computational Linguistics.

Yikun Jiang, Huanyu Wang, Lei Xie, Hanbin Zhao, Hui Qian, John Lui, and 1 others. 2024. D-llm: A token adaptive computing resource allocation strategy for large language models. *Advances in Neural Information Processing Systems*, 37:1725–1749.

Sangwoo Kwon, Seong Hoon Seo, Jae W. Lee, and Yeonhong Park. 2025. DP-LLM: Runtime model adaptation with dynamic layer-wise precision assignment. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Xiangjie Li, Chenfei Lou, Yuchi Chen, Zhengping Zhu, Yingtao Shen, Yehan Ma, and An Zou. 2023. Predictive exit: Prediction of fine-grained early exits for computation-and energy-efficient inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 8657–8665.

Akide Liu and 1 others. 2024. Minicache: Kv cache compression for large language models. *arXiv preprint arXiv:2405.14365*.

Xuan Luo, Weizhi Wang, and Xifeng Yan. 2025. Diffskip: Differential layer skipping in large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 7221–7231.

Ruijie Miao, Yihan Yan, Xinshuo Yao, and Tong Yang. 2024. An efficient inference framework for early-exit large language models. *arXiv preprint arXiv:2407.20272*.

Priyadarshini Panda, Abhronil Sengupta, and Kaushik Roy. 2016. Conditional deep learning for energy-efficient and enhanced pattern recognition. In *Design, Automation & Test in Europe Conference & Exhibition*. IEEE.

Reiner Pope and 1 others. 2023. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*.

Baptiste Rozière and 1 others. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. 2022. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472.

Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *23rd International Conference on Pattern Recognition*. IEEE.

Jason Wei and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*.

Jiaming Xu, Jiayi Pan, Yongkang Zhou, Siming Chen, Jinhao Li, Yaoxiu Lian, Junyi Wu, and Guohao Dai. 2025. Specee: Accelerating large language model inference with speculative early exiting. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture*, pages 467–481.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Minghui Chen, Zheng Lin, and Weiping Wang. 2025. Dynamic early exit in reasoning models. *arXiv preprint arXiv:2504.15895*.

Rowan Zellers and 1 others. 2019. Hellaswag: Can a machine really finish your sentence? In *ACL*.

Yi Zhang, Kai Zhang, Zheyang Li, Wenming Tan, Ye Ren, and Jilin Hu. 2025. Beyond dynamic quantization: An efficient static hierarchical mix-precision framework for near-lossless LLM compression. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 2573–2587, Suzhou (China). Association for Computational Linguistics.

A Appendix

A.1 Impact of Different Quantization Backends in Exit River

To demonstrate the framework's architecture-agnostic design, we evaluate the performance of River-LLM when integrated with various quantization backends for the Exit River path. As a dynamic inference framework, River-LLM can incorporate any mature compression technique that preserves

Table 4: Different Exit River Quantization on Llama3.1 8B.

Quant. Method	Exit Threshold	Token/s	GSM8K Score
Backbone	N/A	25.3	78.2
River+HQQ	0.5	45	74.4
HQQ	full quant	47.5	69.8
River+AWQ	0.5	46.9	77.3
AWQ	full quant	48.1	76.2

Intrinsic KV Integrity during the exit process. Table 4 summarizes the results for Llama-3.1-8B using HQQ (the default method used by River-LLM) and AWQ as alternative quantization schemes.

The results indicate that advancements in static quantization directly translate to improved performance within the River-LLM framework. When the Exit River is upgraded from the default HQQ to the more advanced AWQ, the model achieves higher accuracy across reasoning tasks. Specifically, the River + AWQ configuration achieves a GSM8K score of 77.3, outperforming the static AWQ baseline (76.2) while maintaining competitive throughput.

This precision gain demonstrates that River-LLM acts as a complementary path-level optimization for operator-level quantization. By adaptively routing high-entropy tokens through the full-precision backbone, River-LLM effectively mitigates the cumulative quantization noise inherent in static low-bit models. These findings confirm that River-LLM can enhance the accuracy of existing lightweight methods with a negligible impact on acceleration efficiency.

A.2 Benchmark results of Phi4-mini and Ministral3 8B

Table 5 presents the additional benchmark evaluation of River-LLM on the Phi4-mini and Ministral3 8B architectures. Consistent with the findings on Llama models, the framework preserves generation quality across diverse benchmarks while reducing the overall computational depth. A key observation from these experiments is that newer model architectures demonstrate stronger early-layer semantic understanding, causing tokens to converge to stable representations at much shallower layers. Consequently, we applied stricter exit thresholds ($\tau = 0.8$ and $\tau = 0.9$) for these models to simulate a near-lossless acceleration scenario. Under these elevated thresholds, River-LLM maintains high-fidelity performance across both common sense and complex reasoning tasks while still bypassing a significant portion of the backbone, further confirming the architecture-agnostic scalability of the KV-shared exit mechanism.

A.3 Detailed Memory Usage of River-LLM and Other Baselines

Table 6 provides a granular decomposition of peak GPU memory consumption, partitioned into model parameters, KV cache, and temporary activations. A key observation is the scalability of the KV-shared architecture in long-sequence scenarios. While representative early exit frameworks like Balcony and EE-LLM require auxiliary KV cache sets for their respective exit layers or recomputation states (resulting in higher GPU memory consumption than the original backbone) River-LLM maintains a single set of KV caches identical to the backbone baseline. Consequently, as the context length increases to 64K, River-LLM avoids the memory inflation inherent in competing dynamic depth strategies.

