SpecExtend: A Drop-in Enhancement for Speculative Decoding of Long Sequences

Anonymous ACL submission

Abstract

Speculative decoding is a widely adopted technique for accelerating inference in large language models (LLMs), but its performance degrades on long inputs due to increased attention cost and reduced draft accuracy. We introduce SpecExtend, a drop-in enhancement that improves the performance of speculative decoding on long sequences without any additional training. SpecExtend integrates efficient attention mechanisms such as FlashAttention and Hybrid Tree Attention into both the draft and target models, reducing latency across all stages. To improve draft accuracy, we propose Cross-model Retrieval, a novel cache update strategy that uses the target model's attention scores to dynamically select relevant context for the draft model. Extensive evaluations on three long-context understanding datasets show that SpecExtend accelerates standard tree-based speculative decoding by up to 2.22× for inputs up to 16K tokens, providing an effective solution for speculative decoding of long sequences.

1 Introduction

Large Language Models (LLMs) have achieved remarkable success across a wide range of natural language processing (NLP) tasks (Achiam et al., 2023; Grattafiori et al., 2024). However, their practical deployment is often hindered by high inference latency, which is primarily caused by the autoregressive nature of decoding (Zhou et al., 2024). To address this issue, various optimization techniques have been proposed, with speculative decoding emerging as an effective, lossless solution (Leviathan et al., 2023; Chen et al., 2023). Speculative decoding consists of two phases: First, a smaller draft model is used to efficiently generate multiple candidate tokens. Then, the original target model verifies these tokens in parallel. This allows generating multiple tokens within a single target model decoding step, accelerating inference without altering the output distribution.

However, the performance of speculative decoding frameworks drops significantly as input length increases. We identify two primary causes: (1) increased latency in both drafting and verification steps due to the quadratic complexity of standard attention, and (2) reduced draft accuracy, as the draft model is typically smaller and trained only on short sequences. Meanwhile, retraining draft models on long contexts is costly, highlighting the need for a drop-in solution that improves long-input performance while preserving the original benefits of existing frameworks.

To this end, we propose SpecExtend, a drop-in enhancement for speculative decoding on long inputs. SpecExtend accelerates the forward passes of both the draft and target models by integrating efficient attention mechanisms across all stages. In order to improve drafting accuracy on long inputs without additional training, we introduce Cross-model Retrieval, a novel cache update strategy for speculative decoding. We dynamically update the draft model's KV cache with globally relevant context, guided by the target model's attention scores.

SpecExtend is compatible with a wide range of speculative decoding frameworks, including tree-based structures, self-speculative draft models, and dynamic tree expansion techniques. We adopt these settings to evaluate SpecExtend's effectiveness using both off-the-shelf LLMs and EAGLE (Li et al., 2024c) as draft models. On three long-context understanding datasets, SpecExtend accelerates standard tree-based speculative decoding by up to 2.22× for inputs up to 16K tokens, resulting in an overall speedup of 2.87× over naive autoregressive generation. SpecExtend preserves performance on short inputs and does not require retraining, offering a robust drop-in solution for enhancing speculative decoding on long inputs.

