LIGHT WINGS: MORE EFFICIENT SPECULATIVE DECODING VIA LINEAR ATTENTION

Jiaping Wang[1] *Yifan Xu*[2] *Yifan Chen*[2 †] *Jianwen Li*[1 †]

[1] East China Normal University [2] Hong Kong Baptist University

ABSTRACT

Speculative decoding (SD) accelerates large language model (LLM) inference through introducing a fast draft model to propose candidate token sequences. However, conventional draft models employ softmax-based attention for capturing original transformers, incurring quadratic time and memory complexity during prefilling. Given the limited resources assigned to drafters, this design restricts their scalability in long-context scenarios. To this end, we propose a new drafter architecture built on *linear attention*, which achieves linear computational complexity. Specifically, our method maintains a fixed-size key-value cache during autoregressive generation, enabling efficient, scalable inference with minimal overhead. We evaluated the proposed approach on multiple model scales and benchmarks, demonstrating comparable token acceptance rates and downstream performance to standard attention mechanisms, while achieving an average $2.62\times$ speed-up. The implementations are provided in [1].

***Index Terms*—** Speculative decoding, Linear attention, Efficient inference

I. INTRODUCTION

Large generative pretrained transformer (GPT) language models [2], [3], [4] have become central to modern natural language processing, driving performance in various downstream tasks. However, their deployment in real-time or resource-constrained applications is hindered by autoregressive generation, resulting in high inference latency and memory overhead.

Speculative decoding [5], [6] therefore emerges to accelerate LLM inference. The process consists of two phases: (1) the *drafting* phase, in which the draft models propose future tokens given current context, and (2) the *verification* phase, where the target models check the proposals through a forward pass and accept or reject them based on the target probabilities. Introducing parallelism across token proposals and verifications, speculative decoding amortizes the cost of expensive LLM decoding. Prior efforts fall into three categories: (i) training compact draft models via knowledge distillation, such as Eagle series [7], (ii) integrating auxiliary prediction heads (e.g. Medusa [8]) to forecast future tokens, and (iii) leveraging model-free or structural variants such as n-gram matching [9], lookahead decoding [10] and other methods [11].

The overall efficiency of speculative decoding hinges on **token acceptance rates** (model alignment) as well as **efficiency of draft model decoding**. However, the latter is largely overlooked: most draft models retain standard softmax attention, which thus incurs $\mathcal{O}(N^2)$ time and memory cost during the prefilling phase for the length-N context. In workloads with long inputs but short outputs, such as summarization or retrieval-augmented generation, the extra drafter prefilling computation can overshadow the latency benefits of speculative decoding, as limited resources can be assigned to token proposal on top of expensive target model verification.

To overcome this limitation, we keep drawing inspiration from mainstream works to adopt transformer architectures for alignment with target transformer LLMs, while equipping transformer blocks with linear attention [12], [13] to improve alignment–efficiency trade-offs. Our design achieves $O(N)$ time and space complexity, enabling scalable and memory-efficient processing of long sequences. Crucially, it supports a fixed-size key-value (KV) cache during autoregressive generation, eliminating dynamic cache growth and further reducing memory footprint and access latency. We show that linear attention can preserve strong alignment with target transformer models, while maintaining high token acceptance rates. Our contributions are summarized as follows:

- We introduce linear attention into speculative decoding and propose a new draft model architecture, achieving linear complexity and a constant-size KV cache.
- We provide comprehensive empirical validation (across two model scales, five benchmarks, and two large-scale datasets), showing that linear attention exhibits acceptance rates and downstream performance similar to standard attention, while achieving an average $2.62\times$ speedup for long sequence generation.

II. PRELIMINARIES AND BACKGROUND

Linear Attention. Various linear attention mechanisms have been proposed to reduce the inference cost from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. Among these, kernel-based approaches [14], [15],

† Correspondence to: Yifan Chen yifanc@hkbu.edu.hk, Jianwen Li jwli@sei.ecnu.edu.cn.

[16] reformulate softmax operation by approximating attention score matrices with associative rules, thereby allowing for linear-time sequence modeling.

