



OWL: Overcoming Window Length-Dependence in Speculative Decoding for Long-Context Inputs

Jaeseong Lee^{*}, Seung-won Hwang^{*}, Aurick Qiao,
Gabriele Oliaro[†], Ye Wang, Samyam Rajbhandari

Snowflake AI Research, Seoul National University^{*}, Carnegie Mellon University[†]

Abstract

Speculative decoding promises faster inference for large language models (LLMs), yet existing methods fail to generalize to real-world settings. Benchmarks typically assume short contexts (e.g., 2K tokens), whereas practical workloads involve long contexts. We find current approaches degrade severely with long contexts; for instance, EAGLE3 even slows down the generation speed by 0.81 $\times$. We address these limitations by releasing a new long-context benchmark (LongSpecBench) and introducing a novel model (OWL). OWL achieves about 5 $\times$ higher acceptance length than EAGLE3 on long-context inputs through three innovations: (1) an LSTM-based drafter conditioned only on the last-token state, making it generalize to various lengths, (2) a special token [SPEC] in the verifier that produces richer representation for drafter, and (3) a hybrid algorithm combining both tree and non-tree decoding methods. We release all code and datasets to advance future research.¹

1 Introduction

Long-context generation is increasingly being important— Use cases include reasoning with long thinking path (DeepSeek-AI et al., 2025), multi-turn conversations, and agentic systems (Sadhukhan et al., 2025). In response, large language models (LLMs) have rapidly evolved to handle much longer input contexts— originally from 2K tokens by Vicuna (Chiang et al., 2023) to 128K by Llama-3 (Dubey et al., 2024). This expanded context capability unlocks complex reasoning and comprehensive information access, but it also comes with a steep computational cost: generating each token becomes slower as context length grows, due to the sequential nature of autoregressive decod-

ing and the large memory that must be accessed at every step.

Speculative decoding (Leviathan et al., 2023; Chen et al., 2024; Miao et al., 2024; Cai et al., 2024; Li et al., 2024b,a, 2025; Oliaro et al., 2025; Hu et al., 2025; Luo et al., 2025) is a promising solution to accelerate LLM inference. It uses a faster drafter to predict several upcoming tokens in a tree or sequence structure, and let the target LLM verify the drafted tokens. The tree- or non-tree decoding algorithm will accept proper tokens to keep the output distribution, which is used by the drafter to generate another set of next tokens. In memory-bound scenarios, the cost of verifying the multiple drafted tokens is hidden, leading to significant speedups— State-of-the-art speculative decoding method, EAGLE3 (Li et al., 2025) claims 6.5 $\times$ speedup over standard decoding.

However, such speedups often fail to generalize to real-world scenarios. First, common speculative benchmarks mostly assume a short 128 context, 2K context at maximum (e.g., Vicuna), and a batch size of one. However, in a real-world workload, such a short context easily shifts out of the memory-bound regime by increasing the batch size (Su et al., 2023; Miao et al., 2024; Liu et al., 2025; Li et al., 2025). Benchmarks with much longer contexts are more realistic for speculative decoding.

Second, existing methods degrade sharply in long contexts. For example, EAGLE3 achieves an acceptance length of only 1.28 (Figure 1a), generating just 0.28 extra tokens per verification step.

To address these issues, we introduce both a new long-context benchmark (LongSpecBench) and a new model (OWL). OWL achieves almost 5 $\times$ higher acceptance length than EAGLE3 on long-context inputs, with innovations in each key component of speculative decoding— drafter, verifier, and decoding algorithm:

- Length-general drafter: Unlike EAGLE3’s

^{*}Work done while visiting Snowflake. Correspond to seungwonh@snu.ac.kr

[†]<https://anonymous.4open.science/r/owl-BFB8>

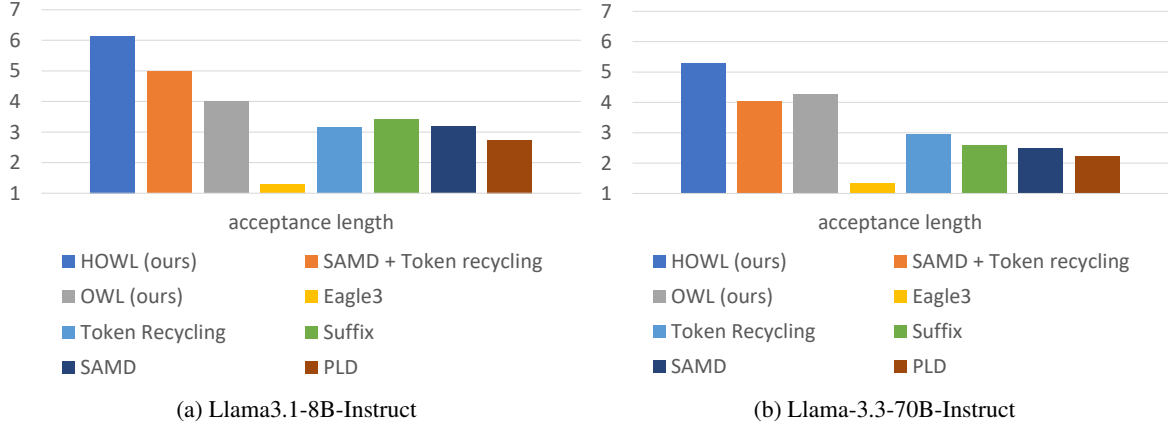


Figure 1: Speculative decoding performance on long-context inputs on Llama-3.1-8B-Instruct model on LongSpecBench. The acceptance length is the number of accepted tokens per verification step with the target LLM. Higher is better. While EAGLE3 (yellow) fails to generalize to long-context inputs, our method (blue) can achieve almost 5× higher acceptance length than EAGLE3.