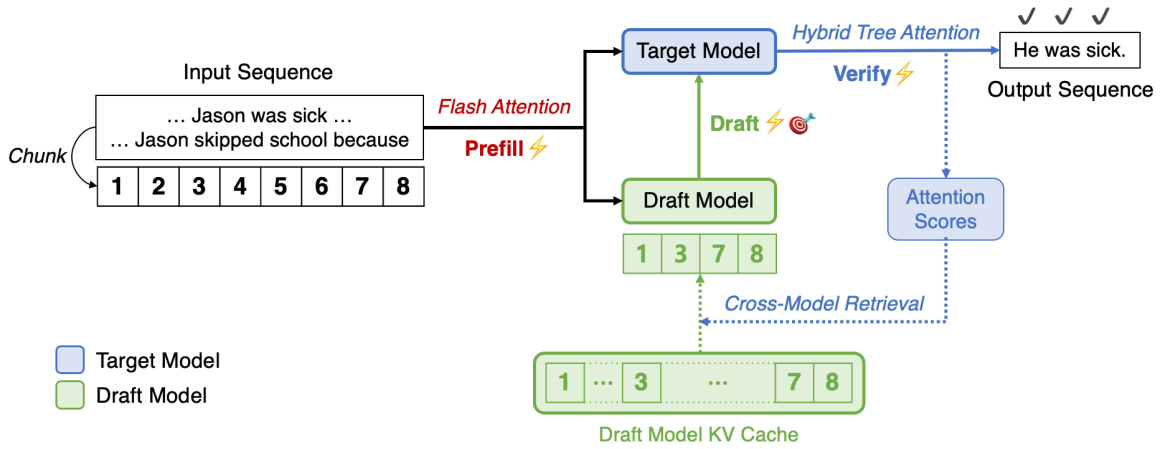

Figure 1: Overview of SpecExtend. FlashAttention accelerates the prefill phases of both target and draft models, and Hybrid Tree Attention accelerates the verification phase. The target model's attention scores are obtained from the verification step and used to select the most relevant input chunks to retain in the draft model's KV cache.

2 SpecExtend

2.1 Efficient Attention Mechanisms

Prefill Acceleration The initial forward pass of a language model computes full self-attention over the entire input sequence, incurring quadratic memory usage and latency. FlashAttention (Dao et al., 2022; Dao, 2023) mitigates this by avoiding materialization of large intermediate matrices in the GPU high-bandwidth memory. We apply FlashAttention to the prefill stages of both the target and draft models, significantly reducing latency and memory usage during this phase (Figure 1).

Target Model Decoding Unlike prefill, the decoding phase uses cached key and value (KV) states and computes attention only with the newly generated tokens as query. FlashDecoding (Dao, 2024) accelerates this step by parallelizing across the KV dimension, improving efficiency for short queries. However, it is incompatible with the tree masks required by tree-based speculative decoding frameworks (Miao et al., 2024). To resolve this, LongSpec (Yang et al., 2025) introduces Hybrid Tree Attention, which splits the KV cache into two parts: the cached segment that requires no masking and the speculative segment that applies the tree mask. FlashDecoding is applied to the cached segment, while standard attention is applied to the speculative part. The outputs are then fused using a log-sum-exp operation, allowing efficient tree attention computation for long inputs.

We apply Hybrid Tree Attention to the target model to accelerate its decoding phase and speed up the verification step of speculative decoding.

2.2 Cross-model Retrieval Cache

As input length increases, drafting speed degrades due to the growing KV cache of the draft model, leading to slower decoding. Meanwhile, drafting accuracy also drops as the draft model has limited capacity and is typically trained on short contexts. To address this without retraining, we aim to (1) truncate the draft model's KV cache for more efficient attention, (2) while preserving context that is most relevant to the target model at the current decoding timestep. We achieve this via cross-model retrieval, which uses the target model's attention scores to select the most relevant input segments to retain in the draft model's cache.

Concretely, we divide the input prefix into fixed-size chunks and rank them by their average attention scores, using the last accepted token as the query. These scores reflect each chunk's relevance at the current timestep. We select the top-k chunks, and the draft model uses this reduced, context-aware cache to generate candidate tokens, improving both speed and accuracy on long inputs.

Importantly, the target model's attention scores are obtained directly from the most recent verification step, requiring no additional forward passes. SpecExtend's algorithm is provided in Appendix B. One challenge is that the target model's Hybrid Tree Attention relies on FlashDecoding, which avoids generating the full attention scores matrix for efficiency. To address this, we compute standard attention and extract attention scores of the final layer only, which we find sufficient for our purposes. As shown in Table 4, this adds minimal latency overhead to the target model's forward pass,

Cache Type	Full KV Cache	StreamingLLM	Cross-model Retrieval (SpecExtend)	Retrieval (TriForce)
Perplexity (↓)	8.311	2.435	2.237	2.191
Accuracy (↑)	0.081	0.166	0.823	0.976

Table 1: Perplexity and draft accuracy of needle tokens in the Needle Retrieval task, using different draft model settings. The first three methods use Vicuna 160M as the draft model, while TriForce uses Vicuna 7B.

