EAGLE-3: Scaling up Inference Acceleration of Large Language Models via Training-Time Test

Yuhui Li[1,2], **Fangyun Wei**[3], **Chao Zhang**[2], **Hongyang Zhang**[1,4]
[1]University of Waterloo [2]Peking University [3]Microsoft Research [4]Vector Institute
yuhui.li@stu.pku.edu.cn, fawe@microsoft.com
c.zhang@pku.edu.cn, hongyang.zhang@uwaterloo.ca

Abstract

The sequential nature of modern LLMs makes them expensive and slow, and speculative sampling has proven to be an effective solution to this problem. Methods like EAGLE perform autoregression at the feature level, reusing top-layer features from the target model to achieve better results than vanilla speculative sampling. A growing trend in the LLM community is scaling up training data to improve model intelligence without increasing inference costs. However, we observe that scaling up data provides limited improvements for EAGLE. We identify that this limitation arises from EAGLE's feature prediction constraints. In this paper, we introduce EAGLE-3, which abandons feature prediction in favor of direct token prediction and replaces reliance on top-layer features with multi-layer feature fusion via a technique named training-time test. These improvements significantly enhance performance and enable the draft model to fully benefit from scaling up training data. Our experiments include both chat models and reasoning models, evaluated on five tasks. The results show that EAGLE-3 achieves a speedup ratio up to 6.5x, with about 1.4x improvement over EAGLE-2. In the SGLang framework, EAGLE-3 achieves a 1.38x throughput improvement at a batch size of 64. The code is available at https://github.com/SafeAILab/EAGLE.

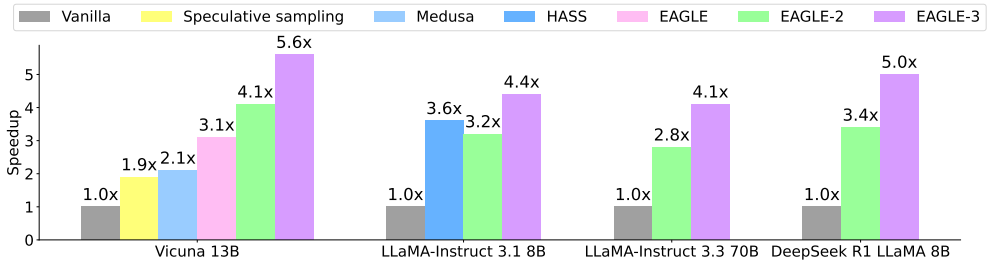

Figure 1: Speedup ratios of different methods at temperature=0. For the standard speculative sampling, Vicuna-13B uses Vicuna-68M as the draft model. In Table 1, we present comparisons with additional methods, but this figure only showcases a subset. Chat model's evaluation dataset is MT-bench, and the reasoning model's evaluation dataset is GSM8K. DeepSeek R1 LLaMA 8B refers to DeepSeek-R1-Distill-LLaMA 8B.

1 Introduction

Modern Large Language Models (LLMs) are being applied to more domains, with their improved capabilities driven by scaling model parameters—some LLMs now exceed hundreds of billions of

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

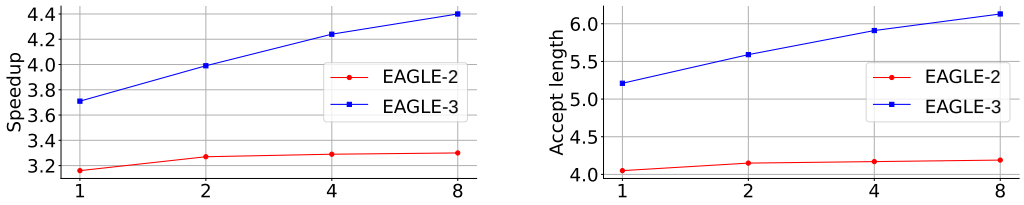

Figure 2: Scaling law evaluated on the MT-bench using LLaMA-Instruct 3.1 8B as the target model, with the x-axis representing the data scale relative to ShareGPT. The new architectural designs in EAGLE-3 enable an increasing scaling curve, which was never observed in the previous works.

parameters. In autoregressive generation, each token requires accessing all model parameters, making LLM inference slow and costly.

Recently, test-time scaling up has gained significant attention. Models like ChatGPT o1 and DeepSeek-R1 [1] engage in deliberate reasoning before responding, pushing the boundaries of LLM capabilities at the cost of longer inference time. However, these models often require lengthy reasoning processes, making them extremely costly, while the increased response time severely impacts user satisfaction. These reasoning models significantly increase the proportion of inference costs in the overall LLM pipeline, driving researchers to explore cheaper and faster inference optimization methods.

Speculative sampling methods can reduce LLM latency by partially parallelizing the generation process. These methods rapidly generate draft tokens and then verify them in parallel. This allows multiple tokens to be produced in a single forward pass, significantly reducing inference latency. As a state-of-the-art speculative sampling method, EAGLE [2] decodes an LLM by leveraging the top-layer features of the target model (i.e., the representations before the LM head) to perform *next-feature prediction*. As shown in the first box in Figure 3, EAGLE trains the draft model by feeding all previous features $f_1, ..., f_t$ into the target model to predict the next feature $\hat{f}_{t+1}$, with the training data feature f_{t+1} as the label. In the test phase (the second box), EAGLE autoregressively predicts the next feature $\hat{f}$ and then uses the target model's LM head to obtain the draft token $\hat{t}$. By leveraging the rich information from the target model, EAGLE achieves significantly better acceleration compared to vanilla speculative sampling. Subsequent methods such as HASS [3] and Falcon [4] also adopt the approach of predicting the next feature using the current feature sequence.

