

SPEED: Specialized Position Experts for Efficient Speculative Decoding

Langlin Huang¹, Chengsong Huang¹, Jixuan Leng², Di Huang¹, Jiaxin Huang¹
 Washington University in St. Louis¹ Carnegie Mellon University²
 {h.langlin, jiaxinh}@wustl.edu

Abstract

Speculative decoding accelerates Large Language Model (LLM) inference by using a small draft model to predict multiple tokens and a large target model to verify them in parallel. Recent studies leverage features of target and draft models to enhance draft accuracy, but suffer from degraded accuracy of draft tokens at later positions due to accumulated deviations in draft model-generated features. Worse still, features of inconsistent levels of deviation provide a noisy training signal. In this paper, we propose **SPEED**, applying specialized position experts to draft tokens at preassigned positions. Position experts substantially improve the acceptance rate at later draft positions, as each expert only needs to focus on handling a certain level of feature deviation. Experiment results on six datasets demonstrate that **SPEED** effectively outperforms baselines on average acceptance length and speed-up ratio. Our codebase is available at <https://github.com/shrango/PosS>.

1 Introduction

Speculative decoding (Leviathan et al., 2022; Chen et al., 2023) is an effective approach to accelerate the autoregressive decoding of Large Language Models (LLMs) through a draft-then-verify framework. Specifically, it employs a lightweight draft model to generate candidate tokens autoregressively, which are then verified in parallel by the larger target model to determine which candidate tokens are accepted, thereby reducing overall decoding time. The effectiveness of speculative decoding largely depends on the average acceptance length τ from draft sequence length L .

Recent efforts (Cai et al., 2024; Li et al., 2024a;b; 2025c;a) in speculative decoding use model hidden states from previous contexts, called features, as part of the input to improve draft prediction accuracy. EAGLE (Li et al., 2024a;b; 2025c) employs a one-layer Transformer as the draft model and trains it to predict the next token with features from the target model. However, EAGLE exhibits a training-inference discrepancy: the draft model is trained only on target model features, but at inference uses draft model features instead when target features are unavailable. HASS (Zhang et al., 2024) and EAGLE-3 (Li et al., 2025c) address this discrepancy by training the draft model to predict the next token with features from previous draft steps, aligning with the inference stage. However, these approaches involve a noisy training signal because the input to the draft model is from diverse sources, but they use a single set of model parameters to handle such inconsistent input.



Figure 1: Position-wise acceptance rate (pos-acc) of different drafting methods, grouped by EAGLE-2 and EAGLE-3 framework.

As draft model predictions deviate from the target model’s predictions, the reliability of draft features decreases. Moreover, the deviated draft features are used to predict the next positions, thereby accumulating feature deviations in a chain reaction. To evaluate draft accuracy across positions, we introduce the position-wise acceptance rate (pos-acc) metric, which measures the conditional probability of accepting the i^{th} token given the acceptance of the preceding $(i - 1)^{\text{th}}$ token. Figure 1 reveals the degrading tendency of pos-acc.

To address this challenge, we propose Specialized Position Experts for Efficient Speculative Decoding (**SPEED**), a novel framework comprising multiple position-specialized draft layers. Each position expert is trained to predict tokens at its assigned positions and only needs to handle a limited, expected level of feature deviation. This disentanglement enables more accurate draft-token predictions than a single draft model, which must handle varying degrees of feature deviation across positions.

We conduct extensive experiments on all six Spec-Bench (Xia et al., 2024) benchmark datasets and demonstrate that **SPEED** consistently outperforms baseline methods. On average of the test datasets, **SPEED** surpasses the strong baseline EAGLE-3 on average acceptance length by 9.2% (from 4.69 to 5.12) and on speed-up ratio by up to 10.5% (from 2.96x to 3.27x). We also carry out a comprehensive analysis and reveal that **the efficiency of SPEED comes from reduced rounds of speculative generation**, as a higher position-wise acceptance rate at deeper positions enables longer acceptance length τ per round.

Our primary contributions include:

- We introduce position-wise acceptance rate (pos-acc) as a crucial metric for analyzing the draft quality of speculative decoding approaches.
- We propose **SPEED**, a novel framework that employs position-specialized layers to address the challenge of accumulated levels of feature deviation in draft predictions.
- We conduct extensive experiments and analysis to demonstrate that **SPEED** outperforms baseline methods on both average acceptance length and speed-up ratio.

2 Preliminary

2.1 Speculative Decoding

Speculative decoding harnesses the principle of speculative execution (Kung & Robinson, 1979), where a smaller, faster draft model θ_D works alongside a larger target language model θ_T that we aim to accelerate. The standard speculative decoding (Leviathan et al., 2022) operates in two key stages. In the draft stage, draft model θ_D autoregressively generates a candidate sequence of length L . In the verification stage, target model θ_T evaluates all L draft tokens in parallel with a single forward pass, where tokens that align with the target distribution are accepted.

2.2 Hidden State Assisted Speculative Decoding

Recent research (Cai et al., 2024; Li et al., 2024a;b; 2025c) discovers the potential of model hidden states, referred to as features. With the help of rich information from target model features, a few extra layers, rather than a complete auxiliary model, suffice for effective draft generation. Medusa (Cai et al., 2024) uses multiple language model heads to project a feature vector into different output spaces to predict several subsequent tokens simultaneously. EAGLE-1,2 (Li et al., 2024a;b) concatenate input token embedding with feature vectors. EAGLE-3 (Li et al., 2025c) replaces features from the last-layer hidden states with those from low, middle, and high-level layers. EAGLE family employs a one-layer Transformer as the draft model, θ_D , and reuses the target model’s LM head for token prediction.

Figure 2 provides an example of EAGLE at the inference stage. At generation step t , EAGLE’s draft model θ_D predicts the next token x_{t+1} based on context $x_{\leq t}$ and features $f_{<t}$:

$$P(x_{t+1}) = \text{Head}(\theta_D([x_t; f_{t-1}^{(T)}], [x_{t-1}; f_{t-2}^{(T)}], \dots, [x_1; f_0^{(T)}])) \quad (1)$$