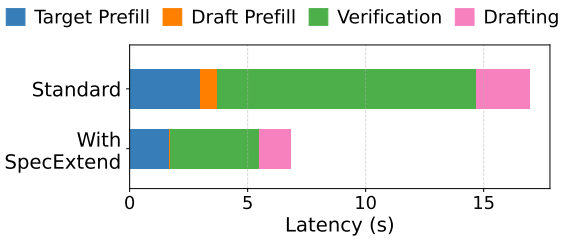

Figure 2: Latency breakdown of standard tree-based speculative decoding with and without SpecExtend.

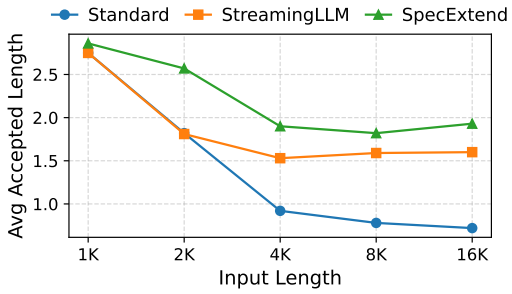

Figure 3: Average accepted length across input lengths with different draft model KV cache settings.

and the cache update step is also faster than a single draft model forward pass. Moreover, due to the locality of context in long sequences, retrieval cache updates can be applied adaptively or less frequently, further minimizing overhead. Ablations on the retrieval parameters are provided in Appendix E.1.

Needle Retrieval Evaluation We assess the effectiveness of cross-model retrieval using the Needle Retrieval task (Li et al., 2024a; Contributors, 2023). Specifically, we measure how well the draft model utilizes the retrieved context to locate and draft the tokens of a needle in long inputs. We compare its accuracy against three draft model cache types: (1) Full KV Cache, (2) StreamingLLM (Xiao et al., 2023), which retains only the initial and most recent tokens using a static cache policy, and (2) TriForce (Sun et al., 2024), which also uses the target model's attention scores to retrieve top chunks, but employs the same large model for both drafting and verification. While accurate, drafting with the target model is slow due to its large model weights. TriForce serves as an upper bound on how well the retrieved context can be utilized by a smaller draft model.

As shown in Table 1, while the StreamingLLM cache improves general coherence, it struggles to draft the needle tokens accurately due to loss of global context. In contrast, SpecExtend approaches TriForce's performance despite using a smaller draft model, simultaneously enhancing draft speed and accuracy.

3 Experiments

3.1 Experiment Setting

We evaluate SpecExtend's performance on standard speculative decoding baselines using Vicuna-7B (Chiang et al., 2023) and LongChat-7B (Li et al., 2023) as target models. Draft models include both EAGLE (Li et al., 2024c) and off-the-shelf LLMs, Vicuna-68M (Yang et al., 2024) and LLaMA-68M (Miao et al., 2023). We adopt tree-based drafting with dynamic tree expansion (Wang et al., 2025) and use long summarization task, aiming to generate 256 tokens on GovReport (Huang et al., 2021), PG-19 (Rae et al., 2019), and BookSum (Kryściński et al., 2021). All experiments are conducted with a temperature of 0 on 2 A100 80GB GPUs. Further details are provided in Appendix D.

3.2 Main Results

Figure 2 shows that SpecExtend effectively reduces inference time across all stages of speculative decoding. Meanwhile, Figure 3 shows that the cross-model retrieval cache significantly improves draft accuracy on long inputs, outperforming the static cache policy of StreamingLLM. These improvements lead to consistent speedup gains across all three datasets with both off-the-shelf LLMs and EAGLE as draft models, as shown in Table 2. We provide ablation studies on each component of SpecExtend in Appendix E.2.

For 8K and 16K-token inputs from PG-19, SpecExtend accelerates standard speculative decoding with LLM draft models by 2.37× and 2.22×, respectively, yielding overall speedups of 2.39× and 2.87× over naive autoregressive generation. For EAGLE-based frameworks, SpecExtend achieves 2.02× and 2.09× speedups over the standard EAGLE frameworks, yielding overall speedups of 2.67× and 3.09×. Importantly,