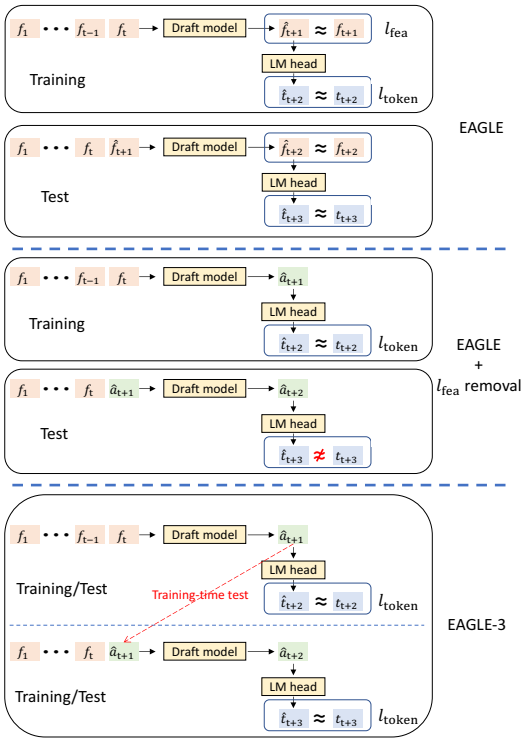

Figure 3: Illustration of **training-time test** (the bottom part) and its comparison with other draft methods (the upper and middle parts). f denotes the feature, t denotes the token, and a represents the unconstrained vectors. We use the hat to denote the predictions from models. For EAGLE and EAGLE + l_{fea} removal (the upper and middle parts), the training and test processes are different. However, for EAGLE-3 (the bottom part), the training and test processes are the same.

Recent LLMs have increasingly relied on larger training datasets to achieve better performance. For example, LLaMA series models with sizes of 7B (8B) have used 1T, 2T, and 15T tokens of training data for LLaMA 1 [5], LLaMA 2 [6], and LLaMA 3 [7], respectively, resulting in significant improvements across various metrics while keeping the model architecture and inference cost largely unchanged. Similarly, we aim to improve the acceptance rate and acceleration ratio of EAGLE by increasing its training data. Unfortunately, we observe that the gains from additional training data for EAGLE are limited. We analyze the reasons behind this phenomenon. As shown in the upper part of Figure 3, in the test

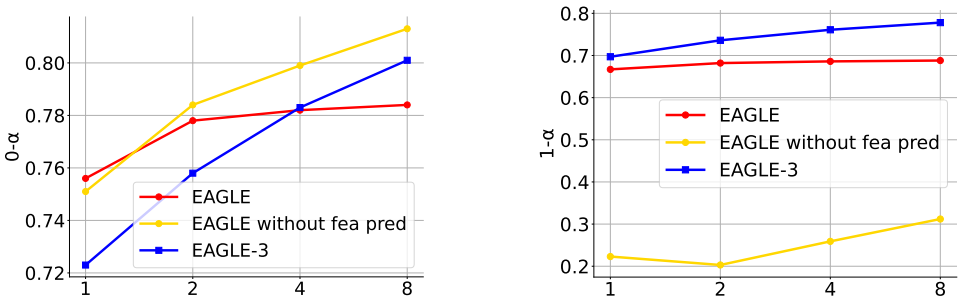

Figure 4: Comparison of acceptance rates across different methods, with the x-axis representing the data scale relative to ShareGPT.

phase, EAGLE performs autoregressive prediction at the feature level (as both input and output of the draft model are with hat), predicting the next feature and then feeding the feature into the LM head of the target model to obtain the token distribution. In the training phase, EAGLE's loss function consists of two components: the feature prediction loss l_{fea} and the token prediction loss l_{token}. Thanks to the feature prediction loss, the draft model after training (when the input of the draft model is without hat and the output of the draft model is with hat) can adapt to the test case and acquire multi-step prediction capabilities. However, with token prediction as the ultimate goal, feature prediction can be seen as an additional constraint, which limits the expressiveness of the draft model and makes it difficult to benefit from increased data. After removing the feature constraint and expanding the training data (the middle part of Figure 3), as shown in Figure 4, the acceptance rate 0-α of the first draft token improves significantly. However, the output of the draft model in the training phase, denoted as $\hat{a}_{t+1}$, is far away from the ground-truth f_{t+1}, causing the input sequence $f_1, f_2, \cdots, f_t, \hat{a}_{t+1}$ in the test phase to deviate significantly from the training distribution, resulting in a very low acceptance rate 1-α for the second draft token, as shown in Figure 4. We can address this issue by moving the output $\hat{a}_{t+1}$ back into the input of draft model in the training process (the bottom of Figure 3), similar to autoregressive inference. Using this method, the benefits of increasing training data become more pronounced. We name this technique as training-time test.

EAGLE and speculative sampling methods such as Medusa [8] reuse the top-layer features of the target model, specifically the features immediately before the LM head. For an LM head with a full-rank weight matrix, the top-layer features corresponding to the logits of the next token are unique, ensuring that the information contained in these features aligns directly with the logits of the next token. However, predicting the next-next token based solely on top-layer features—which are inherently limited to the next token—poses a significant challenge. Fortunately, the training-time test technique described above enables the use of features from intermediate layers instead of relying solely on the top layer, as the feature prediction loss l_{fea} has been removed during training.

This paper introduces EAGLE-3, an enhanced version of EAGLE that achieves a significant speedup:

- **A training-time test architecture for the draft model:** We remove the feature prediction constraint and directly predict tokens while simulating multi-step generation during training. This direct token prediction provides complete flexibility in the draft model's input. Instead of reusing only the top-layer features, we integrate and leverage low-, mid-, and high-level features from the target model, capturing rich semantic information from different layers.

- **A new scaling law for inference acceleration in LLMs:** With the new architecture, we observe that increasing the amount of training data for the draft model leads to a proportional increase in the speedup ratio of EAGLE-3. This scaling behavior was not observed in the original EAGLE architecture, as shown in Figure 2

- **Improved inference acceleration:** EAGLE-3, trained with approximately 8x more data than EAGLE, achieves a 1.4x latency speedup over EAGLE-2 at batch size 1. Speculative sampling is often thought to reduce throughput at large batch sizes. However, in SGLang [9], a production-grade framework, EAGLE-3 improves throughput by 38% at a batch size of 64. We expect larger data size would lead to further improved speedup ratio.

2 Preliminaries

2.1 Speculative Sampling

Speculative sampling [10, 11, 12, 13] is a lossless LLM acceleration technique that alternates between drafting and verification, where drafting is performed at low cost and verification is parallelized, corresponding to the generation of drafts and the verification process, respectively. We use t_i to denote the i-th token and $T_{a:b}$ to represent the token sequence $t_a, t_{a+1}, \cdots, t_b$. When $T_{1:j}$ is used as the prefix, the two stages of speculative sampling are as follows.