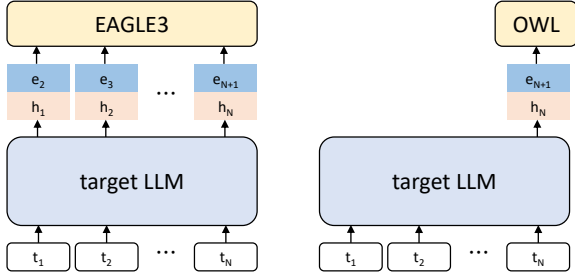


Figure 2: EAGLE3 (left) and OWL (right).

transformer, which fails to generalize beyond its trained 2K window (Tay et al., 2021), OWL uses an LSTM drafter conditioned only on the last-token state, avoiding context-length dependence (Figure 2).

- Specialized token for verifier: We introduce [SPEC] to provide a richer representation to the drafter, by letting the verifier predict an additional token beyond the verified tokens.
- Hybrid algorithm of tree- and non-tree decoding: We combine our tree-decoding-based model with an existing non-tree decoding method, achieving higher acceptance length and speedups.

First, we attribute the failure of existing EAGLE3 models to the transformer architecture (Vaswani et al., 2017) they use. The transformers do not generalize well beyond their seen context window, which EAGLE3 set as 2K following the common benchmarks. One possible approach would training a transformer with long-context. However, it requires a special dataset with

long-context. Additionally, we report EAGLE3-L, even after such training, is inferior to our proposal (Table 4).

Instead, OWL removes the need to feed all input tokens to the drafter. As depicted in Figure 2, we rely on the hidden state of a single token, the last token only, to generate the next tokens. To materialize it, we leverage the LSTM (Hochreiter and Schmidhuber, 1997) architecture. This design makes the drafter agnostic to the context length, and thus length-generalized.

Second, we introduce a special token in the verifier, to provide the drafter richer representation. We let the target LLM try to estimate an additional token beyond the accepted tokens at the verification step. We materialize this idea by appending a special token [SPEC] to the input (Figure 4). With the hidden state generated from [SPEC], we can increase the acceptance length of our drafter further, even keeping the latency intact.

Lastly, we explore the complementary benefit of tree- and non-tree decoding-based speculative decoding methods. We find that while some non-tree decoding based methods, such as SuffixDecoding (Oliaro et al., 2025), have lower acceptance length than our tree-decoding based methods, they can provide extremely high acceptance length in some cases (Figure 6). Since the best case scenario of non-tree decoding is accepting all the sequence of drafted tokens, we conditionally leverage this to enhance our best case scenario.

Experiments on variant scales of LLMs, such as Llama-3.1-8B-Instruct and Llama3.3-70B-Instruct (Dubey et al., 2024), demonstrate the ef-

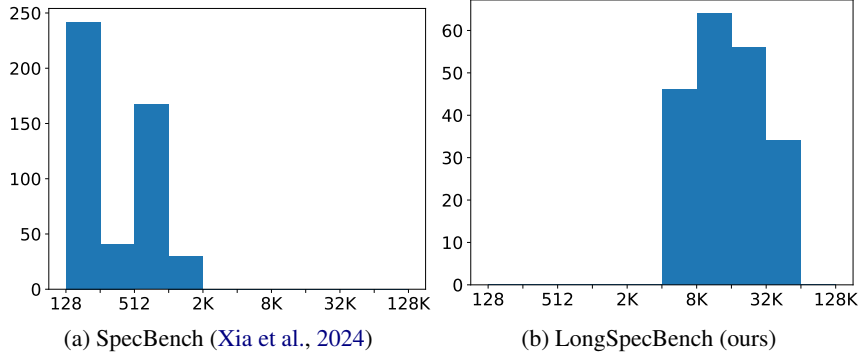


Figure 3: Context length distribution of existing benchmarks and LongSpecBench. Existing benchmarks, such as SpecBench (Xia et al., 2024), focus on short-context inputs (within 2K tokens). In contrast, LongSpecBench (ours) contains long-context inputs, with a length of tokens up to 64K tokens.

fectiveness of our method. We publicly release our code and datasets to facilitate future research.¹

2 Related Work

Speculative decoding (Leviathan et al., 2023) is a lossless acceleration technique of LLM decoding. It speculates multiple tokens in a row with a fast drafter, and verifies the tokens with the target LLM in parallel. In a memory-bound scenario, such additional computational cost is hidden.

While the initial technique speculated tokens in a sequence-like structure, tree-decoding emerged to speculate tokens in a tree-like structure (Miao et al., 2024; Chen et al., 2024; Cai et al., 2024; Li et al., 2024b), although some recent works (Oliaro et al., 2025) proposed non-tree decoding methods that perform comparably.

Recently, EAGLE3 (Li et al., 2025) achieved the state-of-the-art in benchmarks such as SpecBench (Xia et al., 2024). In response, inference engines such as vLLM (Kwon et al., 2023) or SGLang (Zheng et al., 2024) also integrated EAGLE3 as a built-in speculative decoding to accelerate LLM inference.