Setting			SpecExtend	1K			2K			4K			8K			16K		
				τ	Tok/s	Speedup	τ	Tok/s	Speedup	τ	Tok/s	Speedup	τ	Tok/s	Speedup	τ	Tok/s	Speedup
GovReport	V-7B	V-68M	No	1.73	100.31	1.78×	0.64	55.64	1.16×	0.60	41.91	1.14×	0.62	25.71	1.08×	0.59	16.56	1.38×
			Yes	**2.80**	**128.59**	**2.28×**	**2.52**	**109.58**	**2.29×**	**2.04**	**76.48**	**2.08×**	**2.06**	**55.16**	**2.33×**	**2.07**	**33.84**	**2.82×**
		EAGLE	No	**3.61**	144.77	2.57×	3.04	107.52	2.24×	2.27	66.62	1.81×	1.35	31.74	1.34×	1.00	19.35	1.61×
			Yes	3.58	**145.53**	**2.58×**	**3.08**	**113.47**	**2.37×**	**2.80**	**85.99**	**2.34×**	**2.82**	**62.90**	**2.66×**	**2.51**	**37.05**	**3.08×**
	LC-7B	LC-68M	No	1.73	100.31	1.78×	0.64	55.64	1.16×	0.60	41.91	1.15×	0.62	25.71	1.12×	0.59	16.56	1.51×
			Yes	**2.01**	**109.26**	**1.94×**	**1.82**	**90.27**	**1.89×**	**1.66**	**68.84**	**1.89×**	**1.81**	**52.17**	**2.30×**	**1.68**	**31.11**	**2.84×**
		EAGLE	No	**3.10**	131.06	2.33×	2.47	97.53	2.04×	1.75	60.17	1.65×	1.52	32.90	1.44×	1.18	19.84	1.81×
			Yes	3.04	**133.14**	**2.37×**	**2.56**	**103.39**	**2.17×**	**2.43**	**80.50**	**2.21×**	**2.53**	**60.14**	**2.63×**	**2.25**	**35.13**	**3.21×**
PG-19	V-7B	V-68M	No	1.16	76.50	1.37×	0.52	51.00	1.09×	0.55	39.16	1.15×	0.55	21.80	1.01×	0.54	14.73	1.29×
			Yes	**1.75**	**96.74**	**1.74×**	**1.69**	**84.74**	**1.81×**	**1.61**	**63.94**	**1.88×**	**1.65**	**47.64**	**2.39×**	**1.70**	**32.88**	**2.87×**
		EAGLE	No	**2.29**	107.31	1.92×	2.18	88.92	1.89×	1.88	54.71	1.60×	1.18	26.43	1.32×	0.92	16.98	1.48×
			Yes	2.29	**107.53**	**1.93×**	**2.19**	**94.41**	**2.02×**	**2.04**	**69.92**	**2.06×**	**2.19**	**53.06**	**2.67×**	**2.05**	**35.43**	**3.09×**
	LC-7B	LC-68M	No	1.16	76.50	1.36×	0.52	51.00	1.07×	0.55	39.16	1.09×	0.55	21.80	1.00×	0.54	14.73	1.18×
			Yes	**1.22**	**80.25**	**1.43×**	**1.33**	**73.69**	**1.55×**	**1.42**	**62.27**	**1.74×**	**1.42**	**44.96**	**2.06×**	**1.45**	**30.67**	**2.46×**
		EAGLE	No	**2.19**	111.10	1.97×	2.00	86.80	1.82×	1.48	54.21	1.51×	1.28	26.85	1.23×	1.06	17.54	1.40×
			Yes	2.11	**110.31**	**1.96×**	**2.02**	**93.50**	**1.97×**	**1.97**	**71.84**	**2.01×**	**1.99**	**51.55**	**2.36×**	**1.82**	**33.07**	**2.66×**
BookSum	V-7B	V-68M	No	1.36	88.12	1.57×	0.56	53.33	1.13×	0.51	39.30	1.08×	0.52	24.21	1.05×	0.58	15.63	1.30×
			Yes	**1.75**	**97.45**	**1.73×**	**1.66**	**81.37**	**1.73×**	**1.56**	**62.97**	**1.73×**	**1.70**	**50.21**	**2.18×**	**1.78**	**35.61**	**2.98×**
		EAGLE	No	**2.33**	111.70	1.99×	1.95	82.44	1.75×	1.87	58.01	1.59×	1.14	29.30	1.27×	0.94	18.76	1.57×
			Yes	2.31	**111.82**	**1.99×**	**1.99**	**88.64**	**1.89×**	**2.08**	**70.90**	**1.95×**	**2.15**	**54.53**	**2.37×**	**2.11**	**38.03**	**3.18×**
	LC-7B	LC-68M	No	1.36	88.12	1.57×	0.56	53.33	1.14×	0.51	39.30	1.11×	0.52	24.21	1.20×	0.58	15.63	1.28×
			Yes	**1.45**	**91.05**	**1.63×**	**1.55**	**83.60**	**1.80×**	**1.54**	**66.79**	**1.90×**	**1.61**	**49.47**	**2.45×**	**1.50**	**32.21**	**2.64×**
		EAGLE	No	**2.10**	**107.67**	**1.92×**	1.94	86.35	1.85×	1.37	53.42	1.51×	1.22	30.14	1.49×	1.06	18.39	1.50×
			Yes	2.07	106.86	1.91×	**1.97**	**90.48**	**1.94×**	**1.88**	**71.50**	**2.03×**	**1.92**	**52.35**	**2.59×**	**1.83**	**34.65**	**2.84×**