In the drafting stage, speculative sampling utilizes a draft model (a smaller version from the same series as the target model) to autoregressively generate k tokens to form the draft. $\hat{T}_{j+1:j+k}$, while also recording the probability $\hat{p}$ for each token.

In the verification stage, speculative sampling invokes the target model to evaluate the draft $\hat{T}_{j+1:j+k}$ and records its probability p. Speculative sampling then determines the acceptance of draft tokens sequentially, from front to back. For token $\hat{t}_{j+i}$, the probability of acceptance is given by $\min(1, p_{j+i}(\hat{t}_{j+i})/\hat{p}_{j+i}(\hat{t}_{j+i}))$. If the token is accepted, the process moves to the next token. Otherwise, a token is sampled from the distribution $\text{norm}(\max(0, p_{j+i} - \hat{p}_{j+i}))$ to replace $\hat{t}_{j+i}$, and the remaining tokens in the draft are discarded. Appendix A.1 of [10] proves that speculative sampling is consistent with the distribution of vanilla autoregressive decoding.

2.2 EAGLE and EAGLE-2

The draft model with limited capacity struggles to precisely approximate the large-scale target model. EAGLE leverages the top-layer features of the target model as additional information and performs autoregression at the feature level, simplifying the drafting process. EAGLE performs autoregression at the feature level and then uses the LM head of the target model to obtain the draft token. Due to the sampling results at the token layer being hidden, feature-level autoregression introduces uncertainty. EAGLE addresses this issue by feeding the token sequence from the previous time step, i.e., the sampling results, into the draft model. Unlike the chain-like drafts of Vanilla speculative sampling, EAGLE generates multiple draft tokens at the same position, resulting in a tree-like draft. In the verification stage, EAGLE uses tree attention to parallelize the verification of the draft tree. Interestingly, EAGLE inspired the *multi-token prediction* technique used in the pre-training of DeepSeek-v3 [14], which in turn inspired new architectural designs in EAGLE-3.

EAGLE [2] and Medusa [8], among others, use tree-shaped drafts, where the structure of the draft tree is predefined, static, and context-independent. The difficulty of drafting is closely related to the context, and a static draft tree can lead to resource wastage. EAGLE-2 [15] approximates the acceptance rate using the confidence of the draft model and dynamically generates the draft tree based on this, performing pruning of the draft tree at the end of the drafting stage. EAGLE-3 also adopts the context-aware dynamic draft tree proposed in EAGLE-2.

3 EAGLE-3

In this section, we provide a detailed description of the implementation of EAGLE-3.

3.1 Inference Pipeline

Consistent with other speculative sampling methods, EAGLE-3 alternates between the drafting and verification stages. The difference between EAGLE-3 and EAGLE lies in the drafting stage, which we introduce with an example, as shown in Figure 5. Consider the prefix "How can". During the prefill phase or the previous verification stage, the target model performs a forward pass to generate the next token, "I". We record the low, middle, and high-level feature sequences from the target model's forward pass, denoted as l, m, and h, respectively. We concatenate the k-dimensional vectors l, m, and h to form a $3k$-dimensional vector, then pass it through a fully connected (FC) layer to reduce it to k-dimensions, obtaining a feature g that integrates information from different layers. Here, k refers to the hidden size of the target model.

Our goal is to generate a draft token sequence with the prefix "How can I". By inputting only g_{how} and g_{can}, the draft model cannot access the random sampling process. Therefore, similar to EAGLE [2],

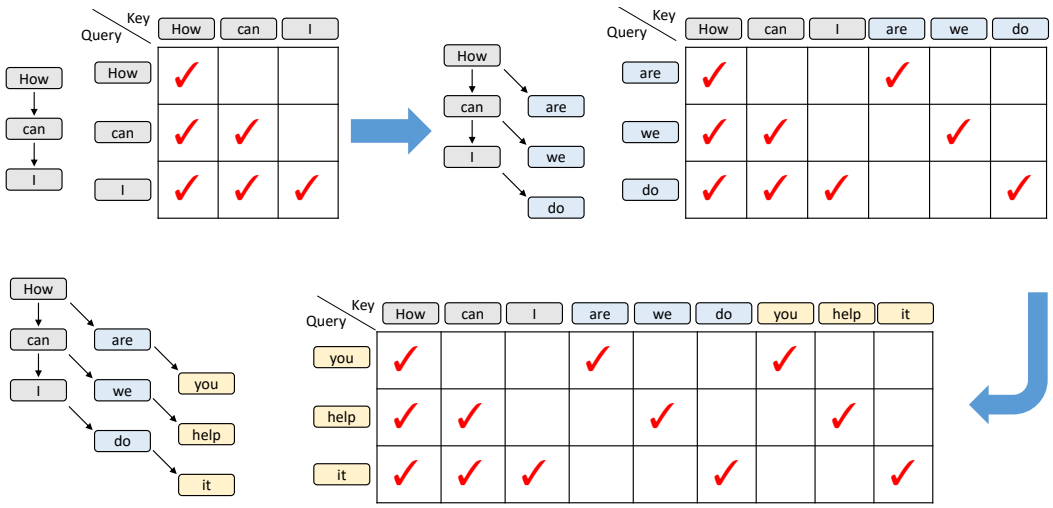

Figure 6: Diagram of the attention causal masks during training-time test. It sequentially shows a native training step (the first step) and two simulated training steps (the second and third steps). The arrows between tokens represent contextual relationships. The gray tokens represent the training data while the blue and yellow tokens represent the first- and second-round predictions by the draft model, respectively. We use the training dataset as the labels for each token position.

we introduce the embedding e_{I} of the sampled token "I". The concatenated vector is then passed through an FC layer to reduce its dimensionality to k, and subsequently inputted into a single layer decoder, producing the output a. Finally, we input a_{I} into the LM head and sample to obtain the draft token "do".