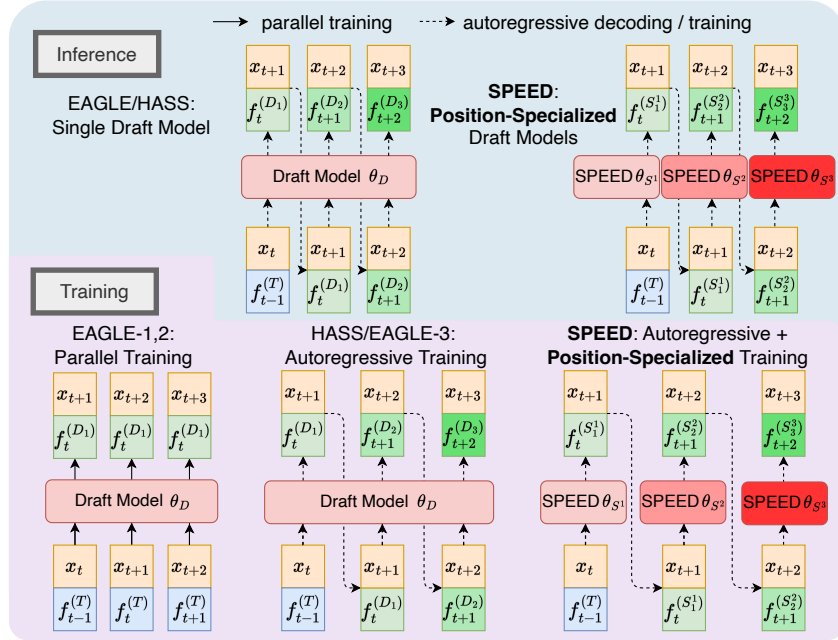


Figure 2: The inference and training stages of EAGLE, HASS, and **SPEED**. The dashed lines represent autoregressive decoding or training, and the solid lines represent parallel training. The input concatenates context word embeddings x and features from previous step f . Draft features in different colors are not in the same hidden space. EAGLE-1,2 exhibits a training-inference discrepancy. HASS and EAGLE-3 address the discrepancy, but train one set of parameters on varied inputs, involving a noisy training signal. **SPEED** further addresses this by applying multiple specialized position experts to handle inputs with large differences. This figure presents the main idea of **SPEED**, where one layer is assigned one position. In practice, one **SPEED** layer is assigned three positions.

Then, θ_D autoregressively generates draft tokens x_{t+2}, x_{t+3} . Notably, inputs are derived from different sources, denoted by superscripts: $f^{(T)}$ represents a feature from the target model; $f^{(D_i)}$ represents a feature from the i^{th} draft step of the draft model D . $f^{(D)}$ is used instead of $f^{(T)}$ when the target model features are unavailable, before the forward pass completion of subsequent tokens. Therefore, the k^{th} draft position is predicted by:

$$P(x_{t+k}) = \text{Head}\left(\theta_D([x_{t+k-1}; f_{t+k-2}^{(D_{k-1})}], \dots, [x_{t+1}; f_t^{(D_1)}], [x_t; f_{t-1}^{(T)}], \dots, [x_1; f_0^{(T)}])\right) \quad (2)$$

Specifically, Eq. (2) degenerates to Eq. (1) when $k = 1$.

Although EAGLE-1,2 performs inference with Eq. (2), it is solely trained on Eq. (1).

This exhibits a fundamental training-inference discrepancy: θ_D needs to predict the subsequent tokens ($k > 1$) with its own generated features during inference, but it never observes its own prediction errors during training, which impairs the prediction accuracy.

HASS and EAGLE-3 explicitly address the discrepancy by performing recursive feature alignment during training, as shown in the middle of Figure 2. Eventually, they improve the acceptance probabilities at later positions compared to EAGLE-2.

3 Method

In this section, we introduce our **SPEED** approach. We first introduce the concept of position-wise acceptance rate to reveal the fundamental limitations in existing approaches in Section 3.1. We then propose **SPEED** with position specialized training in Section 3.2 to address the limitation.

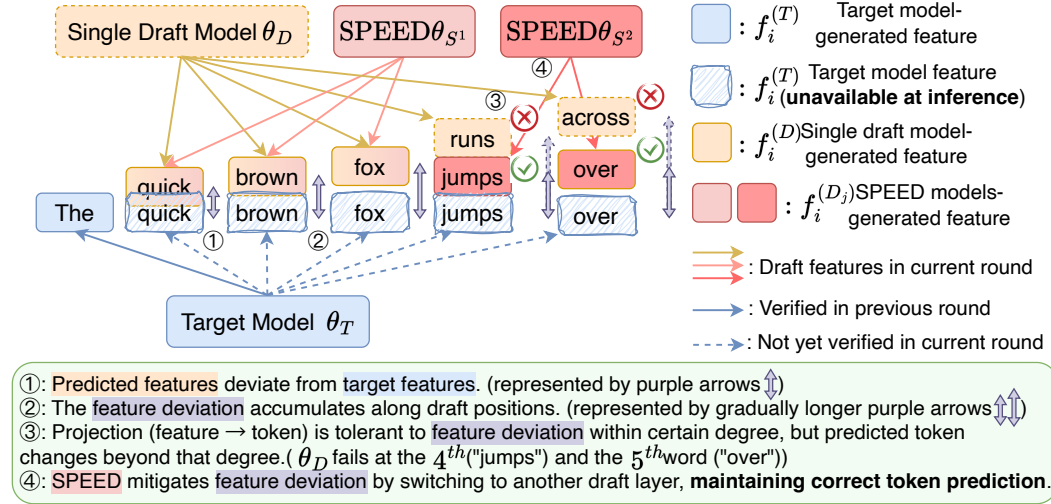


Figure 3: This figure illustrates how **SPEED** improves over single draft models like EAGLE and HASS at inference time. Draft model-generated features have two functions: generate draft tokens via LM head projection and serve as input to the next draft position. When generating draft tokens, a slight deviation between draft and target features does not change the predicted tokens. However, when biased features are used to predict the next positions, the existing deviation is amplified, causing larger deviations in subsequent steps and eventually leading to incorrect token predictions. **SPEED**, however, resets the deviation propagation to a low level by switching to another draft layer. As a result, **SPEED** maintains draft-token accuracy at later positions, thereby achieving a higher acceptance rate.

3.1 Position-Wise Acceptance Rate

Previous speculative decoding frameworks rely heavily on the generalizability of a single draft layer for multi-position token generation. EAGLE-1,2 trains θ_D only on the immediate next position but expects it to generalize to subsequent positions at inference time. While HASS and EAGLE-3 train θ_D on both the immediate and later positions, only one draft model is trained to generalize across diverse feature sources and different draft positions. As the draft model is a single Transformer layer, its generalizability is inherently limited.

To demonstrate the generalization limitation of EAGLE and HASS, we introduce **position-wise acceptance rate (pos-acc)**, which measures the probability that a token at position i is accepted given its preceding token at position $i - 1$ is accepted. The **pos-acc** is defined as:

$$\text{pos-acc}_i = \begin{cases} P(A_i), & i = 1, \\ P(A_i | A_{i-1}) = \frac{P(A_i)}{P(A_{i-1})}, & i > 1. \end{cases} \quad (3)$$

where A_i denotes the event that the token at position i is accepted in previous verifications. Notice that the target model acceptance follows a strict sequential dependency: if x_i is accepted, its preceding tokens $x_{[1:i-1]}$ must have been accepted, and therefore $A_i \subset A_{i-1}$.