In particular, after ignoring normalizing factors and the activation functions, the simplified causal linear attention is as follows:

$$\boldsymbol{O} = \left([\boldsymbol{Q}\boldsymbol{K}^\top] \odot \boldsymbol{M}\right)\boldsymbol{V}. \tag{1}$$

Here, $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ are the query, key, and value matrices; $\boldsymbol{M}$ is a causal lower-triangular mask that ensures autoregressive generation. The intrinsic complexity of eq. (1) remains linear even in the presence of $\boldsymbol{M}$, and optimized GPU implementations have been developed to achieve efficient linear-time execution [17], [18]. These advances have fostered the development of scalable sequence models based on linear or hybrid attention architectures, such as TransNormerLLM [13], and Minimax-M1 [19], which demonstrate competitive performance with standard transformers while offering superior efficiency for long-sequence modeling.

Speculative Decoding (SD). Speculative decoding is a technique for accelerating large language model (LLM) inference. The key idea of speculative decoding is to employ a lightweight draft model to generate candidate token sequences, which are verified in parallel by the large target model within a single forward pass. The acceptance or rejection of tokens is governed by the consistency between the output distributions of the draft and the target models, ensuring that the resulting generation process remains exactly aligned with the one of the target model.

More concretely, rejection sampling is adopted for verification. For a token $\boldsymbol{x}$ generated from the drafter, we let $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$ denote the probabilities assigned by the target and draft models, respectively, and let $r \sim \mathcal{U}(0,1)$ be a uniform random variable. Token $\boldsymbol{x}$ will be accepted if $r < p(\boldsymbol{x})/q(\boldsymbol{x})$, and rejected otherwise. In the case of rejection, resampling is performed from an adjusted distribution $p'(\boldsymbol{x}) \propto \max\{0, p(\boldsymbol{x}) - q(\boldsymbol{x})\}$ for all tokens in the vocabulary. The distribution p' re-normalizes the difference between target and draft probabilities, thereby preserving exactness of the target distribution.

Overall, by amortizing the cost of verifying multiple draft tokens within a single target model forward pass, speculative decoding can substantially reduce inference latency, achieving significant speedup when the draft model is both computationally efficient and well-aligned with the large target model.

III. METHOD

In this section, we present our integration of linear attention into the Eagle3 model (Section III-A) and analyze its time and space complexity in different stages of speculative decoding (Sections III-B and III-C). We focus on the "draft" phase of speculative decoding, and decompose draft model attention computation in this phase into three parts:

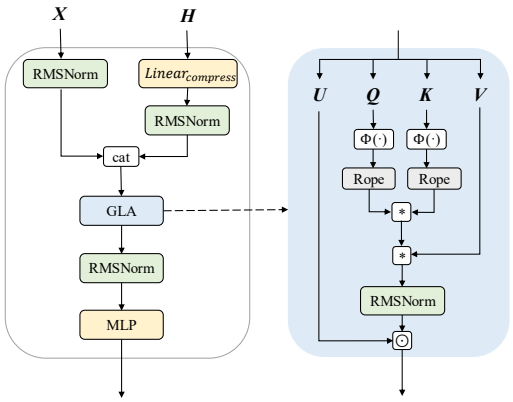

Fig. 1. Architecture of a draft model equipped with linear attention. The right panel shows the Gated Linear Attention.

i *Prefilling*: Processes the entire input prompt to produce the initial KV cache for subsequent generation.
ii *Decoding*: Generates tokens autoregressively using the existing KV cache, updating it with the new token.
iii *Post-Verification Prefilling*: After the target model verifies and accepts the n tokens from the draft model, these accepted tokens are fed back into the draft model to update its KV cache in a single pass.

These stages share the same core attention formulation, but differ in input length and KV cache state, directly affecting their computational and memory characteristics.