In Step 1, with the prefix "How can", we reuse g_{how} and g_{can} from the target model. In Step 2, the prefix becomes "How can I". Ideally, we would reuse g_{how}, g_{can}, and g_{I} from the target model. However, this is not possible because the token "I" has not yet been checked by the target model, and we cannot obtain g_{I}. Instead, we use the output a_{I} from the draft model in the previous step to replace g_{I}, and concatenate a_{I} with the embedding e_{do} of the sampled result "do" as the input to the draft model in Step 1. In Step 3, we similarly cannot obtain g_{do}, so we use a_{do} as a replacement, concatenating a_{do} with e_{it} as the input to the draft model. The same approach is followed for subsequent steps.

3.2 Draft Model Training

The input to the draft model in EAGLE is either, or at least approximately, the top-layer features $f_1, f_2, \cdots, f_t$ of the target model. In contrast, the input to the draft model in EAGLE-3 may include the features $g_1, g_2, \cdots, g_t$ from the target model, or it may include the output $a_{t+1}, a_{t+2} \cdots, a_{t+j}$ from the draft model. Therefore, we need to train the draft model to adapt to different inputs. During training, we perform test steps, where we generate a and feed it back into the draft model for further training.

The core of the draft model in EAGLE-3 is a Transformer decoder layer. Aside from the self-

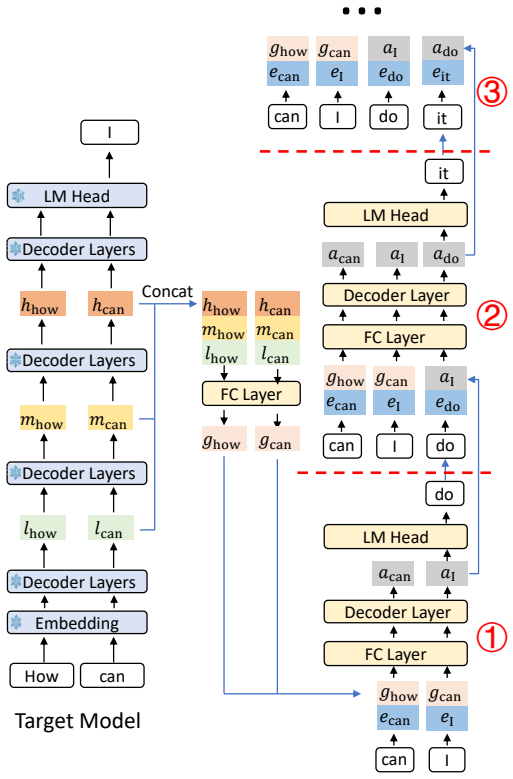

Figure 5: Diagram of the EAGLE-3 inference pipeline. l, m, and h represent the low, middle, and high-level features of the target model, respectively. e denotes the embedding.

attention operation, no other components interact with the context, so no further modifications are required during training or testing. The only component that requires slight modification is the self-attention, which we will describe in detail below.

Although the actual input consists of features, for clarity, we describe the process using tokens as input. As shown in Figure 6, the original training data is a sequence of length 3, "How can I", with a normal sequential dependency in the context. Therefore, the attention mask is a standard lower triangular matrix. The outputs at the three positions are "are", "we", and "do", which have a tree-like contextual relationship with "how", "can", and "I". As a result, when the input "are", "we", and "do" is fed into Step 2, the attention mask needs to be adjusted accordingly, as shown in the top-right corner of Figure 6. All attention masks are diagonal, except when the original training data is used as the key. Using matrix multiplication in this case would result in significant computational waste, so we can use vector dot products to calculate the attention score only for the corresponding positions.

HASS [3] and EAGLE-3 both make similar modifications to the attention mechanism to simulate the testing process during training, but this is not the main focus of EAGLE-3. The motivations, methods, and outcomes of the two approaches are distinctly different. The motivation behind HASS is to mitigate the error accumulation caused by inaccurate feature predictions in EAGLE. HASS still performs feature prediction, includes a feature prediction loss l_{fea}, and the input to the draft model must be the top-layer features. In contrast, the motivation behind EAGLE-3 is to remove unnecessary constraints to enhance the model's expressive power. EAGLE-3 no longer requires the draft model's output to fit the top-layer features of the target model, thus avoiding error accumulation. After removing feature prediction, the input to EAGLE-3 is completely free, and it is replaced by a fusion of features from different layers of semantic information. The removal of the feature prediction loss also enables us to discover a new scaling law for inference acceleration which was never found before. Figure 1 and Table 1 also shows the speedup of EAGLE-3 and HASS, with EAGLE-3 demonstrating significantly better performance.

4 Experiments

Models. We conduct experiments with open-source chat and reasoning models, including Vicuna 13B [16], LLaMA-Instruct 3.1 8B, LLaMA-Instruct 3.3 70B [7], and DeepSeek-R1-Distill-LLaMA 8B [17]. Due to the GPU constraint, we are unable to test EAGLE-3 on models larger than 70B.

Draft Models. Same to EAGLE and EAGLE-2, the draft model of EAGLE-3 consists of a single transformer layer. So, the scale of draft models in EAGLE, EAGLE-2, and EAGLE-3 is nearly the same.

Tasks. Following EAGLE [2] and Spec-Bench [18], we evaluate on five tasks, using the same weights for all tasks without fine-tuning on the respective tasks. For multi-turn conversation, code generation, mathematical reasoning, instruction following, and summarization,, we chose the MT-bench [19], HumanEval [20], GSM8K [21], Alpaca [22], and CNN/Daily Mail [23] datasets, respectively.

Metrics. EAGLE-3 does not modify the target model's weights and uses strict speculative sampling acceptance conditions, ensuring no loss in performance. Therefore, we do not evaluate generation quality. Instead, we use the following metrics to assess the acceleration performance:

- **Speedup Ratio:** The actual test speedup ratio relative to vanilla autoregressive decoding.
- **Average Acceptance Length τ:** The average number of tokens generated per drafting-verification cycle, which corresponds to the number of tokens accepted from the draft.
- **Acceptance Rate n-α:** The proportion of draft tokens accepted, which directly reflects the draft model's approximation to the target model. Following EAGLE's setup, we use a chain-like draft rather than a tree-like draft when testing acceptance rates. EAGLE suffers from error accumulation, meaning that the input to the draft model may be its own estimates rather than the exact values from the target model. Therefore, EAGLE uses n-α to represent the acceptance rate when the input contains n estimated features, under the condition that the previous estimated tokens are all accepted by the target model. In other words, the acceptance rate for inputs $f_1, f_2, \cdots, f_i, \hat{f}_{i+1}, \cdots, \hat{f}_{i+n}$, where f is the exact value and $\hat{f}$ is the draft model's estimate. Similarly, we use n-α to represent the acceptance rate in EAGLE-3 when the input contains n self-predicted values a, i.e., the acceptance rate for inputs $g_1, g_2, \cdots, g_i, a_{i+1}, \cdots, a_{i+n}$, where g is the fused feature from the target model.