We point out that higher pos-acc is crucial for achieving a higher acceptance length τ at each draft-verification round. For a draft sequence of length L , the probability of accepting all draft tokens up to position k ($k \leq L$) is:

$$P(A_k) = P(A_1 \cap A_2 \cap \dots \cap A_k) = \prod_{i=1}^k \text{pos-acc}_i, \quad (4)$$

This chain-rule decomposition reveals that the overall acceptance length depends on the product of **pos-acc** and is particularly sensitive to degradation at any single position. Notably, pos-acc at early positions contributes to all subsequent positions, so it is more important for overall performance.

Figure 1 exhibits the empirical pos-acc of EAGLE-2,3 and HASS. EAGLE-2’s pos-acc deteriorates rapidly for positions $k > 1$, because its draft model is trained solely to predict the next immediate token. HASS and EAGLE-3 can maintain relatively higher pos-acc at later positions because a single draft model is trained on multiple subsequent positions. However, their pos-acc at position $k = 1$ is lower than that of other methods by about 1% to 2% due to their compromise with other positions. This critically impairs the overall acceptance length due to the multiplicative nature of the acceptance probability in Eq.(4).

3.2 Specialized Position Experts

To address the aforementioned limitation, we introduce Specialized Position Experts for Efficient Speculative Decoding (**SPEED**) to preserve the early-position acceptance rate while enhancing later-position predictions. **SPEED** consists of multiple position-specialized draft layers, called position experts. Each expert is trained for certain positions and generates draft tokens at its preassigned positions. Therefore, the input and output hidden spaces are relatively fixed and easy to learn. In practice, an expert is assigned to 3 positions. Intuitively, features of neighboring draft positions exhibit similar levels of deviation, making it feasible for a single expert to handle. We discuss the choice of the number in Appendix B.

Figure 2 sketches **SPEED** with one-expert-one-position example. Each of the three position expert $\{\theta_{s_i}\}_{i=1}^3$ is assigned to predict its corresponding draft token x_{t+i} . Training follows exactly the same assignment, so training and inference are aligned. In practice, each expert is assigned to three consecutive positions.

Figure 3 illustrates the draft process at the inference stage, showing the deficit of a single draft model and how **SPEED** improves over it. All methods with EAGLE frameworks require the features from the target model for drafting. During the draft phase, the target model-generated features for most positions are unavailable because these draft tokens have not yet been verified by the target model. In this case, the input to the draft model is replaced with draft model-generated features, which inevitably deviate from the target features. Although the slightly deviated features may still predict the current token accurately, the deviation is propagated to the next draft position because these features also serve as input to the draft model for the next-token prediction. The feature deviation propagates and accumulates along draft positions until it becomes too big to yield a correct token. Existing work, HASS and EAGLE-3, overlooks the inter-position feature discrepancy and trains one model to handle varied inputs, as exhibited in Figure 2. However, the limited capacity of a single draft model prevents it from generalizing well across all inputs. As a result, they either perform worse at later or earlier positions.

The key point of **SPEED** is to improve draft models’ ability to handle various deviations. Unlike the single draft model, each position expert in **SPEED** is position-aware. The draft layers beyond the first one are trained only to make accurate predictions from biased feature inputs, enabling them to mitigate deviations accumulated in previous steps. Another benefit of position-specialized draft layers is that they avoid conflicting optimization directions. The lower pos-acc of EAGLE-3 at the first position is likely a result of training on all positions. Since it trains extensively on draft model-generated features with varying levels of deviation, it performs worse when the input consists only of target features. However, this is not a problem for **SPEED** because tasks with largely different optimization directions are distributed among different position specialists.

We further highlight that **SPEED** is orthogonal to EAGLE-2,3 frameworks. **SPEED** uses the same loss as HASS on EAGLE-2 framework and the same loss as EAGLE-3 on EAGLE-3 framework, but optimizes our designed draft model architecture. **SPEED** predicts draft token x_{t+k} with Eq. (5), which differs from Eq. (2) only in the superscripts.

$$P(x_{t+k}) = \text{Head}\left(\theta_{(S^{\lceil k/3 \rceil})}([x_{t+k-1}; f_{t+k-2}^{(S^{\lceil (k-1)/3 \rceil})}], \dots, [x_{t+1}; f_t^{(S^1)}], [x_t; f_{t-1}^{(T)}], \dots, [x_1; f_0^{(T)}])\right) \quad (5)$$

4 Experiment

4.1 Experiment Setup

Metrics. We evaluate model performance using two key metrics: the speed-up ratio and the average acceptance length, τ .

- **Speed-up Ratio:** The speed-up ratio measures the improvement in generation efficiency compared to the vanilla target model decoding, calculated as the ratio between throughputs (tokens generated per second) of a speculative decoding approach to that of the target model autoregressive decoding. A higher speed-up ratio indicates better performance.
- **Average Acceptance Length τ :** The average acceptance length represents the mean number of tokens accepted in each round of L drafting positions. It reflects how effectively the draft model can predict longer sequences that match the target model output. Longer acceptance lengths generally correlate with improved efficiency as they reduce the number of draft iterations needed.

Temperature=0								
Model	Method	MT-Bench	Alpaca	GSM8K	Natural Questions	CNN/DM	HumanEval	Avg.
L3 8B	EAGLE-2	4.11	4.32	4.25	3.38	3.61	4.70	4.06
	HASS	4.42	4.62	4.61	3.54	3.92	5.20	4.39
	Gumiho	4.27	4.19	4.59	3.58	3.84	5.18	4.28
	SPEED(E2)	4.52	4.82	4.81	3.64	4.05	5.41	4.54
L2 13B	EAGLE-2	4.86	4.64	5.01	4.15	4.30	5.78	4.79
	HASS	5.28	5.16	5.40	4.43	4.59	6.37	5.21
	Gumiho	4.78	4.55	4.97	4.13	4.41	5.82	4.78
	SPEED(E2)	5.33	5.17	5.48	4.52	4.70	6.43	5.27
L31 8B	EAGLE-3	4.73	5.07	4.89	3.71	4.18	5.55	4.69
	SPEED(E3)	5.15	5.50	5.43	4.13	4.54	5.95	5.12
Temperature=1								
L3 8B	EAGLE-2	3.83	4.15	4.09	3.18	3.39	4.50	3.86
	HASS	4.01	4.39	4.49	3.40	3.65	5.00	4.16
	Gumiho	3.90	3.95	4.33	3.32	3.59	4.84	3.99
	SPEED(E2)	4.13	4.46	4.67	3.37	3.76	5.12	4.25
L2 13B	EAGLE-2	4.69	4.44	4.82	4.12	4.25	5.54	4.64
	HASS	5.04	4.92	5.24	4.36	4.60	6.03	5.03
	Gumiho	4.57	4.40	4.80	4.03	4.25	5.66	4.62
	SPEED(E2)	5.12	4.98	5.39	4.35	4.54	6.15	5.09
L31 8B	EAGLE-3	4.31	4.62	4.75	3.45	3.85	5.30	4.38
	SPEED(E3)	4.66	4.98	5.24	3.81	4.14	5.69	4.75

Table 1: Average acceptance length τ of all methods. L3 8B represents Llama-3-8B-Instruct, L2 13B represents Llama-2-13B-Chat, L31 8B represents Llama-3.1-8B-Instruct.