Table 2: Average accepted length (τ), decoding speed (tokens/s) and speedups of different frameworks with and without SpecExtend. Speedup is measured relative to naive autoregressive generation.

Method	GovReport					PG-19					BookSum				
	1K	2K	4K	8K	16K	1K	2K	4K	8K	16K	1K	2K	4K	8K	16K
FlashDecoding	1.06×	1.07×	1.12×	1.23×	1.51×	1.07×	1.08×	1.18×	1.38×	1.52×	1.06×	1.09×	1.10×	1.26×	1.58×
TriForce	1.25×	1.26×	1.22×	1.18×	1.02×	1.12×	1.19×	1.16×	1.15×	1.13×	1.18×	1.20×	1.18×	1.18×	1.11×
MagicDec	1.07×	1.08×	1.05×	1.13×	1.24×	1.03×	1.07×	1.06×	1.10×	1.19×	1.03×	1.04×	1.06×	1.18×	1.23×
Standard	1.78×	1.16×	1.14×	1.08×	1.38×	1.37×	1.09×	1.15×	1.09×	1.29×	1.57×	1.14×	1.08×	1.05×	1.30×
Standard + SpecExtend	**2.28×**	**2.29×**	**2.08×**	**2.29×**	**2.65×**	**1.74×**	**1.81×**	**1.88×**	**2.34×**	**2.70×**	**1.74×**	**1.74×**	**1.73×**	**2.14×**	**2.81×**

Table 3: Speedup comparison of off-the-shelf methods for long input generation with Vicuna 7B. Standard refers to standard tree-based speculative decoding.

SpecExtend preserves baseline performance on shorter inputs across all settings, demonstrating robustness to input length.

3.3 Comparison with Other Methods

We apply SpecExtend to standard tree-based speculative decoding and compare its performance on long inputs against other off-the-shelf acceleration methods, including FlashDecoding (Dao, 2024), TriForce (Sun et al., 2024), and MagicDec (Sadhukhan et al., 2024). We use Vicuna-7B, 68M as the target and draft models, respectively. For MagicDec, we implement StreamingLLM-based drafting with self-speculation. As shown in Table 3, SpecExtend-enhanced speculative decoding outperforms all baselines across input lengths, achieving up to 2.81× speedup on 16K-token inputs from BookSum. In contrast, TriForce and MagicDec yield marginal speedups, as model weights remain the dominant bottleneck in moderately long regimes, yet both rely on drafting with the large target model.

4 Conclusion

We present SpecExtend, a drop-in enhancement that improves the performance of speculative decoding frameworks on long inputs. By integrating efficient attention mechanisms and cross-model retrieval, SpecExtend accelerates all stages of speculative decoding while improving draft quality without retraining. Experiments across multiple settings and datasets demonstrate that SpecExtend achieves up to 2.22× speedup for sequences up to 16K tokens, while preserving baseline performance on shorter inputs. Our approach is compatible with a wide range of speculative decoding setups and offers a practical solution to performance degradation on long inputs.