However, we find that the high acceptance length of EAGLE3 is not viable when the context length is beyond the trained context window. We design a benchmark to consider such a scenario, and propose a length-generalized speculative decoding method.

3 Proposed Method

3.1 LongSpecBench: A New Benchmark with Long-Context Inputs

Existing benchmarks for speculative decoding, such as SpecBench (Xia et al., 2024) or the EA-

GLE3 benchmark (Li et al., 2025), primarily focus on short-context inputs, typically within 2K tokens (Figure 3a, 8a). However, real-world applications often involve much longer contexts, which can significantly impact the performance of speculative decoding methods.

To address this gap, we build LongSpecBench, a new benchmark specifically designed to evaluate the effectiveness of speculative decoding techniques on long-context inputs.

To utilize the real-world use cases, we leverage WildChat-4.8M,² which records conversations between human users and ChatGPT (Zhao et al., 2024). We sample 200 examples with input length distributed ranging from 4K to 64K tokens (Figure 3b).

Surprisingly, this new benchmark reveals that existing speculative decoding methods, such as EAGLE3, do not generalize well to long-context inputs. For instance, EAGLE3 fails to produce a high acceptance length on LongSpecBench—only 1.28 (Figure 1a). This highlights the need for new approaches that can effectively handle long-context scenarios.

3.2 Length-Generalized Speculative Decoding When Input Exceeds Trained Length

In this section, we describe our proposed method, consisting of innovation in each of three key components of speculative decoding: drafter, verifier, and decoding algorithm. First, we propose a length-generalized drafter to address the context-length dependence of existing transformer-based drafters (§3.2.1). Second, we introduce a specialized token for the verifier to signal the target LLM to predict beyond the verified tokens (§3.2.2). Lastly,

²hf.co/datasets/allenai/WildChat-4.8M

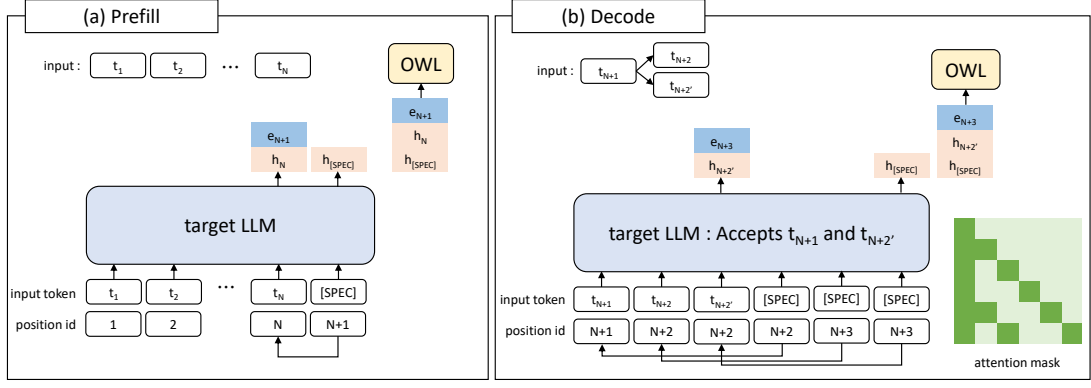


Figure 4: Overview of [SPEC] for verifier in the inference stage, each for (a) prefill stage and (b) decode stage.

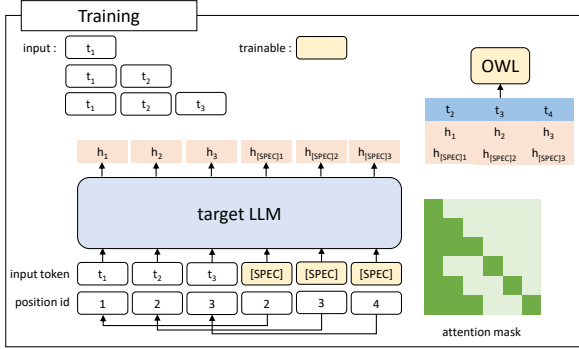


Figure 5: Overview of [SPEC] for verifier in the training stage.

we present a hybrid algorithm that combines tree-decoding and non-tree decoding methods to leverage the strengths of both (§3.2.3).

3.2.1 OWL: Length-Generalized Drafter

As transformer architecture is dependent on the trained context window (Tay et al., 2021; Yang et al., 2025), EAGLE3, following the transformer architecture, has similar limitations.

In contrast, we propose OWL, a length-generalized drafter by removing the dependency on all the input tokens. As depicted in Figure 2, unlike EAGLE3, which feeds all input tokens to the drafter, OWL relies only on the hidden state of a single token, the last token only, to speculate the next tokens. To materialize it, we leverage the LSTM (Hochreiter and Schmidhuber, 1997) architecture. This design makes the drafter agnostic to the context length, and thus length-generalized. The ability to understand the long context would be yielded to the target LLM (Dubey et al., 2024). Empirically, we find that even a short context length, such as 256 tokens, is sufficient to train such a drafter to support long-context inputs.