Datasets. We conduct comprehensive experiments on Spec-Bench [Xia et al. \(2024\)](#), following EAGLE. This includes six datasets: MT-Bench ([Zheng et al., 2023](#)) for multi-turn conversation, Alpaca ([Taori et al., 2023](#)) for instruction following, GSM8K ([Cobbe et al., 2021](#)) for mathematical reasoning, Natural Questions ([Kwiatkowski et al., 2019](#)) for question answering, CNN/Daily Mail (shortened to CNN/DM) ([Nallapati et al., 2016](#)) for summarization, and HumanEval ([Chen et al., 2021](#)) for code generation.

Target Models. The experiments are conducted with three open-source models: Llama-3-8B-Instruct (L3 8B), Llama-2-13B-chat (L2 13B), and Llama-3.1-8B-Instruct (L31 8B).

Draft Methods. We evaluate the following methods for comparison. **EAGLE-2:** the base method in the EAGLE-2 framework, trained with a classification loss on the token and a regression loss on the feature. **HASS:** EAGLE-2 with recursive feature alignment training and a topk token distillation loss. **Gumiho** [Li et al. \(2025a\)](#): drafting the first two positions autoregressively like EAGLE and the following positions in parallel like Medusa. **SPEED(E2):** our method in EAGLE-2 framework, trained with HASS loss. **EAGLE-3:** the base method in EAGLE-3 framework. It trains using only the classification loss. **SPEED(E3):** our method in EAGLE-3 framework, trained with EAGLE-3 loss.

Temperature=0								
Model	Method	MT-Bench	Alpaca	GSM8K	Natural Questions	CNN/DM	HumanEval	Avg.
L3 8B	EAGLE-2	2.77x	2.79x	2.87x	2.29x	2.27x	3.08x	2.68x
	HASS	2.94x	2.97x	3.11x	2.38x	2.47x	3.48x	2.89x
	Gumiho	3.04x	2.97x	3.19x	2.58x	2.71x	3.69x	3.03x
	SPEED(E2)	2.96x	3.10x	3.17x	2.45x	2.50x	3.53x	2.95x
L2 13B	EAGLE-2	2.99x	2.95x	3.23x	2.71x	2.49x	3.71x	3.01x
	HASS	3.28x	3.34x	3.52x	2.96x	2.72x	4.15x	3.33x
	Gumiho	2.99x	2.92x	3.15x	2.70x	2.47x	3.79x	3.00x
	SPEED(E2)	3.28x	3.32x	3.59x	2.96x	2.74x	4.12x	3.34x
L31 8B	EAGLE-3	2.99x	3.11x	3.05x	2.34x	2.63x	3.62x	2.96x
	SPEED(E3)	3.35x	3.45x	3.41x	2.71x	2.84x	3.88x	3.27x
Temperature=1								
L3 8B	EAGLE-2	2.67x	2.55x	2.09x	2.02x	2.80x	2.47x	2.43x
	HASS	2.77x	2.79x	2.14x	2.09x	3.03x	2.56x	2.56x
	Gumiho	2.40x	2.47x	2.55x	2.09x	2.07x	2.92x	2.42x
	SPEED(E2)	2.71x	2.86x	2.12x	2.18x	3.11x	2.58x	2.59x
L2 13B	EAGLE-2	2.95x	2.88x	3.13x	2.76x	2.51x	3.48x	2.95x
	HASS	3.22x	3.30x	3.46x	2.97x	2.67x	3.89x	3.25x
	Gumiho	2.91x	2.88x	3.12x	2.70x	2.49x	3.60x	2.96x
	SPEED(E2)	3.22x	3.23x	3.49x	2.96x	2.73x	3.92x	3.26x
L31 8B	EAGLE-3	2.64x	2.65x	2.93x	2.08x	2.30x	3.10x	2.62x
	SPEED(E3)	2.90x	2.84x	3.11x	2.09x	2.49x	3.21x	2.77x

Table 2: Speed-up ratios of all methods. L3 8B represents Llama-3-8B-Instruct, L2 13B represents Llama-2-13B-Chat, L31 8B represents Llama-3.1-8B-Instruct.

Implementations. Our implementation is built upon the open-source repositories of EAGLE¹, HASS², and SpecForge³. As EAGLE-2 is a widely adopted method and HASS is built upon it, we mainly experiment with the EAGLE-2 framework. Besides, we experiment **SPEED** on the latest EAGLE-3 framework. To distinguish our method in the two frameworks, we name them “**SPEED(E2)**” and “**SPEED(E3)**”. Because EAGLE-3 introduces a much larger training set, we reproduce it using similar training steps to those in the EAGLE-2 framework for a fair comparison. All models apply tree-draft inference implemented by EAGLE-2,3. The detailed settings are introduced in Appendix A.

4.2 Main Results

We introduce the main results in this section. Table 1 presents the average acceptance lengths, and Table 2 presents the speed-up ratio.

Our methods achieve the highest overall acceptance length under different settings, demonstrating the effectiveness of position specialists in making accurate draft predictions. In the EAGLE-2 framework, **SPEED(E2)** achieves the best speed-up ratio under almost all settings. Although Gumiho outperforms **SPEED(E2)** at L3 8B with temperature=0, it is less stable and performs worse under other settings. In the EAGLE-3 framework, **SPEED(E3)** significantly outperforms the baseline on average acceptance length and the speed-up ratio. This is because the target model provides more powerful features in the EAGLE-3 framework, increasing the potential of draft models to predict longer, which is what **SPEED** better at. This further demonstrates the great potential of **SPEED**: the superiority of **SPEED** over other draft methods will be greater as the input feature becomes stronger.

4.3 SPEED in vLLM

To evaluate the performance of **SPEED** on real-world applications, we conduct an experiment on vLLM Kwon et al. (2023), a widely applied high-performance LLM generation framework. The experiments are conducted on one A100 80GB GPU and use

Batch Size	1	2	4	8
EAGLE-3	1.72x	1.79x	1.64x	1.67x
SPEED(E3)	2.15x	2.12x	2.07x	1.87x

Table 3: Speedup ratio with vLLM framework.