III-A. Equip Draft Model with Linear Attention

For the draft model in speculative decoding, although some preprints proposed to leverage exotic linear-time sequence modeling architectures [20], we adopt the classical transformer-based design of Eagle3 [7], which aligns well with the target transformer. Specifically, we introduce a single-layer transformer block with Gated Linear Attention (GLA) [21]. Its input combines the initial embeddings with concatenated hidden states from the second, middle, and second to last layers of the target model (see Figure 1). For clarity, RMSNorm is omitted in the following descriptions, as it does not affect the core mechanism or complexity.

Let the tensor $\boldsymbol{X} \in \mathbb{R}^{B\times N\times D}$ denote the input embeddings of the target model, where B, N, and D are the batch size, the sequence length, and the embedding (hidden) dimension, respectively, with $D \ll N$. For the concatenated hidden states of the low, medium, and high layers $\boldsymbol{H} \in \mathbb{R}^{B\times N\times 3D}$, we first compress them through a fully connected linear layer:

$$\boldsymbol{H}_c = \mathrm{Linear}_{\mathrm{compress}}(\boldsymbol{H}) \in \mathbb{R}^{B\times N\times D},$$

and then further concatenate the compressed embedding $\boldsymbol{H}_c$ with the original $\boldsymbol{X}$ to obtain the intermediate output:

$$\boldsymbol{I} = \mathrm{cat}(\boldsymbol{H}_c, \boldsymbol{X}, \mathrm{dim} = -1) \in \mathbb{R}^{B\times N\times 2D}. \tag{2}$$

Unlike conventional attention mechanisms, our linear variant projects $\boldsymbol{I} \in \mathbb{R}^{B\times N\times D}$ via two transformations:

$$\mathrm{Linear}_{\mathrm{qu}}(\boldsymbol{I}) \in \mathbb{R}^{B\times N\times 2D_q}, \quad \mathrm{Linear}_{\mathrm{kv}}(\boldsymbol{I}) \in \mathbb{R}^{B\times N\times 2D_v},$$

yields concatenated $(\boldsymbol{Q}, \boldsymbol{U}) \in \mathbb{R}^{B\times N\times D_q}$ and $(\boldsymbol{K}, \boldsymbol{V}) \in \mathbb{R}^{B\times N\times D_v}$, where D_q usually equals the embedding dimension D, D_v is the hidden dimension of the key / value matrices (they share the same dimension D_v), and the tensor $\boldsymbol{U}$ will shortly serve as gated unit matrices.

We adopt the grouped query attention (GQA) mechanism, allowing multiple attention heads to share the same key/value matrices. Furthermore, we apply the SiLU activation to the query and key projections to enhance the model's expressiveness, and employ rotary position embedding (RoPE) [22] to inject positional information into $\boldsymbol{Q}$ and $\boldsymbol{K}$.

With these enhancements and as defined in eq. (1), linear attention follows eq. (3), avoiding the full $N \times N$ attention matrix in *Prefilling* via specialized $\mathcal{O}(N)$ methods [17], such as the recurrence scan below:

$$\boldsymbol{O} = \sum_{i=1}^{D} \mathrm{diag}(\boldsymbol{Q}_i) \cdot \mathrm{cumsum}\big(\mathrm{diag}(\boldsymbol{K}_i)\boldsymbol{V}\big), \tag{3}$$

where $\boldsymbol{Q}_i, \boldsymbol{K}_i$ denote the i-th rows of $\phi(\boldsymbol{Q}), \phi(\boldsymbol{K})$, respectively ($\phi(\cdot)$ is a transform map specified by the linear attention mechanism). Here, $\mathrm{diag}(\boldsymbol{X})$ constructs a diagonal matrix from vector $\boldsymbol{X}$, and $\mathrm{cumsum}(\cdot)$ computes the cumulative sum along the sequence dimension.

Lastly, we introduce a gating mechanism through the aforementioned gated unit matrix $\boldsymbol{U}$ and output $\boldsymbol{U} \odot \boldsymbol{O}$. This design enhances non-linearity and enables adaptive feature selection, further improving the capacity of the draft model to generate candidates of high acceptance.