In detail, we establish the architecture of OWL as follows. Given the predicted next token t_{N+1} , the last hidden states $h_N \in \mathbb{R}^{d_0}$, and a trainable embedding layer E , we first sum up embedding of t_{N+1} and the projection of h_N , inspired by MLP-Speculator (Wertheimer et al., 2024):

$$e_{N+1} = E(t_{N+1}) \quad (1)$$

$$s^m = W^m(h_N) + \alpha \cdot e_{N+1}, m \in f, i, o, c \quad (2)$$

where we define projection $W^f, W^i, W^o, W^c \in \mathbb{R}^{d_0 \times d}$ for forget, input, output, and cell state. α is defined following Wertheimer et al. (2024) as follows:

$$\alpha_0 = 2^{-\frac{1}{2n}} \quad (3)$$

$$\alpha = \frac{2\alpha_0}{(1 - \alpha_0^2) \cdot d} \quad (4)$$

where n is the maximum tree depth we aim to generate.

Now we imitate the flow of LSTM forward as follows:

$$g^m = \sigma(s^m), m \in f, i, o \quad (5)$$

$$s^c = f(s^c) \cdot g^i \quad (6)$$

$$z = z \cdot g^f + s^c \quad (7)$$

$$h_{N+1} = f(z) \cdot g^o \quad (8)$$

where σ is the sigmoid function, and f is the layer normalization along with an activation function, which we use GeLU. z is the cell state, which we initialize with zeros. We speculate t_{N+2} from h_{N+1} using a trainable head, and recurrently speculate the next token in a similar manner, reusing the trainable weight parameters.

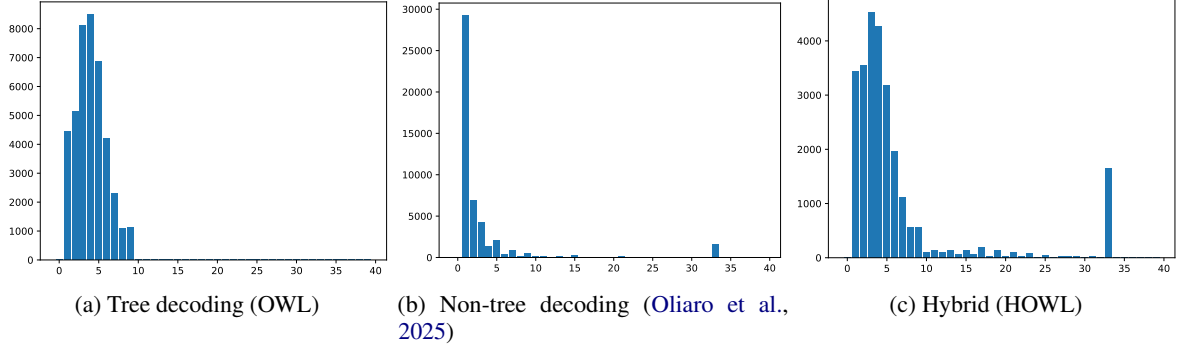


Figure 6: Histogram of acceptance length on Llama-3.1-8B-Instruct model on LongSpecBench. While OWL (left) achieves a higher acceptance length than the non-tree decoding method (middle) on average, the non-tree decoding method often achieves extremely high acceptance length. We build a hybrid method combining the benefits of the two, further improving the average acceptance length (right).

3.2.2 [SPEC] in Verifier: Empowering Drafter with Richer Representation

With our simplified architecture, OWL can generalize to long-context inputs. However, the acceptance length is still limited by the drafter’s capability. To empower the drafter with richer representation, we let the target LLM try to estimate an additional token beyond the accepted tokens at the verification step. Then we leverage that estimation in the drafter to increase the acceptance length further.

We describe how to introduce [SPEC] to the verifier in 1) the prefill stage in inference, 2) the decode stage in inference, and 3) the training stage below.

Prefill Stage As depicted in Figure 4 (a), we append [SPEC] to the input at the prefill stage. This signals the target LLM to generate the hidden state for estimating the additional token. The estimated hidden state is consumed by the drafter to generate draft tokens.

Decode Stage At the decode stage, as shown in Figure 4 (b), we append [SPEC] after each tree path. We enable this by appending the same number of [SPEC] as the number of tree nodes of the drafted tree, and manipulating the position ids and attention mask accordingly. For example, the first [SPEC] follows the path (t_{N+1}) , the second [SPEC] follows the path (t_{N+1}, t_{N+2}) , and the third [SPEC] follows the path $(t_{N+1}, t_{N+2'})$. When a path is accepted, we take the hidden state of the corresponding [SPEC] as the estimation of the next token. This allows the target LLM to predict beyond the accepted tokens of the drafted tree, providing the drafter richer representation.

Importantly, we keep the computational cost the

same as the original tree decoding method by decreasing the tree size by half. In section 4, we empirically show that this approach significantly increases the acceptance length even if we pass the same number of tokens to the verifier target LLM.

Training Stage To enable signaling with [SPEC] while keeping the target LLM intact, we train the embedding of [SPEC] along with the drafter in an efficient manner.

The efficiency of autoregressive training comes from computing the next token hidden states of all prefixes in a single forward pass. To train the drafter with [SPEC], we need to compute the hidden states of [SPEC] for all possible prefixes. As shown in Figure 5, we append the same number of [SPEC] as the number of prefixes in the given input. We then manipulate the position ids and attention mask to ensure that each [SPEC] attends only to its preceding tokens. This allows us to compute the hidden states for all possible prefixes in a single forward pass, maintaining training efficiency.