¹<https://github.com/SafeAILab/EAGLE>

²<https://github.com/HARmonizedSS/HASS>

³<https://github.com/sgl-project/SpecForge>

Llama-3.1-8B-Instruct as the target model. In this experiment, **SPEED** is trained for 30 epochs for convergence, and EAGLE-3 uses the official checkpoint⁴. The speedup ratios are exhibited in Table 3. **SPEED**(E3) consistently outperforms EAGLE-3 at all batch sizes, demonstrating the effectiveness of **SPEED** on industry-standard framework.

5 Analysis

5.1 Position-Wise Acceptance Rate

In Section 3.1, we introduce the metric **position-wise acceptance rate (pos-acc)** to reflect the acceptance rate of a specific position, which largely affects the overall acceptance length. Here we demonstrate that **SPEED** largely improves pos-acc by mitigating the feature deviation at each position and well balancing all positions.

Figure 4 shows the pos-acc with a draft depth of 8 on different models. EAGLE-2, with the least position generalization ability, has pos-acc lower than 65% from the 5th position on. HASS maintains adequate pos-acc at the first few positions, but consistently lower than **SPEED** at later ones, due to the limited generalization ability of the single draft model. EAGLE-3, with an advanced framework design, achieves higher pos-acc at later positions. However, its accuracy at the first position falls behind other methods because its features vary widely between the target and draft models. The single draft model is trained to balance all positions, so pos-acc_1 is compromised. In contrast, **SPEED** maintains substantially higher pos-acc on all positions, as each expert is specialized for fixed positions. This demonstrates the effectiveness of **SPEED** in mitigating position deviation and making accurate predictions.

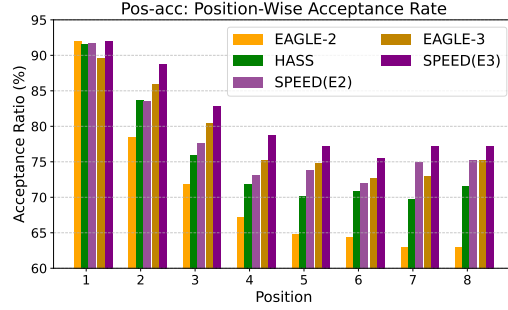


Figure 4: Position-wise acceptance rate of EAGLE, HASS, and variants of **SPEED**. Experiments are conducted on the MT-Bench dataset with a draft depth of 8. **SPEED** maintains a relatively higher pos-acc than corresponding baselines even at the 8th position.

5.2 Computational Efficiency Tradeoff on Draft Depth

Given concerns that more expert layers may incur additional time cost, we systematically analyze the time cost of speculative decoding and its relationship with draft depth.

A complete round of speculative generation involves two primary phases: **draft phase** and **verification phase**. We quantitatively analyze the time cost through three key metrics: (1) per-round computation time, (2) total round counts for test set generation, and (3) total time cost for test set generation. We demonstrate a comprehensive analysis in Figure 5 and present the following noteworthy observations.

Larger draft depth increases draft phase computation time. We present in Figure 5(a) the sum of per-round computation time over 5,000 rounds across varying draft depths, decomposed into draft phases and verification phases (bar chart), as well as the total rounds needed (line chart). Empirical results show that the increased total pre-round time is mainly attributed to the draft phase, and the length of draft sequences does not influence verification time.

EAGLE-3 framework reduces draft time but increases verification time. In Figure 5(a), comparing to models in the EAGLE-2 framework, models in the EAGLE-3 framework cost less time on draft, but more time on verification. The reduction in draft time results from the

⁴<https://huggingface.co/yuhui/EAGLE3-LLaMA3.1-Instruct-8B>

vocabulary-pruning setting, and the increase in verification time is because of the additional feature aggregation designs in EAGLE-3.

SPEED achieves the lowest overall computation time with reduced round counts. The overall computation time is the multiplication of the number of rounds and the pre-round time. In Figure 5(a), the bar chart demonstrates that **SPEED** has similar per-round calculation time to baseline methods, and the line chart shows that **SPEED** requires fewer rounds to complete the whole test set, which is the result of a larger acceptance length. The overall time cost in Figure 5(b) confirms that **SPEED** is faster than corresponding baselines. It is surprising that EAGLE-3 is the slowest when the draft depth is 4 and 5. This is because the first position accuracy of EAGLE-3 is negatively affected when training on large draft depth, as discussed in Section 5.1.

5.3 Ablation Study on Draft Model Prediction Depth

Figure 6 presents the throughput and average acceptance length under different draft depths. The average acceptance length τ increases with the draft depth consistently, but the improvement diminishes at larger depths. The diminishing improvement, along with the linearly increasing time cost of draft depth, creates an optimal point for throughput. In the experiment on the MT-Bench dataset, with Llama3/3.1 as the target model, we empirically demonstrate that the throughput peaks at draft depth = 5 and 7 for models of the EAGLE-2 and EAGLE-3 frameworks, respectively. This demonstrates that increasing **pos-acc** at later positions is beneficial to improving the overall throughput.

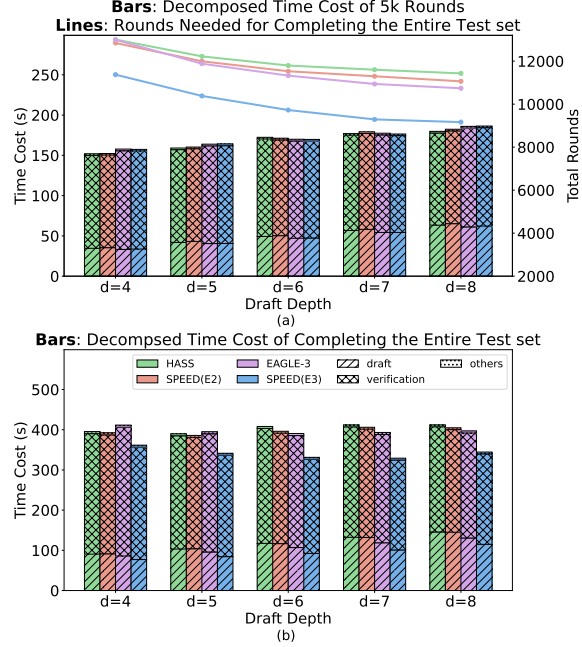


Figure 5: Computation time of different phases across varying draft depths. The bar plots present the decomposition of time spent on each speculation phase, where (a) measures the time spent on 5k rounds, and (b) measures the time to complete the entire test set. The line plot presents the number of rounds needed to complete a dataset.

6 Related Work

6.1 Linear Speculative Decoding

Early works (Xia et al., 2022) introduce the fundamental concept of speculative decoding for LLMs, followed by various improvements in linear speculative decoding. This includes adaptive calibration techniques (Gautam et al., 2025), dynamic candidate length adjustment (Huang et al., 2024b), and methods to optimize the latency-throughput trade-off (Sadhukhan et al., 2024). Recent advances focus on multi-token prediction (Gloeckle et al., 2024), efficient multi-sampling (Ni et al., 2024), and token recycling (Luo et al., 2024). Some also explore parallel decoding strategies with adaptive n-gram techniques (Ou et al., 2024; Wu et al., 2024; Liu et al., 2024; Wei et al., 2024).