Limitations

While SpecExtend significantly improves the speedup of speculative decoding frameworks, token generation speed still degrades as input length increases. This is primarily due to the inherent growth in attention computation, even when using efficient mechanisms. In particular, the target model's prefill and decoding steps for verification remain a bottleneck for long inputs, as SpecExtend still operates on the full KV cache for the target model's forward passes. Nevertheless, SpecExtend effectively extends the range over which speculative decoding frameworks maintain high performance. Additionally, our method does not accelerate existing frameworks to the point of outperforming those specifically trained for long inputs, such as LongSpec. Nevertheless, our proposed cross-model retrieval cache can be integrated into other solutions to provide further speedup, and SpecExtend provides substantial off-the-shelf acceleration when applied to tree-based frameworks.

Ethical Considerations

This study focuses solely on improving the inference efficiency of LLMs through a drop-in enhancement to speculative decoding. Our work does not involve training new models, collecting or annotating data, or interacting with human subjects. All experiments are conducted using publicly available models and datasets. We do not explore or enable any commercial applications or downstream use cases that raise ethical concerns. Therefore, we believe this research does not introduce any notable ethical concerns.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*.

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

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Tri Dao. 2024. Flash decoding. https://princeton-nlp.github.io/flash-decoding/. Accessed: 2024-05-16.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization. *Preprint*, arXiv:2104.02112.

Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. 2021. Booksum: A collection of datasets for long-form narrative summarization. *arXiv preprint arXiv:2105.08209*.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph E. Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. 2023. How long can open-source llms truly promise on context length? https://lmsys.org/blog/2023-06-29-longchat.

Mo Li, Songyang Zhang, Yunxin Liu, and Kai Chen. 2024a. Needlebench: Can llms do retrieval and reasoning in 1 million context window? *arXiv preprint arXiv:2407.11963*.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024b. EAGLE-2: Faster inference of language models with dynamic draft trees. In *Empirical Methods in Natural Language Processing*.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024c. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *International Conference on Machine Learning*.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2025. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *arXiv preprint arXiv:2503.01840*.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Rae Ying Yee Wong, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. 2023. Specinfer: Accelerating generative llm serving with speculative inference and token tree verification. *Preprint*, arXiv:2305.09781.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, and 1 others. 2024. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pages 932–949.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, Chloe Hillier, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. *arXiv preprint*.

Ranajoy Sadhukhan, Jian Chen, Zhuoming Chen, Vashisth Tiwari, Ruihang Lai, Jinyuan Shi, Ian En-Hsu Yen, Avner May, Tianqi Chen, and Beidi Chen. 2024. Magicdec: Breaking the latency-throughput tradeoff for long context generation with speculative decoding. *arXiv preprint arXiv:2408.11049*.

Hanshi Sun, Zhuoming Chen, Xinyu Yang, Yuandong Tian, and Beidi Chen. 2024. Triforce: Lossless acceleration of long sequence generation with hierarchical speculative decoding. *arXiv preprint arXiv:2404.11912*.

Jikai Wang, Yi Su, Juntao Li, Qingrong Xia, Zi Ye, Xinyu Duan, Zhefeng Wang, and Min Zhang. 2025. Opt-tree: Speculative decoding with adaptive draft tree structure. *Transactions of the Association for Computational Linguistics*, 13:188–199.

Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. 2022. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. *arXiv preprint arXiv:2203.16487*.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. 2024. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. *arXiv preprint arXiv:2401.07851*.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*.

Penghui Yang, Cunxiao Du, Fengzhuo Zhang, Haonan Wang, Tianyu Pang, Chao Du, and Bo An. 2025. Longspec: Long-context speculative decoding with efficient drafting and verification. *arXiv preprint arXiv:2502.17421*.

Sen Yang, Shujian Huang, Xinyu Dai, and Jiajun Chen. 2024. Multi-candidate speculative decoding. *arXiv preprint arXiv:2401.06706*.

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, and 1 others. 2024. A survey on efficient inference for large language models. *arXiv preprint arXiv:2404.14294*.

A Related Work

A.1 Speculative Decoding

Speculative decoding accelerates LLM inference by using a smaller draft model to generate multiple candidate tokens, which the target model then verifies in parallel (Xia et al., 2022, 2024). With proper verification and correction, it guarantees the same output distribution as standard decoding (Leviathan et al., 2023; Chen et al., 2023). SpecInfer (Miao et al., 2024) extends this approach by drafting and verifying multiple sequences simultaneously using tree attention, achieving further speedups. Several work introduces effective draft models built from subsets of the target model (Cai et al., 2024; Li et al., 2024c), while EAGLE-2 (Li et al., 2024b) and OPT-Tree (Wang et al., 2025) achieve further speedup by dynamically adjusting the draft tree structure during decoding. EAGLE-3 (Li et al., 2025) scales up draft model training by leveraging multi-level features from the target model.