III-B. KV Cache Optimization in Decoding

In LLM inference, a standard transformer block stores the entire sequence of keys and values (the so-called KV cache) after the *Prefilling* stage to avoid repeated re-calculation of $\boldsymbol{K}, \boldsymbol{V}$ for recurring new queries $\boldsymbol{Q}_t$, leading to $\mathcal{O}(ND)$ memory usage for context length N and hidden size D. In contrast, the linear attention mechanism admits a more memory-efficient representation: interactions between keys and values can be summarized by a fixed-size cumulative matrix $\boldsymbol{kv} = \boldsymbol{K}^\top \boldsymbol{V} \in \mathbb{R}^{D_v\times D_v}$, which greatly reduces GPU memory usage.

We showcase the space efficiency of the proposed draft models with linear attention at different stages as follows. First of all, in the initial *Prefilling* stage, the draft model processes a full prompt of length N and doesn't utilize KV cache (c.f. eq. (3)), while the model will store the $D_v \times D_v$ product $\boldsymbol{kv} = \boldsymbol{K}^\top \boldsymbol{V}$ as the KV cache for future use. During *Decoding*, the draft model sets the temporary KV cache $\boldsymbol{T} = \boldsymbol{kv}$ and tentatively generates n tokens $\boldsymbol{O}_t$'s via:

$$\boldsymbol{T} = \boldsymbol{T} + \boldsymbol{K}_t \boldsymbol{V}_t^\top, \qquad \boldsymbol{O}_t = \boldsymbol{Q}_t^\top \boldsymbol{T}, \tag{4}$$

Table I. Time Complexity of draft models equipped with vanilla attention and linear attention in different stages. Here, n represents the number of tokens in drafting accepted by the target model, usually a small constant.

Stage	Vanilla Attn.	Linear Attn.
Prefilling	$\mathcal{O}(N^2D)$	$\mathcal{O}(ND^2)$
Decoding	$\mathcal{O}(ND)$	$\mathcal{O}(D^2)$
Post-Verification Prefilling	$\mathcal{O}(nND)$	$\mathcal{O}(n^2D + nD^2)$

for any $t \in [N+1, N+n]$. In the *Post-Verification Prefilling* stage, after the target model verifies and accepts n draft tokens, these tokens are fed back into the draft model to update its state in a single pass (eq. (5)).

$$\boldsymbol{O} = \boldsymbol{Q}_\Delta \boldsymbol{K}_\Delta^\top \odot \boldsymbol{M}\, \boldsymbol{V}_\Delta + \boldsymbol{Q}_\Delta\, \boldsymbol{kv}, \tag{5}$$

for the period $\Delta = [N : N+n]$. In summary, the procedure enables *constant-memory* inference in the asymptotic sense, substantially reducing the memory footprint for generation in long-context scenarios and improving the scalability of the draft model within speculative decoding.

III-C. Time Complexity Analysis

We present a stage-wise time complexity comparison in Table I between standard and linear attention in the draft model, following the three stages in the "drafting" phase. In the *Prefilling* stage, linear attention reduces the computational complexity from $\mathcal{O}(N^2D)$ to $\mathcal{O}(ND^2)$. In the *Decoding* stage, the time cost decreases from $\mathcal{O}(ND)$ to $\mathcal{O}(D^2)$. Finally, in the *Post-Verification Prefilling* stage, the complexity drops from $\mathcal{O}(nND)$ to $\mathcal{O}(n^2D + nD^2)$, where we recall that n is the number of tokens accepted and usually a small constant.

This quadratic-to-linear reduction yields considerable inference speedup, especially for long contexts where the overall speedup ratio scales proportionally with N. This shows that linear attention not only offers significant memory savings (as shown in Section III-B) but also delivers substantial end-to-end latency reduction for SD.

IV. EXPERIMENTS

To provide a comprehensive assessment, we first evaluate our linear-attention drafters on benchmarks with short-to-medium sequence lengths. Though linear attention is not intended for these regimes due to overhead constraints, these experiments establish that our method performs at par with standard softmax alternatives even with limited sequence length. We then turn to Section IV-B, where we investigate long-context scenarios (our focus) and verify the significant time and space efficiencies afforded by our proposals.