6.2 Tree-based Speculative Decoding

Tree-based speculative decoding has advanced through several key works. GRIFFIN (Hu et al., 2025) and Sequoia (Chen et al., 2024) enhance token alignment methods, SpecInfer (Miao et al., 2024) improves sampling techniques, and Gumiho Li et al. (2025a) combines parallel and autoregressive drafting as a hybrid architecture. Other notable approaches include dynamic tree pruning (Zhong et al., 2024), early exit mechanisms (Elhoushi et al., 2024), hierarchical method (Sun et al., 2024). Other notable methods include Hydra (Ankner et al., 2024) and GLIDE (Du et al., 2024).

6.3 Efficient Inference

Recent works apply other methods to improve the inference speed. Judge Decoding (Bachmann et al., 2025) uses a small judge model to evaluate parallel reasoning paths, while SpecReason (Pan et al., 2025) and Speculative Thinking (Yang et al., 2025) leverage speculative computation for faster inference. Other efficient reasoning techniques include efficient chain-of-thought methods (Wang et al., 2025; Huang et al., 2025; Zheng et al., 2026), in-context learning methods (Huang et al., 2024a), non-myopic generation (Ma et al., 2024), system-level infra (Huang et al., 2024c), and hybrid generation of large and small models (Huang et al., 2026).

7 Conclusion

This paper proposes **SPEED**, a draft model consisting of multiple specialized position experts. This method investigates feature deviations across draft positions and appropriately handles them using position experts. Experiments show that **SPEED** maintains a high position-wise acceptance rate at later positions, achieving a larger acceptance length and faster generation speed than other methods.

Acknowledgments

We sincerely thank the reviewers and chairs for their valuable feedback.

This material is based upon work supported in part by the National Science Foundation under Award No. 2541822, the NVIDIA Academic Grant Program, WashU Ignite Interdisciplinary Grants and AI for Health Grants. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

References

- Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusingha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. Hydra: Sequentially-dependent draft heads for medusa decoding, 2024. URL <https://arxiv.org/abs/2402.05109>.
- Gregor Bachmann, Sotiris Anagnostidis, Albert Pumarola, Markos Georgopoulos, Artsiom Sanakoyeu, Yuming Du, Edgar Schönfeld, Ali Thabet, and Jonas Kohler. Judge decoding:

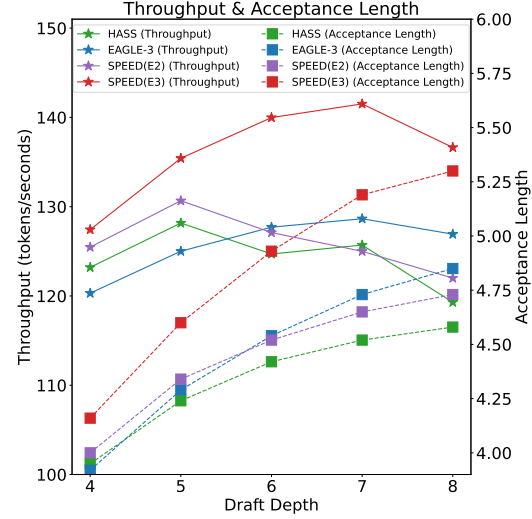


Figure 6: The throughput and average acceptance length on different draft depths. The experiments are conducted on the MT-Bench dataset. The acceptance length consistently increases with depth, while the throughput peaks at specific depths. This reflects the tradeoff among different draft depths.

- Faster speculative sampling requires going beyond model alignment. *arXiv preprint arXiv:2501.19309*, 2025.
- 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.
- Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, L. Sifre, and John M. Jumper. Accelerating large language model decoding with speculative sampling. *ArXiv*, abs/2302.01318, 2023. URL <https://api.semanticscholar.org/CorpusID:256503945>.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL <https://arxiv.org/abs/2107.03374>.
- Zhuoming Chen, Avner May, Ruslan Svirshchevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable, robust, and hardware-aware speculative decoding. *arXiv preprint arXiv:2402.12374*, 2024.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL <https://arxiv.org/abs/2110.14168>.
- Cunxiao Du, Jing Jiang, Xu Yuanchen, Jiawei Wu, Sicheng Yu, Yongqi Li, Shenggui Li, Kai Xu, Liqiang Nie, Zhaopeng Tu, and Yang You. Glide with a cape: A low-hassle method to accelerate speculative decoding, 2024. URL <https://arxiv.org/abs/2402.02082>.
- Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layerskip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*, 2024.
- Aayush Gautam, Susav Shrestha, and Narasimha Annapareddy. Token-driven gammatune: Adaptive calibration for enhanced speculative decoding. *arXiv preprint arXiv:2504.00030*, 2025.
- Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*, 2024.
- Shijing Hu, Jingyang Li, Xingyu Xie, Zhihui Lu, Kim-Chuan Toh, and Pan Zhou. Griffin: Effective token alignment for faster speculative decoding. *arXiv preprint arXiv:2502.11018*, 2025.
- Chengsong Huang, Langlin Huang, and Jiaxin Huang. Divide, reweight, and conquer: A logit arithmetic approach for in-context learning. *arXiv preprint arXiv:2410.10074*, 2024a.
- Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiaxin Huang. Efficient test-time scaling via self-calibration. *arXiv preprint arXiv:2503.00031*, 2025.
- Chengsong Huang, Tong Zheng, Langlin Huang, Jinyuan Li, Haolin Liu, and Jiaxin Huang. Relayllm: Efficient reasoning via collaborative decoding, 2026. URL <https://arxiv.org/abs/2601.05167>.