Table 1: Speedup ratios and average acceptance lengths τ of different methods on A100 GPUs. V represents Vicuna, L31 represents LLaMA-Instruct 3.1, L33 represents LLaMA-Instruct 3.3, and DSL represents DeepSeek-R1-Distill-LLaMA. SpS denotes standard speculative sampling, with its draft model being Vicuna-68M. Methods like Medusa relax acceptance conditions under non-greedy settings, which do not guarantee lossless acceleration. Therefore, we do not compare EAGLE-3 with these methods when temperature=1.

		MT-bench		HumanEval		GSM8K		Alpaca		CNN/DM		Mean	
Model	Method	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ
Temperature=0													
V 13B	SpS	1.93x	2.27	2.23x	2.57	1.77x	2.01	1.76x	2.03	1.93x	2.33	1.92x	2.24
	PLD	1.58x	1.63	1.85x	1.93	1.68x	1.73	1.16x	1.19	2.42x	2.50	1.74x	1.80
	Medusa	2.07x	2.59	2.50x	2.78	2.23x	2.64	2.08x	2.45	1.71x	2.09	2.12x	2.51
	Lookahead	1.65x	1.69	1.71x	1.75	1.81x	1.90	1.46x	1.51	1.46x	1.50	1.62x	1.67
	Hydra	2.88x	3.65	3.28x	3.87	2.93x	3.66	2.86x	3.53	2.05x	2.81	2.80x	3.50
	EAGLE	3.07x	3.98	3.58x	4.39	3.08x	3.97	3.03x	3.95	2.49x	3.52	3.05x	3.96
	EAGLE-2	4.26x	4.83	4.96x	5.41	4.22x	4.79	4.25x	4.89	3.40x	4.21	4.22x	4.83
	EAGLE-3	**5.58x**	**6.65**	**6.47x**	**7.54**	**5.32x**	**6.29**	**5.16x**	**6.17**	**5.01x**	**6.47**	**5.51x**	**6.62**
L31 8B	EAGLE-2	3.16x	4.05	3.66x	4.71	3.39x	4.24	3.28x	4.12	2.65x	3.45	3.23x	4.11
	HASS	3.55x	4.41	3.78x	4.85	3.45x	4.47	3.57x	4.55	2.77x	3.55	3.42x	4.37
	EAGLE-3	**4.40x**	**6.13**	**4.85x**	**6.74**	**4.48x**	**6.23**	**4.82x**	**6.70**	**3.65x**	**5.34**	**4.44x**	**6.23**
L33 70B	EAGLE-2	2.83x	3.67	3.12x	4.09	2.83x	3.69	3.03x	3.92	2.44x	3.55	2.85x	3.78
	EAGLE-3	**4.11x**	**5.63**	**4.79x**	**6.52**	**4.34x**	**6.15**	**4.30x**	**6.09**	**3.27x**	**5.02**	**4.12x**	**5.88**
DSL 8B	EAGLE-2	2.92x	3.80	3.42x	4.29	3.40x	4.40	3.01x	3.80	3.53x	3.33	3.26x	3.92
	EAGLE-3	**4.05x**	**5.58**	**4.59x**	**6.38**	**5.01x**	**6.93**	**3.65x**	**5.37**	**3.52x**	**4.92**	**4.16x**	**5.84**
Temperature=1													
V 13B	SpS	1.62x	1.84	1.72x	1.97	1.46x	1.73	1.52x	1.78	1.66x	1.89	1.60x	1.84
	EAGLE	2.32x	3.20	2.65x	3.63	2.57x	3.60	2.45x	3.57	2.23x	3.26	2.44x	3.45
	EAGLE-2	3.80x	4.40	4.22x	4.89	3.77x	4.41	3.78x	4.37	3.25x	3.97	3.76x	4.41
	EAGLE-3	**4.57x**	**5.42**	**5.15x**	**6.22**	**4.71x**	**5.58**	**4.49x**	**5.39**	**4.33x**	**5.72**	**4.65x**	**5.67**
L31 8B	EAGLE-2	2.44x	3.16	3.39x	4.39	2.86x	3.74	2.83x	3.65	2.44x	3.14	2.80x	3.62
	HASS	2.58x	3.31	3.48x	4.50	2.87x	3.77	3.04x	3.98	2.42x	3.11	2.89x	3.73
	EAGLE-3	**3.07x**	**4.24**	**4.13x**	**5.82**	**3.32x**	**4.59**	**3.90x**	**5.56**	**2.99x**	**4.39**	**3.45x**	**4.92**
L33 70B	EAGLE-2	2.73x	3.51	2.89x	3.81	2.52x	3.36	2.77x	3.73	2.32x	3.27	2.65x	3.54
	EAGLE-3	**3.96x**	**5.45**	**4.36x**	**6.16**	**4.17x**	**5.95**	**4.14x**	**5.87**	**3.11x**	**4.88**	**3.95x**	**5.66**
DSL 8B	EAGLE-2	2.69x	3.41	3.01x	3.82	3.16x	4.05	2.64x	3.29	2.35x	3.13	2.77x	3.54
	EAGLE-3	**3.20x**	**4.49**	**3.77x**	**5.28**	**4.38x**	**6.10**	**3.16x**	**4.30**	**3.08x**	**4.27**	**3.52x**	**4.89**

Implementation. We use the AdamW optimizer, with beta values (β_1, β_2) set to (0.9, 0.95) and implemented gradient clipping of 0.5. The learning rate is set to 5e-5. We simulate 5 steps during training-time test. We use ShareGPT and UltraChat-200K [24] as training data, containing approximately 68K and 464K data entries, respectively. We call the target model to generate responses rather than using a fixed dataset. For the reasoning model DeepSeek-R1-Distill-LLaMA 8B, we also use the OpenThoughts-114k-math dataset for training. We use 16x A100 GPUs for the training of EAGLE-3 head for 70B models in two weeks. If not specified, we use the A100 GPU to test 70B models and the RTX 3090 for other models. The testing environment for all methods accelerating the same target model is identical.