In detail, suppose we get logits $y_{k \oplus 1}, \dots, y_{k \oplus n}$ speculated from $t_k, h_{k-1}, h_{[SPEC]_{k-1}}$. Then the loss $\mathcal{L}$ is formulated as

$$\mathcal{L} = \frac{1}{N} \sum_k \left(\frac{1}{n} \sum_j CE(y_{k \oplus j}, t_{k+j}) + CE(y_{[SPEC]_{k-1}}, t_k) \right) \quad (9)$$

where CE is cross-entropy loss, and $y_{[SPEC]_{k-1}}$ is generated from $h_{[SPEC]_{k-1}}$ using the head of the target LLM.

3.2.3 HOWL: Hybrid Algorithm with Non-Tree Decoding

While the above architectural innovations for the drafter and verifier significantly improve the ac-

ceptance length, we further explore the decoding algorithm itself. We find out the complementary benefit of tree- and non-tree decoding based speculative decoding methods.

As depicted in Figure 6, we find that while some non-tree decoding based methods, such as SuffixDecoding (Oliaro et al., 2025), have lower acceptance length than our tree-decoding-based method in average, they can provide extremely high acceptance length in some cases. Since the best case scenario of non-tree decoding is accepting all the sequence of drafted tokens, we conditionally leverage this to enhance our best case scenario.

Algorithm 1 describes HOWL, estimating the acceptance length of the non-tree decoding method, the non-tree decoding version of SuffixDecoding (Oliaro et al., 2025), as *score*, used for routing decision. We follow Oliaro et al. (2025) to estimate *score*.

If *score* is higher than a threshold *c*, we use non-tree decoding method to verify the drafted tokens. To ensure the best case scenario, we do not append any [SPEC] to the input in this case, since doing so would limit the acceptance length when the same computational budget is assumed.

If *score* is not higher than a threshold *c*, we use our tree-decoding-based method. If the previous step used the non-tree decoding method, we did not use [SPEC], thus we use OWL trained without [SPEC]. Otherwise, we use OWL trained with [SPEC]. In any case of tree-decoding based method, we append [SPEC] to the input and manipulate the position ids and attention mask accordingly, as described in Figure 4 (b) before the verification step. Note that this does not alter the non-tree decoding algorithm *SuffixLinear*, since this is only used in the *OurTreeVerify* step.

In practice, we set *c* as the largest acceptance length OWL can achieve, to ensure we use OWL in the average case. As a result, the best-case scenario of non-tree decoding is enabled, while we still leverage the benefits of our tree-decoding-based method in the average-case scenario.

4 Experiments

We investigate the following research questions:

- **RQ1:** How does OWL perform on long-context inputs compared to existing speculative decoding methods?
- **RQ2:** Does OWL generally perform well on benchmarks with various context lengths?

Algorithm 1 Hybrid Decoding Algorithm

Require: $l \leftarrow$ Input sequence
Require: $S \leftarrow$ Token id of [SPEC]
Require: $c \leftarrow$ Threshold for switching tree/non-tree decoding
Require: $D_{spec} \leftarrow$ OWL trained with [SPEC]
Require: $D_{nospec} \leftarrow$ OWL trained without [SPEC]
1: $SuffixPrefillCache(l)$
2: $t_{next}, h_{last}, h_S = Prefill(l)$ $\triangleright$ Figure 4(a)
3: $lastwaslinear \leftarrow$ False
4: **while** End of sequence not reached **do**
5: $d, score \leftarrow SuffixLinear(l, t_{next})$
6: **if** $score > c$ **then**
7: $lastwaslinear \leftarrow$ True
8: $d, t_{next}, h_{last} \leftarrow NonTreeVerify(d)$
9: **else**
10: **if** $lastwaslinear$ **then**
11: $d \leftarrow D_{nospec}(t_{next}, h_{last})$
12: $lastwaslinear \leftarrow$ False
13: **else**
14: $d \leftarrow D_{spec}(t_{next}, h_{last}, h_S)$
15: **end if**
16: $Prepare(d) \triangleright$ Append [SPEC], Attention mask, Position id manipulation
17: $d, t_{next}, h_{last}, h_S \leftarrow OurTreeVerify(d)$ $\triangleright$ Figure 4(b)
18: **end if**
19: $l \leftarrow l; d$ $\triangleright$ sequence concatenation
20: $SuffixCache(d)$
21: $N \leftarrow N + |d|$
22: **end while**
23: **return** l

- **RQ3:** Does [SPEC] increase acceptance length even if the inference budget is intact?
- **RQ4:** What if we allow a much longer dataset to train a custom EAGLE3-L to increase the context length?

4.1 Experimental Setup

Target LLMs We evaluate on two different scales of LLMs, Llama-3.1-8B-Instruct (Dubey et al., 2024) and Llama-3.3-70B-Instruct (Dubey et al., 2024), which are used in related works (Li et al., 2025; Oliaro et al., 2025).

Comparisons We compare our method with various existing speculative decoding methods, including PLD (Saxena, 2023), SuffixDecoding (Oliaro et al., 2025), SAMD (Hu et al., 2025), Token Recycling (Luo et al., 2025), and EAGLE3 (Li et al., 2025). We also compare with hybrid methods, combining SAMD and Token Recycling (Hu et al., 2025). We detail the inference setups in the Appendix.