- Kaixuan Huang, Xudong Guo, and Mengdi Wang. Specdec++: Boosting speculative decoding via adaptive candidate lengths. *arXiv preprint arXiv:2405.19715*, 2024b.
- Xi Huang, Yinxu Tang, Junling Li, Ning Zhang, and Xuemin Shen. Toward effective retrieval augmented generative services in 6g networks. *IEEE Network*, 38(6):459–467, 2024c. doi: 10.1109/MNET.2024.3436670.
- H. T. Kung and John T. Robinson. On optimistic methods for concurrency control. In *TODS*, 1979. URL <https://api.semanticscholar.org/CorpusID:61600099>.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tac1a.00276. URL <https://aclanthology.org/Q19-1026/>.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.
- Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, 2022. URL <https://api.semanticscholar.org/CorpusID:254096365>.
- Jinze Li, Yixing Xu, Haiduo Huang, Xuanwu Yin, Dong Li, Edith C. H. Ngai, and Emad Barsoum. Gumiho: A hybrid architecture to prioritize early tokens in speculative decoding, 2025a. URL <https://arxiv.org/abs/2503.10135>.
- Shenggui Li, Yikai Zhu, Chao Wang, Fan Yin, Shuai Shi, Yubo Wang, Yi Zhang, Yingyi Huang, Haoshuai Zheng, and Yineng Zhang. Specforge: Train speculative decoding models effortlessly. <https://github.com/sgl-project/specforge>, 2025b.
- Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *International Conference on Machine Learning*, 2024a.
- Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-2: Faster inference of language models with dynamic draft trees. In *Empirical Methods in Natural Language Processing*, 2024b.
- Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference acceleration of large language models via training-time test, 2025c. URL <https://arxiv.org/abs/2503.01840>.
- Tianyu Liu, Yun Li, Qitan Lv, Kai Liu, Jianchen Zhu, and Winston Hu. Parallel speculative decoding with adaptive draft length. *arXiv preprint arXiv:2408.11850*, 2024.
- Xianzhen Luo, Yixuan Wang, Qingfu Zhu, Zhiming Zhang, Xuanyu Zhang, Qing Yang, Dongliang Xu, and Wanxiang Che. Turning trash into treasure: Accelerating inference of large language models with token recycling. *arXiv preprint arXiv:2408.08696*, 2024.
- Chang Ma, Haiteng Zhao, Junlei Zhang, Junxian He, and Lingpeng Kong. Non-myopic generation of language models for reasoning and planning. *arXiv preprint arXiv:2410.17195*, 2024.
- Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, et al. 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*, pp. 932–949, 2024.

- Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond, 2016. URL <https://arxiv.org/abs/1602.06023>.
- Yunsheng Ni, Chuanjian Liu, Yehui Tang, Kai Han, and Yunhe Wang. Ems-sd: Efficient multi-sample speculative decoding for accelerating large language models. *arXiv preprint arXiv:2405.07542*, 2024.
- Jie Ou, Yueming Chen, and Wenhong Tian. Lossless acceleration of large language model via adaptive n-gram parallel decoding. *arXiv preprint arXiv:2404.08698*, 2024.
- Rui Pan, Yinwei Dai, Zhihao Zhang, Gabriele Oliaro, Zhihao Jia, and Ravi Netravali. Specreason: Fast and accurate inference-time compute via speculative reasoning. *arXiv preprint arXiv:2504.07891*, 2025.
- Ranajoy Sadhukhan, Jian Chen, Zhuoming Chen, Vashisth Tiwari, Ruihang Lai, Jinyuan Shi, Ian En-Hsu Yen, Avner May, Tianqi Chen, and Beidi Chen. Magicdec: Breaking the latency-throughput tradeoff for long context generation with speculative decoding. *arXiv preprint arXiv:2408.11049*, 2024.
- 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.
- Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.
- Jikai Wang, Juntao Li, Lijun Wu, and Min Zhang. Efficient reasoning for llms through speculative chain-of-thought. *arXiv preprint arXiv:2504.19095*, 2025.
- Zhepei Wei, Wei-Lin Chen, Xinyu Zhu, and Yu Meng. Fast and accurate language model decoding via parallel token processing. In *Adaptive Foundation Models: Evolving AI for Personalized and Efficient Learning*, 2024. URL <https://openreview.net/forum?id=coZ0nwI78u>.
- Pengfei Wu, Jiahao Liu, Zhuocheng Gong, Qifan Wang, Jinpeng Li, Jingang Wang, Xunliang Cai, and Dongyan Zhao. Parallel decoding via hidden transfer for lossless large language model acceleration. *arXiv preprint arXiv:2404.12022*, 2024.
- Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. *arXiv preprint arXiv:2203.16487*, 2022.
- 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. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics ACL 2024*, pp. 7655–7671, Bangkok, Thailand and virtual meeting, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.456. URL <https://aclanthology.org/2024.findings-acl.456>.
- Wang Yang, Xiang Yue, Vipin Chaudhary, and Xiaotian Han. Speculative thinking: Enhancing small-model reasoning with large model guidance at inference time. *arXiv preprint arXiv:2504.12329*, 2025.
- Lefan Zhang, Xiaodan Wang, Yanhua Huang, and Ruiwen Xu. Learning harmonized representations for speculative sampling. *arXiv preprint arXiv:2408.15766*, 2024.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL <https://arxiv.org/abs/2306.05685>.

Tong Zheng, Chengsong Huang, Runpeng Dai, Yun He, Rui Liu, Xin Ni, Huiwen Bao, Kaishen Wang, Hongtu Zhu, Jiaxin Huang, et al. Parallel-probe: Towards efficient parallel thinking via 2d probing. *arXiv preprint arXiv:2602.03845*, 2026.

Shuzhang Zhong, Zebin Yang, Ruihao Gong, Runsheng Wang, Ru Huang, and Meng Li. Propd: Dynamic token tree pruning and generation for llm parallel decoding. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–8, 2024.

A Implementation Details

A.1 Implementation at Training Stage

We mainly follow the settings of the existing work. As the implementations of EAGLE-2 and EAGLE-3 vary widely, we introduce models for each framework separately.

Under the EAGLE-2 framework, we implement EAGLE-2, HASS, and **SPEED**(E2). These models are trained on the ShareGPT dataset, which contains 68K data entries (about 120K single dialogues after preprocessing by SpecForge (Li et al., 2025b)), aligning with EAGLE-1,2 and Medusa. They are trained for 40 epochs, as is implemented by HASS. **SPEED** apply the losses (including loss weights) of HASS.

Under the EAGLE-3 framework, we implement EAGLE-3 and **SPEED**(E3). Following EAGLE-3, the UltraChat-200K dataset, with 464K data entries, is added to the training set. Despite using a much larger training set, the official EAGLE-3 model is still trained for 40 epochs⁵. For a fair comparison with other models, we train EAGLE-3 and **SPEED**(E3) for total update steps similar to models of the EAGLE-2 framework, which is 10 epochs.

In **SPEED**, the second and third layers depend on output of the previous layer. Therefore, training all layers at first may bring in some noise, not necessarily reducing the performance but slowing down the convergence. To initialize **SPEED** training, we start training from half-trained EAGLE checkpoints. In the EAGLE-2 framework, for example, the EAGLE-2 model, trained for 20 epochs, is used to initialize all position experts of **SPEED**. **SPEED** is then trained for the remaining 20 epochs. In the EAGLE-3 framework, this number becomes 5, and all the processes are the same.

For Gumiho, we evaluate with its repository and provided checkpoints⁶.

A.2 Implementation at Inference Stage

The main experiments in this paper apply the tree-draft strategy and follow Li et al. (2024b) for hyperparameter settings in the EAGLE-2 framework, resulting in a draft depth of 6. In the EAGLE-3 framework, we increase the draft depth to 7 for optimal performance, as guided by the analysis in Section 5.2.