Comparison. We use vanilla autoregressive decoding as the baseline, which serves as the benchmark for speedup ratios (1.00x). We compare EAGLE-3 with recent lossless speculative sampling methods, including standard speculative sampling [10, 11, 25], PLD [26], Medusa [8], Lookahead [27], Hydra [28], HASS [3], EAGLE [2], and EAGLE-2 [15].

4.1 Effectiveness

Figure 1 and Table 1 demonstrate the acceleration performance of EAGLE-3. On all tasks and target models, EAGLE-3 achieves the highest speedup ratio and average acceptance length. EAGLE-3 provides a speedup of approximately 3.0x-6.5x compared to vanilla autoregressive generation, with a 20%-40% improvement over EAGLE-2. Different tasks affect the draft model's acceptance rate, so both the average acceptance length and speedup ratio are task-dependent. Due to the presence of many fixed templates in code generation tasks, generating drafts is the easiest, which is why EAGLE-3 performs best on HumanEval, achieving a speedup ratio of up to 6.5x and an average acceptance length of up to 7.5. DeepSeek-R1-Distill-LLaMA 8B is an exception, with the highest

speedup ratio on the mathematical reasoning dataset GSM8K. This may be because we trained the draft model of DeepSeek-R1-Distill-LLaMA 8B using the OpenThoughts-114k-math dataset.

Figure 7 shows the acceptance rates of EAGLE and EAGLE-3 on MT-bench with LLaMA-Instruct 3.1 8B as the target model. The acceptance rate of EAGLE-3 is significantly higher than that of EAGLE. As the input from the draft model itself increases, the acceptance rate of EAGLE drops significantly, whereas EAGLE-3's acceptance rate remains almost unchanged, demonstrating the effectiveness of the Training-time test.

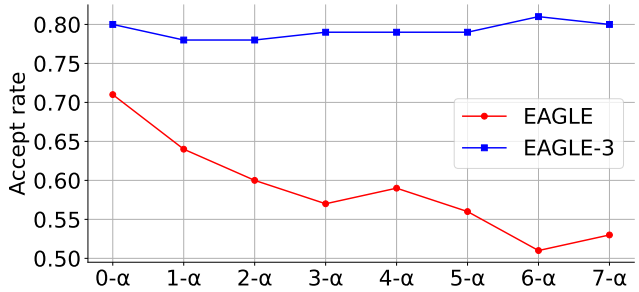

Figure 7: Acceptance rate of EAGLE and EAGLE-3 on MT-bench, with the target model being LLaMA-Instruct 3.1 8B. Hereby, n-α refers to the acceptance rate when the input contains n estimated features, under the condition that the previous estimated tokens are all accepted by the target model.

4.2 Ablation Study

The improvements of EAGLE-3 mainly come from three aspects: first, the removal of the feature regression constraint, second, the improvement from reusing only the top-layer features to reusing a mix of low, middle, and high-level features, and third, the increase of training data. We conducted an ablation study on MT-bench with LLaMA-Instruct 3.1 8B as the target model. The results, shown in Table 2, indicate that the first and second improvements in EAGLE-3 significantly enhance the acceptance length and speedup ratio, demonstrating the rationality of the EAGLE-3 design. Figure 2 shows how the speedup ratio increases w.r.t. the amount of training data. The new architectural designs in EAGLE-3 enable an increasing scaling curve, which was never observed in the previous works.

Table 2: Ablation study results with LLaMA-Instruct 3.1 8B as the target model. "Remove fea con" refers to the first improvement of EAGLE-3, which removes the feature prediction constraint. "Fused features" refers to the second improvement of EAGLE-3, where low, middle, and high-level feature fusion replaces the use of top-layer features.

	MT-bench		GSM8K	
Method	Speedup	τ	Speedup	τ
EAGLE-2	3.16x	4.05	3.39x	4.24
+ remove fea con	3.82x	5.37	3.77x	5.22
+ fused features (ours)	4.40x	6.13	4.48x	6.23

4.3 EAGLE-3 in SGLang

Speculative sampling algorithms reduce memory accesses and lower latency during memory-bound decoding by leveraging redundant computational power. As batch sizes increase, this redundancy decreases, reducing the effectiveness of speculative sampling. Efficiency improvements are more challenging in highly optimized production-grade frameworks. The performance of EAGLE-3 for large batches on a single H100 GPU and LLaMA-Instruct 3.1 8B in the SGLang v0.4.4 environment [9] was evaluated in Table 3. This part of the experiment did not use the tree structure, the chain length was set to 3, and the testing dataset was MT-Bench. EAGLE reduces throughput at batch size of 24, whereas EAGLE-3 still achieves a 38% throughput improvement at a batch size of 64.

Table 3: Throughput improvement under different batch sizes on H100 and LLaMA-Instruct 3.1 8B for the MT-Bench dataset, with SGLang without speculative sampling as the baseline (1.00x).

Batch size	2	4	8	16	24	32	48	56	64
EAGLE	1.40x	1.38x	1.23x	1.02x	0.93x	0.94x	0.88x	0.99x	0.99x
EAGLE-3	1.81x	1.82x	1.62x	1.48x	1.39x	1.32x	1.38x	1.34x	1.38x

We also tested the throughput of EAGLE-3 at batch size = 1 on H100 when the target model is LLaMA-Instruct 3.1 8B and the testing dataset is MT-bench. The results are shown in Table 4.

Table 4: Throughput at batch size = 1 on a single H100 GPU when the target model is LLaMA-Instruct 3.1 8B and the testing dataset is MT-bench.