Training Details We train OWL on Ultrachat-200k³ (Ding et al., 2023) and Magicoder⁴ (Wei

³hf.co/datasets/HuggingFaceH4/ultrachat_200k

⁴hf.co/datasets/ise-uiuc/Magicoder-OSS-Instruct-75K

		Llama-3.1-8B	Llama-3.3-70B
Methods	PLD (Saxena, 2023)	2.75	2.24
	Suffix Decoding (Oliaro et al., 2025)	3.41	2.61
	SAMD (Hu et al., 2025)	3.18	2.48
	Token Recycling (Luo et al., 2025)	3.16	2.97
	EAGLE3 (Li et al., 2025)	1.28	1.35
	OWL (ours)	4.00	4.27
Hybrid Methods	SAMD + Token recycling (Hu et al., 2025)	4.98	4.05
	HOWL (ours)	6.14	5.31

Table 1: Acceptance length comparison on LongSpecBench with Llama-3.1-8B-Instruct and Llama-3.3-70B-Instruct as the base models.

		Speedup
Methods	baseline	1.00×
	PLD	1.59×
	Suffix Decoding	2.18×
	SAMD	2.16×
	Token Recycling	1.75×
	EAGLE3	0.81×
	OWL without [SPEC]	2.00×
	OWL (ours)	2.35×
Hybrids	SAMD + Token Recyling	2.77×
	HOWL (ours)	3.08×

Table 2: Token generation (tokens/sec) speed comparison on LongSpecBench with Llama-3.3-70B-Instruct as the base model.

et al., 2024). Inspired by Wertheimer et al. (2024), we first chunk the data by size of 64 and generate 256 tokens in the preprocessing step, using vLLM (Kwon et al., 2023). We then train OWL with these generated chunks with sequence length of 256. We train with batch size of 2048, learning rate of 1e-3, for 3000 iterations. We use hidden size d of 12288. All training is done with 8×H200 GPUs.

To train the long-context version of EAGLE3, we leverage SpecForge (Shenggui Li, 2025), following their default settings, except for setting the number of epochs as 16 to give a fair training time as OWL. We set the maximum sequence length as 32K, since we couldn’t increase more due to the OOM error.

4.2 RQ1: OWL is the Best on Long-Context Inputs over Various Model Sizes

Among various comparisons, OWL achieves the highest acceptance length on LongSpecBench, for both Llama-3.1-8B-Instruct and Llama-3.3-70B-Instruct (Table 1). Surprisingly, EAGLE3 shows an

acceptance length of around 1.28, which is far below the training-free methods, such as PLD, Suffix Decoding, SAMD, or Token Recycling. In contrast, OWL achieves an acceptance length of around 4.00-4.27, which is way beyond existing methods.

Moreover, OWL shows better acceptance length on larger models, such as Llama-3.3-70B-Instruct. This is different from the trend of training-free retrieval methods such as PLD, Suffix Decoding, SAMD, and Token Recycling, whose acceptance length gets lower. We hypothesize that as the output distribution becomes more delicate, training-free methods suffer from predicting the complicated output distribution.

When we compare the hybrid methods, HOWL achieves an acceptance length of 6.14 at most, almost 5× than that of EAGLE3. Since we combine OWL with the training-free retrieval method, Suffix Decoding, the degradation in the larger model is expected.

These improvement translates into the highest token generation speed as well (Table 2). Surprisingly, the state-of-the-art method, EAGLE3, makes the generation speed even slower, to 0.81×, due to their small acceptance length and drafting overhead. In contrast, OWL and HOWL neatly show the best speedup, compared with other speculative decoding baselines.

4.3 RQ2: OWL Generalizes over Various Context Length

Table 3 shows similar acceptance length over benchmarks with different length distribution, while others do not. For example, while EAGLE3 shows the best acceptance length on the short benchmark, SpecBench, it shows the worst acceptance length on the long benchmark, LongSpecBench. Retrieval methods, such as PLD, Suffix Decoding, SAMD, and Token Recycling

	SpecBench	LongSpecBench	min
PLD	1.41	2.75	1.41
Suffix Decoding	1.56	3.41	1.56
SAMD	1.42	3.18	1.42
Token Recycling	2.73	3.16	2.73
EAGLE3	5.79	1.28	1.28
OWL	4.14	4.00	4.00

Table 3: Acceptance length comparison on SpecBench (Xia et al., 2024) and LongSpecBench with Llama-3.1-8B-Instruct as the base model. The acceptance length of OWL is consistent across benchmarks with various length.

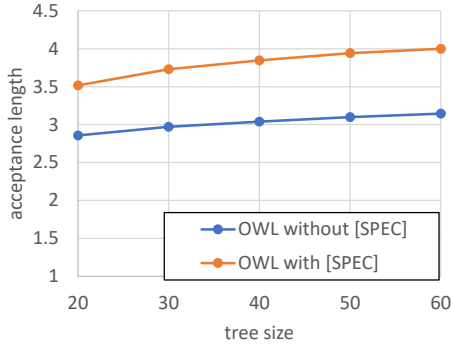


Figure 7: Acceptance length comparison between OWL with and without [SPEC], with same tree size is set.

show much lower acceptance length on the short benchmark, SpecBench, compared with the longer benchmark, LongSpecBench. In contrast, the acceptance length of OWL is above 4 on both benchmarks.

4.4 RQ3: [SPEC] Significantly Improves Acceptance Length

When we compare passing the same number of tokens to the verifier, with and without [SPEC], including [SPEC] in the tokens significantly increases acceptance length (Figure 7). For example, if we generate 30 tokens from the drafter and append 30 [SPEC] s to make a tree size of 60, this increases acceptance length by almost 1, compared with generating 60 tokens from the drafter and directly passing to the verifier. The acceptance length is increasing more rapidly as we increase the tree size, benefiting from tree-decoding more.