IV-A. Sanity Check: Evaluation on Standard Benchmarks with Short to Medium-Length Sequences

Drafter Training. Adopting the training protocol from Eagle3, we utilized a subset of the UltraChat dataset [23].

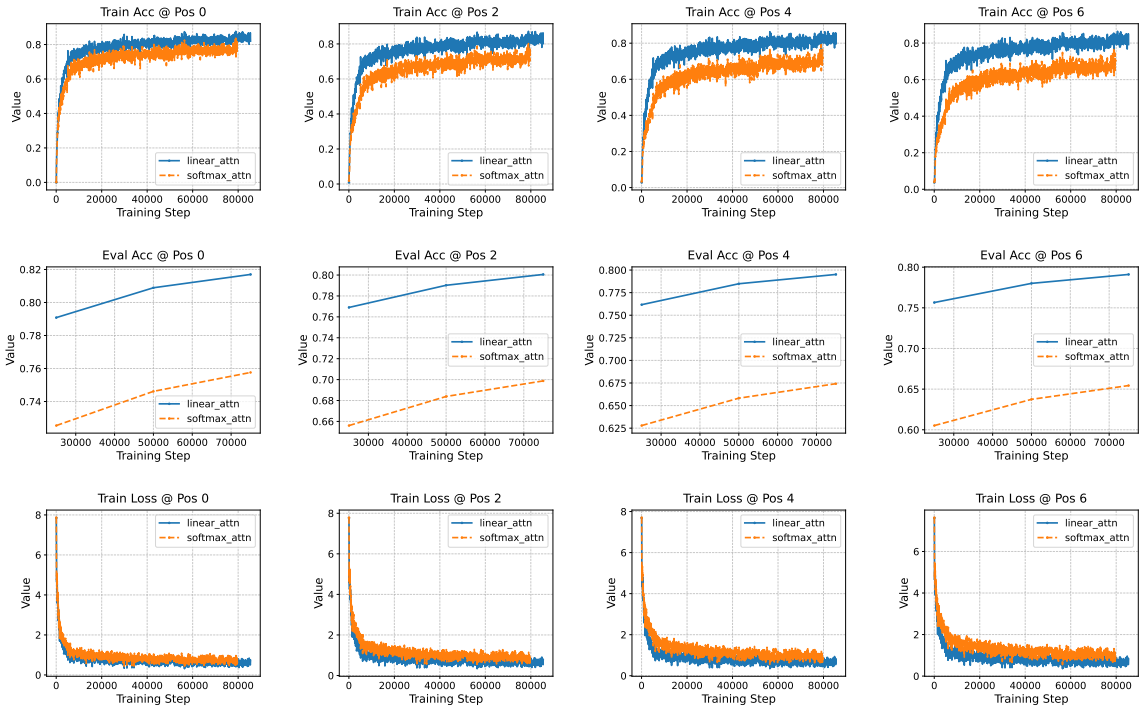

Fig. 2. Comparison of drafters employing linear attention and classical softmax attention, trained on the full ShareGPT dataset for Qwen3-8B. We track the drafter prediction accuracy at positions 0,2,4 and 6 (the 1st, the 3rd, the 5th and the 7th tokens generated by the drafter in autoregression) on both the training and evaluation sets of the ShareGPT dataset. The linear-attention drafter exhibits stable training and lower loss. Besides, at the same training step, it achieves an even higher acceptance rate on multi positions in both training set and evaluation set.

Specifically, we curated a training set of $20,916$ dialogues, selecting only those where the initial prompt exceeds 512 tokens. Training was conducted on four NVIDIA RTX 4090 GPUs. We trained drafters for Qwen2.5-0.5B-Instruct and Qwen3-8B using a maximum sequence length of 2K tokens. To assess the impact of the training set, we additionally trained Qwen3-8B drafters with both standard and linear attention on the full ShareGPT dataset [24].