Method	Throughput (bs=1)
SGLang (w/o speculative, 1x H100)	158.34 tokens/s
SGLang + EAGLE-2 (1x H100)	244.10 tokens/s
SGLang + EAGLE-3 (1x H100)	373.25 tokens/s

4.4 EAGLE-3 in vLLM

We also conducted a study on the impact of EAGLE-3 on throughput for large batch sizes based on vLLM [29], a widely used production-grade framework, and the results on RTX3090 and LLaMA-Instruct 3.1 8B are shown in Table 5. EAGLE shows the maximum throughput improvement at a batch size of 24, while EAGLE-3 shows this at 56. This part of the experiment did not use the tree structure, the maximum chain length was set to 2, and the testing dataset was MT-Bench.

Table 5: Throughput improvement under different batch sizes on RTX3090 and LLaMA-Instruct 3.1 8B for the MT-Bench dataset, with vLLM without speculative sampling as the baseline (1.00x).

Batch size	2	4	8	16	24	32	48	56
EAGLE	1.30x	1.25x	1.21x	1.10x	1.03x	0.93x	0.82x	0.71x
EAGLE-3	1.75x	1.68x	1.58x	1.49x	1.42x	1.36x	1.21x	1.01x

5 Related Work

Many methods have been used to accelerate inference in LLMs, such as quantization [30, 31, 32, 33, 34] and distillation [35]. These methods generally have trade-offs, where there is a need to balance model performance with acceleration benefits.

Speculative sampling uses the target model for verification to ensure lossless acceleration. Early speculative decoding methods [36, 37] accelerated generation in greedy settings, while [10, 11] introduced speculative sampling to extend the draft verification framework to non-greedy generation. Many subsequent works have improved upon speculative sampling. EAGLE [2], EAGLE-2 [15], Medusa [8], and Hydra [28] reused the features of the target model. Based on the framework of EAGLE, HASS [3] simulates a multistep draft process during training to mitigate the issues of training-inference inconsistency and error accumulation in EAGLE. GLIDE and CAPE [38] reuse the target model's KV cache, while methods [39, 40, 41, 42, 43, 44, 45, 46, 47] like Draft & Verify [48] use layer skipping or early exits to reuse parts of the target model's parameters.

There are several key differences between HASS [3] and EAGLE-3. First, HASS drafts using only top-layer features, whereas EAGLE-3 integrates low-, mid-, and high-level features. Second, HASS retains the token loss l_{token}, while EAGLE-3 removes it to improve model capacity. Third, unlike HASS, EAGLE-3 exhibits a clear scaling law trend. Finally, EAGLE-3 significantly outperforms HASS, as demonstrated in Figure 1 and Table 1.

6 Conclusion

In this paper, we introduce EAGLE-3. Building upon EAGLE, EAGLE-3 incorporates two key improvements. First, it removes the feature prediction constraint, instead directly predicting draft tokens through a Training-time test. Second, it replaces the use of the target model's top-layer features with a fusion of the target model's lower, middle, and upper-layer features to obtain richer information. With these improvements, EAGLE-3 continues to benefit from the augmentation of training data, achieving a maximum speedup of 6.5x.

Acknowledgement

We would like to thank James Liu, Ke Bao, Yineng Zhang, Lianmin Zheng, Ying Sheng, and many others in the SGLang team for evaluating EAGLE-3 in the SGLang environment. Hongyang Zhang is supported by the NSERC Discovery Grant RGPIN-2022-03215, DGECR-2022-00357, and Google Research Scholar Award.

References

[1] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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

[3] Lefan Zhang, Xiaodan Wang, Yanhua Huang, and Ruiwen Xu. Learning harmonized representations for speculative sampling. *arXiv preprint arXiv:2408.15766*, 2024.

[4] Xiangxiang Gao, Weisheng Xie, Yiwei Xiang, and Feng Ji. Falcon: Faster and parallel inference of large language models through enhanced semi-autoregressive drafting and custom-designed decoding tree. *arXiv preprint arXiv:2412.12639*, 2024.

[5] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[6] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[7] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[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] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. SGLang: Efficient execution of structured language model programs. *Advances in Neural Information Processing Systems*, 37:62557–62583, 2024.

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

[11] 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.

[12] Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. SpecTr: Fast speculative decoding via optimal transport. *Advances in Neural Information Processing Systems*, 36, 2024.

[13] Ziteng Sun, Jae Hun Ro, Ahmad Beirami, and Ananda Theertha Suresh. Optimal block-level draft verification for accelerating speculative decoding. *arXiv preprint arXiv:2403.10444*, 2024.

[14] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. DeepSeek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

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

[16] 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. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023.

[17] Daya Guo DeepSeek-AI, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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

[19] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and chatbot Arena. *arXiv preprint arXiv:2306.05685*, 2023.

[20] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[21] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[22] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7, 2023.

[23] Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*, 2016.

[24] 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. *arXiv preprint arXiv:2305.14233*, 2023.

[25] Joao Gante. Assisted generation: a new direction toward low-latency text generation. *https://huggingface.co/blog/assisted-generation*, 2023.

[26] Apoorv Saxena. Prompt lookup decoding, November 2023.

[27] Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of LLM inference using lookahead decoding. *arXiv preprint arXiv:2402.02057*, 2024.

[28] Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. Hydra: Sequentially-dependent draft heads for Medusa decoding. *arXiv preprint arXiv:2402.05109*, 2024.

[29] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Symposium on Operating Systems Principles*, pages 611–626, 2023.

[30] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *journal of machine learning research*, 18(187):1–30, 2018.

[31] Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Q-bert: Hessian based ultra low precision quantization of bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8815–8821, 2020.

[32] Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. I-bert: Integer-only bert quantization. In *International conference on machine learning*, pages 5506–5518. PMLR, 2021.

[33] Ali Hadi Zadeh, Isak Edo, Omar Mohamed Awad, and Andreas Moshovos. Gobo: Quantizing attention-based nlp models for low latency and energy efficient inference. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 811–824. IEEE, 2020.

[34] Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. Q8bert: Quantized 8bit bert. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*, pages 36–39. IEEE, 2019.

[35] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[36] Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31, 2018.

[37] Xin Sun, Tao Ge, Furu Wei, and Houfeng Wang. Instantaneous grammatical error correction with shallow aggressive decoding. *arXiv preprint arXiv:2106.04970*, 2021.