Moreover, the row OWL without [SPEC] and OWL in Table 2 shows that [SPEC] contributes to the actual speedup as well. These verify that [SPEC] significantly improves speedup without affecting the latency.

	OWL	EAGLE3	EAGLE3-L
train context window	256	2048	32768
train data length	short	short	long
acceptance length	4.00	1.28	3.23

Table 4: LongSpecBench comparison of OWL, EAGLE3, and EAGLE3-L which is a version we trained with a much longer context window to construct a stronger baseline.

	LSTM [SPEC] Hybrid			AL
HOWL	✓	✓	✓	6.14
OWL	✓	✓		4.00
OWL w/o [SPEC]	✓			3.14
RNN-based				2.99
EAGLE3				1.28

Table 5: Contribution of each innovation to the acceptance length (AL) with Llama-3.1-8B-Instruct on LongSpecBench. RNN-based follows the architecture of Wertheimer et al. (2024).

4.5 RQ4: Even if We Train Longer-Context EAGLE3-L, OWL Outperforms as Well

Even if we train an EAGLE3-L by allowing much longer data explicitly to support a longer context window, OWL outperform it (Table 4). We use LongAlign (Bai et al., 2024) and LongWriter (Bai et al., 2025) to support such a long sequence to train the speculator. OWL is more efficient– it does not require a special dataset, or longer sequence length, while achieving higher acceptance length.

4.6 Ablation Studies

Table 5 shows that each innovation contributes to the performance of OWL. While we arbitrarily used an LSTM architecture to remove window length-dependency, we also implement an RNN-based architecture, following Wertheimer et al. (2024). LSTM architecture is better, [SPEC] makes it even better, and HOWL achieves the best.

5 Conclusion

In this paper, we identify the limitations of existing speculative decoding methods on long-context inputs, and propose OWL, a length-generalized speculative decoding method for long-context inputs. Our method can achieve almost 5× higher acceptance length than EAGLE3 on long-context inputs. We publicly release our code and datasets to facilitate future research.

6 Limitations

While OWL shows the best performance on long-context inputs, real-world workload may contain only a small batch of short-context samples as well. For this less-likely scenario, we can consider a hybrid with EAGLE3, which we leave as future work.

Although we empirically showed that LSTM architecture works well for length-generalization, other alternatives, such as state-space models (Gu et al., 2022), are yet unexplored. We leave those directions as future work.

References

- Yushi Bai, Xin Lv, Jiajie Zhang, Yuze He, Ji Qi, Lei Hou, Jie Tang, Yuxiao Dong, and Juanzi Li. 2024. [LongAlign: A Recipe for Long Context Alignment of Large Language Models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1376–1395, Miami, Florida, USA. Association for Computational Linguistics.
- Yushi Bai, Jiajie Zhang, Xin Lv, Linzhi Zheng, Siqi Zhu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2025. LongWriter: Unleashing 10,000+ Word Generation from Long Context LLMs. In *The Thirteenth International Conference on Learning Representations*.
- 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. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 5209–5235. PMLR.
- Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. 2024. Sequoia: Scalable and Robust Speculative Decoding. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- 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.
- DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. [DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning](#). Preprint, arXiv:2501.12948.
- Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. [Enhancing Chat Language Models by Scaling High-quality Instructional Conversations](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3029–3051, Singapore. Association for Computational Linguistics.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 514 others. 2024. The Llama 3 Herd of Models.
- Albert Gu, Karan Goel, and Christopher Re. 2022. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. [Long Short-Term Memory](#). *Neural Computation*, 9(8):1735–1780.
- Yuxuan Hu, Ke Wang, Xiaokang Zhang, Fanjin Zhang, Cuiping Li, Hong Chen, and Jing Zhang. 2025. [SAM Decoding: Speculative Decoding via Suffix Automaton](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12187–12204, Vienna, Austria. Association for Computational Linguistics.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. [Efficient Memory Management for Large Language Model Serving with PagedAttention](#). In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP ’23, pages 611–626, New York, NY, USA. Association for Computing Machinery.
- Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19274–19286. PMLR.
- Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024a. [EAGLE-2: Faster Inference of Language Models with Dynamic Draft Trees](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7421–7432, Miami, Florida, USA. Association for Computational Linguistics.
- Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024b. EAGLE: Speculative Sampling Requires Rethinking Feature Uncertainty. In *Forty-First 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](#). Preprint, arXiv:2503.01840.