Analysis. We compared the training and evaluation accuracy of the Qwen3-8B draft model trained on the full ShareGPT dataset, employing linear attention or standard softmax attention at token positions 0, 2, 4, and 6, respectively. We also analyzed the training loss at these corresponding positions. The results, illustrated in Figure 2, demonstrate that the draft model utilizing linear attention achieves a higher accuracy than the softmax baseline for the same number of training steps, on both training and evaluation sets. Furthermore, similar to softmax attention, the training process using linear attention remains highly stable and exhibits robust convergence and achieves a lower training loss.

Evaluation. We first examined the run time and corresponding accept length via an A800 GPU on five benchmarks: MT-Bench [25], HumanEval [26], Alpaca [27], XSum [28], and WikiQA [29] (over HumanEval we adopt 5-shot accuracy).

The performance of linear-attention drafters under **short to medium-length sequences** is summarized in Table II. (1) Overall speedup ratios for standard and linear attention are comparable, despite their different average accept lengths across datasets. For the Qwen3-8B model trained over UltraChat, both architectures produce identical speedup ratios in HumanEval, despite the slightly reduced accept length of the linear-attention drafter. In the case of the ShareGPT-trained drafters, linear attention instead attains a higher speedup ratio despite a shorter accept length. This dataset has a relatively longer input length, and the linear-attention drafters thus enjoy faster inference than the standard attention; we can find a more significant efficiency improvement in the next experiment on long sequences. (2) In addition, the performance of using Qwen3-0.6B as the draft model for Qwen3-8B (Sps) is even worse than the baseline due to the insufficient depth gap between the draft and target models.

IV-B. Long-Context Scenarios

We continue to investigate long-context scenarios and evaluate the computational efficiency of the draft model equipped with standard softmax or linear attention trained in the same setting as Section IV-A. Two complementary evaluations are conducted on a single A800 GPU: (*i*) a micro-benchmark measuring inference memory footprint and

Table II. Speedup ratio (**SR**) and average accept length (**Acc**) for Qwen2.5-0.5B-Instruct and Qwen3-8B under autoregressive, Eagle3 (softmax attention), and Eagle3 (linear attention) decoding. **Sps** denotes speculative decoding for Qwen3-8B using Qwen3-0.6B as draft (not provided for Qwen2.5-0.5B-Instruct because it's the smallest Qwen2.5 model). **Full** denotes the draft model trained using the complete dataset. Although the performance of draft models whether based on linear or classic softmax attention fluctuates across individual benchmarks in terms of accept length and speedup ratio, the linear attention based drafter achieves a superior average performance, characterized by marginally higher accept lengths and speedup ratios.

Model	Method	MT-Bench		HumanEval		Alpaca		XSum		WikiQA		AVG	
		SR	Acc	SR	Acc	SR	Acc	SR	Acc	SR	Acc	SR	Acc
Qwen2.5-0.5B	Vanilla	1×	-	1×	-	1×	-	1×	-	1×	-	1×	-
	Eagle3-SoftAttn	1.38×	1.75	1.17×	1.37	1.43×	1.61	1.38×	1.44	1.55×	1.57	1.38×	1.55
	Eagle3-LinearAttn	1.28×	1.55	1.16×	1.34	1.51×	1.69	1.44×	1.48	1.78×	1.77	1.43×	1.57
Qwen3-8B	Vanilla	1×	-	1×	-	1×	-	1×	-	1×	-	1×	-
	Sps	0.48×	-	0.50×	-	0.42×	-	0.40×	-	0.45×	-	0.46×	-
	Eagle3-SoftAttn	1.50×	2.01	1.58×	1.73	1.60×	1.89	1.61×	1.83	1.74×	1.84	1.61×	1.86
	Eagle3-LinearAttn	1.42×	1.83	1.58×	1.71	1.67×	1.91	1.66×	1.83	1.79×	1.85	1.62×	1.83
	Eagle3-SoftAttn-Full	2.05×	2.70	2.40 ×	2.98	1.97×	2.45	1.89×	2.27	2.26×	2.70	2.11×	2.62
	Eagle3-LinearAttn-Full	1.98 ×	2.59	2.42 ×	2.95	2.04 ×	2.54	2.00 ×	2.35	2.28 ×	2.72	2.14 ×	2.63

Table III. The run time and GPU memory cost for the Qwen3-8B draft model to generate 5 tokens autoregressively, which corresponds to the total time of the draft model's prefilling operation and 4 decoding operations using linear attention and softmax attention under different input lengths. The computational overhead of the linear attention drafter exhibits near-linear scaling with respect to sequence length.