Regarding parameter footprint, the Backbone-Offloading mode demonstrates significant efficiency gains. By offloading under-utilized backbone layers while retaining the Exit River, the parameter memory is reduced from 14.96 GB to 6.73 GB. This configuration allows the total memory footprint to approach that of a static 4-bit quantization baseline while selectively preserving high-precision execution for the critical initial layers of the model. These results confirm that River-LLM offers a flexible deployment strategy where the trade-off between memory capacity and generation fidelity is explicitly manageable.

A.4 Additional Discussion: River-LLM v.s. Mixed-Precision LLM

Mixed-precision quantization has emerged as an orthogonal paradigm to Early Exit for optimizing LLM inference efficiency. Existing research typically falls into two categories: static and dynamic assignment. Static frameworks, such as SHMQ (Zhang et al., 2025), pre-calculate hierarchical bit-width configurations using Hessian matrices to reflect model-inherent sensitivity, InfoQ (Akbulut et al., 2026) determines static precision assignments by measuring the impact of layer-wise quantization on global mutual information flow and solving an integer linear programming problem prior to deployment. Conversely, dynamic approaches emphasize that layer importance shifts during gen-

Table 5: Additional evaluation results of River-LLM on Phi4-mini and Ministral3 8B across diverse benchmarks.

	Phi4-mini (32 layers)					Ministral3 8B (34 layers)				
Benchmark	Accuracy			Exit Position		Accuracy			Exit Position	
	Backbone	$\tau = 0.8$	$\tau = 0.9$	$\tau = 0.8$	$\tau = 0.9$	Backbone	$\tau = 0.8$	$\tau = 0.9$	$\tau = 0.8$	$\tau = 0.9$
BoolQ	84.3	84.1	**84.3**	2.35	**4.93**	85.9	85.2	**85.4**	2.04	**3.83**
HellaSwag	54.4	53.7	**53.9**	2.05	**4.14**	58.5	58.2	**58.2**	2.00	**2.38**
ARC-c	57.1	54.0	**54.4**	2.02	**3.45**	62.3	62.2	**62.2**	2.01	**2.11**
ARC-e	82.8	81.3	**81.3**	2.02	**3.45**	86.5	85.9	**85.9**	2.01	**2.11**
MMLU	66.7	64.1	**65.4**	2.07	**4.48**	73.1	72.7	**72.7**	2.01	**2.76**
GSM8K	82.1	79.2	**81.0**	6.10	**13.79**	84.5	84.2	**84.3**	6.16	**6.70**
MATH	37.4	33.1	**35.1**	5.77	**13.69**	48.1	46.4	**46.6**	5.68	**6.18**
HumanEval	63.4	62.8	**63.1**	2.55	**7.72**	29.3	22.6	**26.2**	2.51	**7.02**

Table 6: Peak GPU Memory Usage of Different Methods for Llama3.1 8B, `batch_size` $= 1$.

Method	Context Length	Parameters (GB)	KV Cache (MB)	Temp Act. (MB)	Total Mem. (GB)
Backbone	4,096	14.96	512	1.25	15.46
	8,192	14.96	1024	1.25	15.96
	16,384	14.96	2048	1.25	16.96
	32,768	14.96	4096	1.25	18.96
	65,536	14.96	8192	1.25	22.96
Balcony	4,096	17.4	608	1.25	17.99
	8,192	17.4	1216	1.25	18.58
	16,384	17.4	2432	1.25	19.77
	32,768	17.4	4864	1.25	22.15
	65,536	17.4	9728	1.25	26.9
EE-LLM	4,096	16.98	601	1.25	17.58
	8,192	16.98	1202	1.25	18.17
	16,384	16.98	2397	1.25	19.55
	32,768	16.98	4836	1.25	21.72
	65,536	16.98	9701	1.25	26.47
Full Quantize	4,096	4.47	512	1.25	4.97
	8,192	4.47	1024	1.25	5.47
	16,384	4.47	2048	1.25	6.47
	32,768	4.47	4096	1.25	8.47
	65,536	4.47	8192	1.25	12.47
River-LLM	4,096	6.73	512	1.25	7.23
	8,192	6.73	1024	1.25	7.73
	16,384	6.73	2048	1.25	8.73
	32,768	6.73	4096	1.25	10.73
	65,536	6.73	8192	1.25	14.73

eration. PMPD (Chen et al., 2025) introduces a progressive decoding schedule that reduces precision as the sequence length increases. Representing the current state-of-the-art in token-wise adaptation, DP-LLM (Kwon et al., 2025) switches bit-widths at each decoding step based on the statistical properties of input activations, aiming to maximize throughput by navigating the sensitivity of different layers across the generation process.

River-LLM differs from mixed precision frameworks like DP-LLM in its optimization objective, deployment complexity, and accuracy retention. Unlike DP-LLM, which necessitates a calibration phase to train linear regressors for precision assignment, River-LLM is entirely training-free and introduces a nearly negligible decision overhead of approximately 0.0688%. Furthermore, the two paradigms represent divergent acceleration trajectories: while mixed-precision methods focus on aggressive hardware throughput via low-bit optimization, River-LLM prioritizes preserving the original backbone's reasoning fidelity. Our comparative analysis highlights this distinction in performance stability. On the GSM8K benchmark, River-LLM on Llama3.1 8B maintains a high-fidelity accuracy range of 74.4%–78.2% across a target bit range of 4.00–16.00. In contrast, DP-LLM exhibits a more pronounced performance drop on Llama3 8B, with accuracy fluctuating between 36.7% and 46.9% within a lower target bit range of 3.25–4.75. These results indicate that River-LLM offers a superior Pareto frontier for applications requiring near-lossless generation quality, whereas mixed-precision methods are optimized for extreme bit-reduction scenarios that may impose specific hardware requirements. River-LLM thus provides a lightweight, depth-aware mechanism that complements operator-level quantization by adaptively bridging the gap between high-fidelity backbone execution and accelerated inference.