A.2 Long Sequence Generation

As input length increases, standard attention suffers from quadratic compute and memory complexity, causing high inference latency (Zhou et al., 2024). FlashAttention (Dao et al., 2022; Dao, 2023) reduces this overhead by using tiling and online softmax, bringing memory complexity down to linear and accelerating inference. FlashDecoding (Dao, 2024) builds on this by further parallelizing workers across the Key Value dimension, speeding up LLM decoding for long sequences.

Several works apply speculative decoding to long sequence generation. TriForce (Sun et al., 2024) identifies that the memory bottleneck shifts from model weights to the KV cache for extremely

long inputs, and mitigates this with hierarchical speculation using smaller models and retrieval-based KV caches. MagicDec (Sadhukhan et al., 2024) uses a simpler StreamingLLM cache to reduce the KV cache memory. However, the performance of speculative decoding frameworks drop well before the KV cache becomes the main bottleneck, and existing solutions are less effective in this regime of early degradation. Closest to our approach is LongSpec (Yang et al., 2025), which trains draft models specifically designed for long inputs. In contrast, our method provides a drop-in enhancement for existing frameworks, improving long-sequence performance without retraining and preserving their original benefits.

B SpecExtend Algorithm

Algorithm 1 Speculative Decoding with Cross-model Retrieval Cache

Require: Target LM M_q, draft LM M_p, input $x_1, \ldots, x_t$, block size K, target length T, DRAFT, VERIFY, CORRECT, retrieval flag `doRetrieval`, attention scores s, top-k chunks $c_1, \ldots, c_k$

$n \leftarrow t$
while $n < T$ **do**
  ▷ Retrieve and update draft-model cache
  if `doRetrieval` **then**
    $c_1, \ldots, c_k \leftarrow$ SELECTCHUNKS(s)
    UPDATEDRAFTCACHE$(c_1, \ldots, c_k)$
  end if
  $p_1, \ldots, p_K \leftarrow$ DRAFT$(x_{\leq n}, M_p)$
  Sample $\tilde{x}_i \sim p_i$ for $i = 1, \ldots, K$
  ▷ Obtain target model attention scores
  $(q_i,\ s) \leftarrow M_q\big(x \mid x_{\leq n}, \tilde{x}_{<i}\ ;\ \texttt{doRetrieval}\big)$ for $i = 1, \ldots, K+1$
  if VERIFY$(\tilde{x}_i, p_i, q_i)$ **then**
    $x_{n+1} \leftarrow \tilde{x}_i$; $n \leftarrow n + 1$
  else
    $x_{n+1} \leftarrow$ CORRECT(p_i, q_i)
    break
  end if
  if all K drafted tokens accepted **then**
    Sample $x_{n+1} \sim q_{K+1}$; $n \leftarrow n + 1$
  end if
end while

C Latency Overhead of SpecExtend

Table 4 shows the latency overhead of SpecExtend's cross-model retrieval cache. When using 16K inputs, the target model's forward pass shows minimal overhead with and without retrieval, as we compute standard attention for the last layer only. In addition, a retrieval cache update step which requires computing the average attention scores of chunks, ranking them and updating the draft model's KV cache with the top-k chunks, can be done faster than a single forward pass of the draft model.

	Target Forward	Target Forward w/ Retrieval	Draft Forward	Retrieval Cache Update
Latency (ms)	53.76	54.11	0.84	0.34

Table 4: Latency overhead of a single retrieval cache update step on 16K token inputs.

D Experiment Details

The EAGLE models[1] are trained on the ShareGPT dataset using default training settings with 4 A100 40GB GPUs. For each input length from 1K to 16K tokens, we sample 20 inputs, run each input twice, and report metrics averaged over all runs. We set the temperature to 0 and the maximum generation length to 256 tokens. We apply OPT-Tree's dynamic tree expansion strategy with the default settings of 50 total nodes, maximum depth 10, and threshold 0.7. We use the optimal working KV cache size and retrieval parameters described in Appendix E.1.