The deployment on vLLM applies the default setting, which is linear draft. This explains why speedup ratios in Table 3 are lower than main experiments.

B Generalization Capacity of an Expert

Ideally, each position should be handled by an exclusively specialized expert. However, additional experts come with a certain cost: extra time when hardware switches model parameters and extra memory. Empirically, we find that letting one expert handle more than one position harms prediction accuracy only marginally. This is because features of neighboring draft positions have similar hidden space, and handling them is within one expert’s ability. To investigate an expert’s generalization ability, we define the number of positions assigned to an expert as the expert’s generalization capacity.

The additional time cost has been discussed in Section 5.2, and this section will quantitatively analyze the memory cost and the generalization capability of one position expert.

B.1 Extra Memory Usage During Inference

Involving additional draft layers increases GPU memory usage, which scales linearly with the number of position experts. Fortunately, this additional cost is negligible compared to

⁵The number of training epochs/steps of EAGLE-3 is not disclosed in the original paper, but can be found in its official GitHub repository: https://github.com/SafeAILab/EAGLE/blob/main/eagle/train_eagle3/main.py

⁶<https://github.com/AMD-AGI/Gumiho>

the target model size since each specialist costs only one transformer layer (around 218M parameters per specialist for an 8B target model).

Empirically, Figure 7 visualizes the memory usage of the single draft model and **SPEED** with 2, 3, and 6 specialized position experts. This corresponds to generalization capacities of 3, 2, and 1, with draft depth = 6. Here, EAGLE and HASS cost the same GPU memory, and their generalization capacity can be treated as ∞ . The number of experts increases from left to right. In both target model settings, **SPEED** increases memory usage by only a few. The rightmost setting, despite using 6 expert layers, incurs an acceptable extra memory cost.

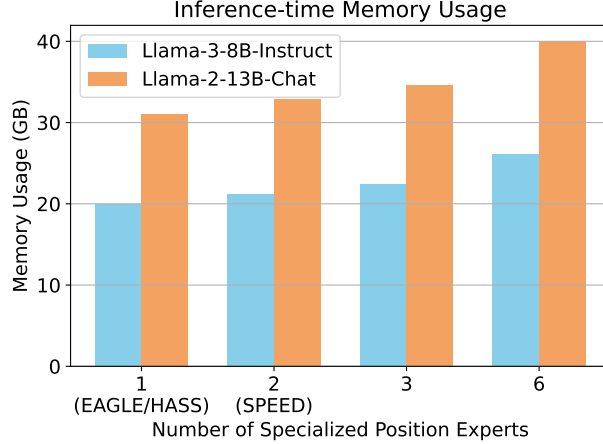


Figure 7: The Inference-time GPU memory usage of different speculative decoding methods. The memory usage is measured on the MT-bench test dataset. **SPEED** methods require slightly more GPU memory than EAGLE, the baseline method.

B.2 Generalization Capability of One Position Expert

Although generalization capacity can be set to any number, it is questionable if an expert is capable of handling that many positions. To analyze the generalization capability, we apply the model settings shown in Figure 7 and report the prediction accuracy of these models. As Figure 8 exhibits, **SPEED** with 1, 2, and 3 positions per expert (corresponding to 6, 3, and 2 experts in total) performs similarly in terms of position-wise acceptance rate. This demonstrates our intuition that neighboring features are similar and that one expert is sufficient to handle three positions.

Considering the additional memory and time cost, as well as the limited accuracy loss, we preassign each specialized position expert with three positions.

Table 4: Experiments of fixed layer allocation and MoE-based layer allocation. The fixed allocation method performs better in terms of average acceptance length and speedup ratio, demonstrating that layer specialization is necessary for **SPEED** and that mixing layer order degrades performance.

Model	Temperature=0													
	MT-Bench		Alpaca		GSM8K		Natural Questions		CNN/DM		HumanEval		Avg.	
	τ	speedup	τ	speedup	τ	speedup	τ	speedup	τ	speedup	τ	speedup	τ	speedup
SPEED(E3)	5.15	3.35x	5.50	3.45x	5.43	3.41x	4.13	2.71x	4.54	2.84x	5.95	3.88x	5.12	3.27x
SPEED(E3)-MoE	4.52	2.88x	4.72	2.95x	5.02	3.13x	3.87	2.45x	3.98	2.53x	5.43	3.51x	4.62	2.91x
Model	Temperature=1													
	τ	speedup	τ	speedup	τ	speedup	τ	speedup	τ	speedup	τ	speedup	τ	speedup
SPEED(E3)	4.66	2.90x	4.98	2.84x	5.24	3.11x	3.81	2.09x	4.14	2.49x	5.69	3.21x	4.75	2.77x
SPEED(E3)-MoE	3.53	1.86x	4.12	2.18x	4.22	2.14x	3.17	1.72x	3.55	1.84x	5.00	2.52x	3.93	2.05x

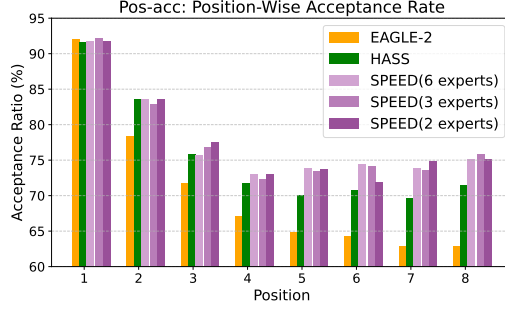


Figure 8: The pos-acc of **SPEED** with different generalization capacity. Statistics are measured on the MT-bench test dataset.

C Dynamic Layer Allocation

Throughout this paper, the position specialization is preassigned, i.e., the first expert handles the first three positions, and the second expert handles positions 4, 5, and 6, etc. In this section, we discuss the dynamic design of layer allocation.

A straightforward way to allocate multiple layers is the Mixture-of-Experts (MoE) approach. We design **SPEED**(E3)-MoE to investigate if standard MoE works for **SPEED** or not. Specifically, **SPEED**(E3)-MoE applies the structure of **SPEED**(E3), and a light-weight router to dynamically decide which layer to use at each position. The router functions in the same manner during training and inference. **SPEED**(E3)-MoE uses the same training data and training steps as **SPEED**(E3). The complete evaluation results on all six datasets are exhibited in Table 4.

The results clearly show that standard MoE does not work with **SPEED**, and here is an explanation for it. In standard MoE, all experts are counterparts of each other, with inputs and outputs in the same hidden state space. In **SPEED**, however, each subsequent layer refines the feature bias produced by the preceding one. Therefore, the input and output spaces are different for each position expert, and randomly mixing them introduces noise that confuses the model. This result, on the other hand, proves the necessity of layer specialization.