[38] Cunxiao Du, Jing Jiang, Xu Yuanchen, Jiawei Wu, Sicheng Yu, Yongqi Li, Shenggui Li, Kai Xu, Liqiang Nie, Zhaopeng Tu, et al. Glide with a cape: A low-hassle method to accelerate speculative decoding. *arXiv preprint arXiv:2402.02082*, 2024.

[39] Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Hasan Genc, Kurt Keutzer, Amir Gholami, and Sophia Shao. Speed: Speculative pipelined execution for efficient decoding. *arXiv preprint arXiv:2310.12072*, 2023.

[40] Seongjun Yang, Gibbeum Lee, Jaewoong Cho, Dimitris Papailiopoulos, and Kangwook Lee. Predictive pipelined decoding: A compute-latency trade-off for exact llm decoding. *arXiv preprint arXiv:2307.05908*, 2023.

[41] Giovanni Monea, Armand Joulin, and Edouard Grave. Pass: Parallel speculative sampling. *arXiv preprint arXiv:2311.13581*, 2023.

[42] Minghan Li, Xilun Chen, Ari Holtzman, Beidi Chen, Jimmy Lin, Wen-tau Yih, and Xi Victoria Lin. Nearest neighbor speculative decoding for llm generation and attribution. *arXiv preprint arXiv:2405.19325*, 2024.

[43] Hanling Yi, Feng Lin, Hongbin Li, Peiyang Ning, Xiaotian Yu, and Rong Xiao. Generation meets verification: Accelerating large language model inference with smart parallel auto-correct decoding. *arXiv preprint arXiv:2402.11809*, 2024.

[44] Fangcheng Liu, Yehui Tang, Zhenhua Liu, Yunsheng Ni, Kai Han, and Yunhe Wang. Kangaroo: Lossless self-speculative decoding via double early exiting. *arXiv preprint arXiv:2404.18911*, 2024.

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

[46] Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layer skip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*, 2024.

[47] Ruslan Svirschevski, Avner May, Zhuoming Chen, Beidi Chen, Zhihao Jia, and Max Ryabinin. Specexec: Massively parallel speculative decoding for interactive llm inference on consumer devices. *arXiv preprint arXiv:2406.02532*, 2024.

[48] Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. Draft & verify: Lossless large language model acceleration via self-speculative decoding. *arXiv preprint arXiv:2309.08168*, 2023.

A Implementation Details

Vanilla: We use models from the Huggingface.transformers library with the PyTorch backend and pre-allocated KV cache. Other methods also use these models as their base.

(Standard) Speculative Sampling: We use the assisted generation feature from the HuggingFace Transformers library.

PLD, Lookahead, Medusa, and Hydra: We use the default settings and the officially released weights.

EAGLE: Vicuna and LLaMA2-Chat draft models use the officially released weights, while LLaMA3-Instruct is trained using the ShareGPT dataset (consistent with Medusa and Hydra).

EAGLE-2: For the 7B (8B), 13B, and 70B original LLMs, we set the total number of draft tokens to 60, 50, and 48, respectively, with a draft tree depth of 6, and select 10 nodes during the expansion phase.

EAGLE-3: EAGLE-3's draft model achieves a significantly higher acceptance rate, allowing us to increase the draft tree depth from 6 to 8 while keeping the number of nodes the same as in EAGLE-2.

B A Comparative Study of EAGLE-3 and HASS

The work most similar to EAGLE-3 is HASS. Both approaches simulate multi-step prediction during training, but this is neither the main focus of EAGLE-3 nor HASS. Training-time testing primarily involves adjusting the attention mask to enforce correct dependencies, which essentially simplifies tree attention into a fixed-shape form (as illustrated in Figure 6). In fact, tree attention has been widely adopted in nearly all speculative decoding methods proposed in recent years. Feeding model outputs instead of ground truth during training, known as scheduled sampling, was also widely explored in the RNN era.

The core contribution of HASS lies in identifying the train-test mismatch in EAGLE and mitigating it through tree attention-based simulation. In contrast, EAGLE-3 focuses on a different issue: the inability of EAGLE to benefit from data scaling. EAGLE-3 attributes this limitation to the feature prediction constraint—an issue also present in HASS. EAGLE-3 removes this constraint and uses tree attention for simulation. This modification enables EAGLE-3 to scale effectively with increased training data, whereas HASS does not. The ability to scale with data is the core contribution of EAGLE-3.

Figure 8 illustrates the performance of EAGLE-3 and HASS across different training data scales. Similar to EAGLE-2, HASS fails to scale up, while EAGLE-3 exhibits rapid performance improvements as more training data becomes available. Moreover, EAGLE-3 identifies that the top-layer features (used in EAGLE and subsequent works including HASS) tend to overfit to next-token prediction and are not well-suited for multi-step draft generation. To address this, EAGLE-3 replaces the top-layer features with a fusion of multi-level features. Therefore, EAGLE-3 also outperforms HASS when trained on smaller datasets.

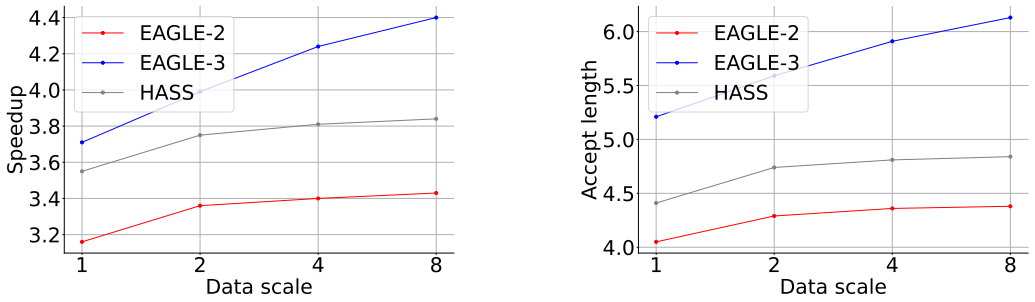

Figure 8: Scaling law evaluated on the MT-bench using LLaMA-Instruct 3.1 8B as the target model, with the x-axis representing the data scale relative to ShareGPT.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Please see abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Please see Section 6.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper does not include theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: See Section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: See supplementary material.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: See Section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: This field of study does not include statistical significance in its experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research described in this paper fully complies with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Section 6.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This paper poses no such risks.

    Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: See Section 4.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: This paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.