E Ablation Studies

E.1 Retrieval Parameters

We conduct ablations on retrieval parameters using Vicuna-7B as the target model, with Vicuna-68M and EAGLE as draft models, on 8K-token inputs from GovReport (Table 5). When varying the StreamingLLM cache size, the optimal working cache size is around 1K for Vicuna-68M and 2K for EAGLE. This difference is due to the disparity in draft model scale (68M vs. 1B parameters). Based on these cache sizes, we find that for Vicuna-68M and EAGLE respectively, a chunk size of 32, top-k values of 32 and 64, and a retrieval frequency of 4 and 8 steps yield the best performance. We note that these optimal settings can vary with the architecture and capacity of the draft model.

E.2 SpecExtend Components

We evaluate the relative speedup of each component of SpecExtend, applying them to a standard tree-based framework with Vicuna-7B and Vicuna-68M. All speedups are measured relative to the standard speculative decoding framework. As shown in Table 6, applying FlashAttention to the

[1] EAGLE models are publicly available under the Apache 2.0 license.

Working Cache Size	Vicuna-68M	EAGLE	Chunk Size	Vicuna-68M	EAGLE	Top-k	Vicuna-68M	EAGLE	Retrieval Frequency	Vicuna-68M	EAGLE
64	32.52	39.10	1	31.05	48.05	2	30.72	38.22	1	33.05	47.78
128	32.91	39.95	2	32.27	49.49	4	32.65	40.36	2	33.54	46.78
256	33.65	41.53	4	32.97	49.55	8	32.76	41.49	4	**33.59**	48.17
512	33.53	42.77	8	33.39	49.18	16	33.19	43.90	8	33.11	**48.52**
1024	**33.69**	44.19	16	33.41	48.92	32	**33.28**	47.21	16	33.16	48.36
2048	32.36	**45.33**	32	**33.52**	**49.68**	64	32.50	**48.09**	32	33.28	48.11
4096	25.84	43.68	64	33.23	48.25	128	25.20	45.14	64	33.29	48.13
8192	24.32	33.10	128	33.20	47.48	256	23.95	32.48	128	33.21	48.20

Table 5: Decoding speed (tokens/s) for different working KV cache size and retrieval parameters. We use Vicuna-7B as the target model and 8K inputs from GovReport.

Setting	1K			2K			4K			8K			16K		
	τ	Tok/s	Speedup	τ	Tok/s	Speedup	τ	Tok/s	Speedup	τ	Tok/s	Speedup	τ	Tok/s	Speedup
Standard	2.75	127.34	-	1.83	87.34	-	0.92	47.41	-	0.78	27.54	-	0.72	17.60	-
Standard + FA	2.71	131.02	1.03×	1.84	91.79	1.05×	0.97	52.74	1.11×	0.81	34.33	1.25×	0.75	22.07	1.25×
Standard + HTA	2.61	122.73	0.96×	1.74	85.57	0.98×	0.92	47.62	1.01×	0.76	31.08	1.14×	0.74	20.95	1.19×
Standard + StreamingLLM	2.75	128.62	1.01×	1.81	85.60	0.98×	1.53	59.11	1.25×	1.59	35.89	1.30×	1.60	22.39	1.27×
Standard + Retrieval	2.86	130.35	1.02×	2.57	104.12	1.19×	1.90	64.85	1.36×	1.78	37.11	1.47×	1.93	25.82	1.46×

Table 6: Ablation on the components of SpecExtend. Standard indicates the standard tree-based speculative decoding with Vicuna 7B/68M. FA and HTA indicate FlashAttention for prefill and Hybrid Tree Attention, respectively. Standard + Retrieval refers to using SpecExtend's cross-model retrieval cache. Speedups are measured relative to the standard framework.

prefill stages yields a 1.25× speedup for 16K inputs. Hybrid Tree Attention introduces slight overhead at shorter lengths but achieves up to 1.19× speedup beyond 8K tokens. Therefore we enable Hybrid Tree Attention only for inputs beyond 4K tokens. The cross-model retrieval cache alone provides up to a 1.46× speedup over the standard setting.