Input length	Softmax Attn.		Linear Attn.	
	Mem(G)	Time(s)	Mem(G)	Time(s)
1024	4.34	0.06	4.41	0.06
2048	5.32	0.13	4.82	0.11
4096	8.77	0.22	5.60	0.21
8192	21.79	0.47	7.20	0.37
16384	73.09	1.22	10.38	0.69
32768	**OOM**	-	12.50	1.37

Table IV. The GPU memory cost and speedup ratio tested on LongBench for Qwen3-8B with draft models trained on UltraChat and ShareGPT using linear attention and softmax attention. **Full** denotes the draft model trained on the complete ShareGPT dataset. Drafters with linear attention can reduce GPU memory from 77G to 33G and gain 1.22× speedup ratio at most.

Method	HotpotQA		Repobench	
	Mem(G)	SR	Mem(G)	SR
vanilla	31	1×	31	1×
SoftAttn	77	1.07×	77	0.95×
LinearAttn	33	1.09×	33	1.05×
SoftAttn-Full	77	1.15×	77	1.10×
LinearAttn-Full	33	1.19 ×	33	1.22×

latency across various input lengths, and (*ii*) end-to-end performance on long-context, short-output tasks.

In (*i*) the micro-benchmark, the draft models of Qwen3-8B in Section IV-A generate 5 tokens given input sequences ranging from 1K to 32K tokens, and we measure their corresponding GPU memory footprint and inference latency. As shown in Table III, softmax attention exhibits quadratic growth in both memory usage and latency, with an out-of-memory (OOM) error occurring at 32K tokens. In contrast, linear attention maintains a constant-size KV cache, with overall memory and time scaling linearly with input length. For example, at 16K length, linear attention reduces memory usage from 73.09 GB to 10.38 GB and latency from 1.22 s to 0.69 s. We remark that the slightly higher memory usage at 1K length for linear attention is due to the additional gating parameter $\boldsymbol{U}$ in its formulation.

For (*ii*) long-context, short-output workloads, we evaluate the Qwen3-8B backbone model on two LongBench [30] tasks, HotpotQA and RepoBench, whose average sequence lengths are over 4K. As shown in Table IV, the application of linear attention reduces GPU memory usage from 77 GB to 33 GB while delivering consistent speed improvements of 1.19 × in HotpotQA and 1.22 × in RepoBench. Notably, in such scenarios, draft models employing softmax attention may yield negative net gains, as the substantial computational overhead of the prefilling stage often outweighs the latency benefits provided by speculative decoding.

V. CONCLUSION

We study linear attention as a drafting mechanism for speculative decoding, addressing the scalability limitations of softmax-based drafters in long-context scenarios. By using linear attention, we reduce complexity from quadratic to linear and enable a constant-size key-value cache. Integration into the Eagle3 framework confirms substantial reductions in memory and computational cost while maintaining high token acceptance rates of standard softmax mechanism. Achieving average speed-ups of 2.62× in text generation and 1.22× in long contexts, our work establishes a scalable paradigm for efficient large language model deployment in real-time applications.

ACKNOWLEDGMENTS

Yifan Chen acknowledges funding from National Natural Science Foundation of China (NSFC) under grant 62502408, Research Grants Council (RGC) under grant 22303424, GuangDong Basic and Applied Basic Research Foundation under grant 2025A1515010259, and Guangdong and Hong Kong Universities "1+1+1" Joint Research Collaboration Scheme under grant 2025A050500.

VI. REFERENCES

[1] "Repo for model implementation," https://github.com/jiapingW/LIGHTWINGS.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, and Prafulla Dhariwal et al., "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[4] 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.

[5] Yaniv Leviathan, Matan Kalman, and Yossi Matias, "Fast inference from transformers via speculative decoding," in *International Conference on Machine Learning*. PMLR, 2023, pp. 19274–19286.