- Xiaoxuan Liu, Jongseok Park, Langxiang Hu, Woosuk Kwon, Zhuohan Li, Chen Zhang, Kuntai Du, Xiangxi Mo, Kaichao You, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. 2025. [TurboSpec: Closed-loop Speculation Control System for Optimizing LLM Serving Goodput](#). *Preprint*, arXiv:2406.14066.
- Xianzhen Luo, Yixuan Wang, Qingfu Zhu, Zhiming Zhang, Xuanyu Zhang, Qing Yang, and Dongliang Xu. 2025. [Turning Trash into Treasure: Accelerating Inference of Large Language Models with Token Recycling](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6816–6831, Vienna, Austria. Association for Computational Linguistics.
- Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. 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*, volume 3 of *ASPLOS '24*, pages 932–949, New York, NY, USA. Association for Computing Machinery.
- Gabriele Oliaro, Zhihao Jia, Daniel Campos, and Aurick Qiao. 2025. [SuffixDecoding: Extreme Speculative Decoding for Emerging AI Applications](#). *Preprint*, arXiv:2411.04975.
- Ranajoy Sadhukhan, Jian Chen, Zhuoming Chen, Vashisth Tiwari, Ruihang Lai, Jinyuan Shi, Ian En-Hsu Yen, Avner May, Tianqi Chen, and Beidi Chen. 2025. MagicDec: Breaking the Latency-Throughput Tradeoff for Long Context Generation with Speculative Decoding. In *The Thirteenth International Conference on Learning Representations*.
- Apoorv Saxena. 2023. Prompt lookup decoding.
- Chao Wang Shenggui Li, Yikai Zhu. 2025. SpecForge: Train speculative decoding models effortlessly.
- Qidong Su, Christina Giannoula, and Gennady Pekhimenko. 2023. [The Synergy of Speculative Decoding and Batching in Serving Large Language Models](#). *Preprint*, arXiv:2310.18813.
- Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2021. Long Range Arena : A Benchmark for Efficient Transformers. In *International Conference on Learning Representations*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. 2024. Magicoder: Empowering Code Generation with OSS-Instruct. In *Proceedings of the 41st International Conference on Machine Learning*, pages 52632–52657. PMLR.
- Davis Wertheimer, Joshua Rosenkranz, Thomas Parnell, Sahil Suneja, Pavithra Ranganathan, Raghu Ganti, and Mudhakar Srivatsa. 2024. [Accelerating Production LLMs with Combined Token/Embedding Speculators](#). *Preprint*, arXiv:2404.19124.
- Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhi-fang Sui. 2024. Unlocking Efficiency in Large Language Model Inference: A Comprehensive Survey of Speculative Decoding. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 7655–7671, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.
- An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, Junyang Lin, Kai Dang, Kexin Yang, Le Yu, Mei Li, Minmin Sun, Qin Zhu, Rui Men, Tao He, and 9 others. 2025. [Qwen2.5-1M Technical Report](#). *Preprint*, arXiv:2501.15383.
- Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. 2024. WildChat: 1M ChatGPT Interaction Logs in the Wild. In *The Twelfth International Conference on Learning Representations*.
- Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. 2024. SGLang: Efficient execution of structured language model programs. In *The Thirty-Eighth Annual Conference on Neural Information Processing Systems*.

A Appendix

A.1 Inference Details

We follow implementations in SpecBench (Xia et al., 2024), and follow their recommended hyperparameters, except for increasing the tree depth of EAGLE3 to 8, following their paper (Li et al., 2025). In detail, we used hyperparameters as follows:

- EAGLE3: We use tree size of 60, top-k of 10, and depth of 8. We use official models provided by the author for Llama-3.1-8B-Instruct⁵ and Llama-3.3-70B-Instruct.⁶
- OWL: Following EAGLE3, we use tree size of 60, top-k of 10, and depth of 8.

⁵hf.co/yuhuili/EAGLE3-LLaMA3.1-Instruct-8B

⁶hf.co/yuhuili/EAGLE3-LLaMA3.3-Instruct-70B

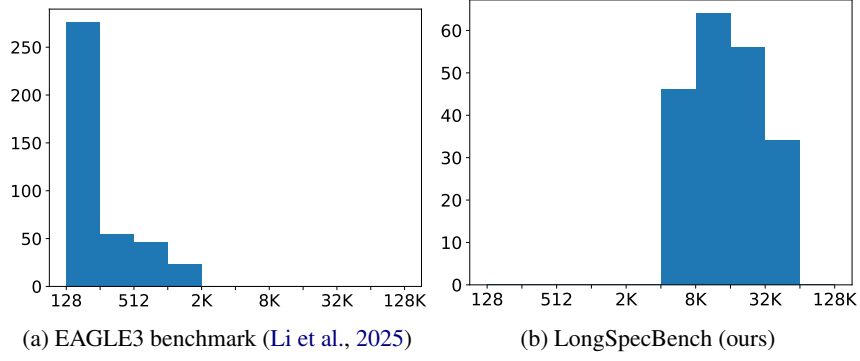


Figure 8: Context length distribution of benchmark used in EAGLE3 (Li et al., 2025) and LongSpecBench.

- PLD: We use max-ngram-size of 3, and num-pred-tokens of 10.
- SAMD: We use n-predicts of 40, max-predicts of 70. For hybrid, we use len-threshold of 5, len-bias of 5.
- Token recycling: We use output-id-topk of 8, with tree version 2.2.2.
- Suffix decoding: We use max-spec-factor of 2. For hybrid, we use max-suffix-depth of 64, and suffix-threshold of 9, the maximum acceptance length OWL can achieve (§3.2.3).

Following related works (Li et al., 2025; Oliaro et al., 2025; Xia et al., 2024; Luo et al., 2025; Hu et al., 2025), we use batch size of 1. We fairly optimized each methods with a static cache design provided by SAMD (Hu et al., 2025), and use fp16 data type. For Llama-3.1-8B-Instruct, we use 1×H200 GPU, and for Llama-3.3-70B-Instruct, we use 8×H200 GPUs.

A.2 Length Distribution of EAGLE3 Benchmark and LongSpecBench

Figure 8 further compares the length distribution between the benchmark EAGLE3 used and LongSpecBench.