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

[7] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang, "EAGLE-3: Scaling up inference acceleration of large language models via training-time test," 2025.

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

[9] Jie Ou, Yueming Chen, and Wenhong Tian, "Lossless acceleration of large language model via adaptive n-gram parallel decoding," 2024.

[10] Yao Zhao, Zhitian Xie, Chen Liang, Chenyi Zhuang, and Jinjie Gu, "Lookahead: An inference acceleration framework for large language model with lossless generation accuracy," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024.

[11] Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen, "Sequoia: Scalable, robust, and hardware-aware speculative decoding," *https://arxiv. org/abs/2402.12374*, 2025.

[12] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are rnns: Fast autoregressive transformers with linear attention," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

[13] Zhen Qin, Dong Li, Weigao Sun, Weixuan Sun, Xuyang Shen, Xiaodong Han, Yunshen Wei, Baohong Lv, Xiao Luo, Yu Qiao, et al., "Transnormerllm: A faster and better large language model with improved transnormer," 2024.

[14] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al., "Rethinking attention with performers," *arXiv preprint arXiv:2009.14794*, 2020.

[15] Yifan Chen, Qi Zeng, Heng Ji, and Yun Yang, "Skyformer: Remodel self-attention with gaussian kernel and nystr\" om method," *Advances in Neural Information Processing Systems*, vol. 34, pp. 2122–2135, 2021.

[16] Yifan Chen, Qi Zeng, Dilek Hakkani-Tur, Di Jin, Heng Ji, and Yun Yang, "Sketching as a tool for understanding and accelerating self-attention for long sequences," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Seattle, United States, July 2022, pp. 5187–5199, Association for Computational Linguistics.

[17] Jiaping Wang, Simiao Zhang, Qiao-Chu He, and Yifan Chen, "Leetdecoding: A pytorch library for exponentially decaying causal linear attention with cuda implementations," 2025.

[18] Zhen Qin, Weigao Sun, Dong Li, Xuyang Shen, Weixuan Sun, and Yiran Zhong, "Lightning attention-2: A free lunch for handling unlimited sequence lengths in large language models," 2024.

[19] MiniMax, "Minimax-m1: Scaling test-time compute efficiently with lightning attention," 2025.

[20] Daewon Choi, Seunghyuk Oh, Saket Dingliwal, Jihoon Tack, Kyuyoung Kim, Woomin Song, Seojin Kim, Insu Han, Jinwoo Shin, Aram Galstyan, Shubham Katiyar, and Sravan Babu Bodapati, "Mamba drafters for speculative decoding," 2025.

[21] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim, "Gated linear attention transformers with hardware-efficient training," in *ICML*, 2024.

[22] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu, "Roformer: Enhanced transformer with rotary position embedding," *Neurocomputing*, vol. 568, 2024.

[23] Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou, "Enhancing chat language models by scaling high-quality instructional conversations," 2023.

[24] Guan Wang, Sijie Cheng, Xianyuan Zhan, Xiangang Li, Sen Song, and Yang Liu, "Openchat: Advancing open-source language models with mixed-quality data," *arXiv preprint arXiv:2309.11235*, 2023.

[25] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al., "Judging llm-as-a-judge with mt-bench and chatbot arena," *Advances in neural information processing systems*, vol. 36, pp. 46595–46623, 2023.

[26] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, and Harri Edwards et al., "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[27] Rohan Taori and Ishaan Gulrajani et al., "Stanford alpaca: An instruction-following llama model," 2023.

[28] Shashi Narayan, Shay B. Cohen, and Mirella Lapata, "Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, 2018.

[29] Yi Yang, Wen-tau Yih, and Christopher Meek, "WikiQA: A challenge dataset for open-domain question answering," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, 2015.

[30] Yushi Bai, Xin Lv, Jiajie Zhang, and Hongchang et al. Lyu, "LongBench: A bilingual, multitask benchmark for long context understanding," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Bangkok, Thailand, Aug. 2024, pp. 3119–3137, Association for Computational Linguistics.