# POSS:Position Specialist Generates Better Draft for Speculative Decoding

**Anonymous Author(s)**
Affiliation
Address
email

## 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 these tokens in parallel. Recent studies leverage the hidden state of the target model to enhance draft model prediction accuracy. However, existing methods suffer from the degrading quality of draft token predictions at later positions, due to error accumulation in draft model generated features. In this paper, we propose Position Specialists (**POSS**), which consist of multiple position-specialized draft layers to generate tokens at assigned position(s). Position specialists greatly improve token acceptance rate at later positions per drafting round, as each specialist only needs to focus on handling a certain level of draft model feature deviation. Experiment results on Llama-3-8B-Instruct and Llama-2-13B-chat across six datasets demonstrate that **POSS** effectively improves over baselines on average acceptance length and speed-up ratio. Our codebase is available at `https://github.com/poss-speculative-decoding/Position-Specialist`.

## 1 Introduction

Speculative decoding [1, 2] is an effective approach to accelerate the generation 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 by the larger target model in parallel to determine accepted tokens from proposed draft tokens, thereby reducing overall decoding time. The effectiveness of speculative decoding largely depends on the average acceptance length $\tau$ (accepted token counts per round) from the prediction depth $L$ (predicted token counts generated by the draft model per round).

Recent efforts [3, 4, 5] utilize the target model hidden states as input to enhance draft model prediction accuracy. EAGLE [4, 5] employs a one-layer Transformer as the draft model and trains it to predict the next token with features from the target model. During inference, however, the target model features at unverified positions are inaccessible and are substituted by draft model features. This training–inference discrepancy harms prediction accuracy severely. HASS [6] partially addresses this by training the draft model to predict next tokens with features from previous draft steps. As draft position increases, the predicted features deviate more from the ground-truth features from target model. Therefore, HASS suffers from relying on a single draft model to predict tokens at multiple positions.

To address this challenge, we propose Position Specialists (**POSS**), a novel framework that consists of multiple position-specialized draft layers, called as position specialists. Each position specialist is trained to predict tokens at its assigned position(s), and only needs to handle an expected level of feature deviation at that position, thus enabling more accurate draft token predictions than a single draft model.

Submitted to NeurIPS 2025 Workshop on Efficient Reasoning. Do not distribute.

## 2 Background: Hidden State Assisted Speculative Decoding

Recent research efforts [3, 4, 5] discover the potential of the target model's hidden state. EAGLE and EAGLE-2 [4, 5] represent a significant breakthrough in speculative decoding through concatenating input embedding with feature vectors. It employs a one-layer Transformer as the draft model $\theta_D$ and reuses LM Head of the target model for token prediction. At generation step $t$, $\theta_D$ predicts the next first token $x_{t+1}$ based on context $x_{\leq t}$ and features $f_{<t}$:

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

where $x$ and $f$ are short for token embedding and model's feature, and the subscripts represent the timesteps. The superscripts of $f$ denote the source of it, e.g., $f^{(T)}$ and $f^{(D_i)}$ represent features from the target model and the $i^{th}$ draft step of the draft model.

When drafting tokens $x_{t+k}(k > 1)$, it differs from Equation (1) in the input. As features $f_{t+j}^{(T)}(j \geq 0)$ are accessible only after verification of the target model, features of the draft model, $f^{(D)}$, are used instead. Then, the prediction of tokens $x_{t+k}(k > 1)$ is given by Equation (2).

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

Although EAGLE-2 inferences with Equation (2), it is solely trained on Equation (1), known as "teacher forcing". This exhibits a fundamental training-inference discrepancy: $\theta_D$ predicts the subsequent tokens ($k > 1$) with its own generated features during inference, but it never observes its own prediction errors during training. HASS [6] mitigates the discrepancy through recursive feature alignment in training, where $\theta_D$ is trained to predict subsequent tokens with its own generated features from earlier timesteps. However, the systematic discrepancy can never be eliminated, and the prediction errors accumulate as draft steps increases. Worse still, existing researches rely heavily on the generalizability of a single draft layer for multi-position token generation, which impairs its ability to effectively predict long draft sequences.

## 3 Method: Position Specialists that Improve Position-Wise Acceptance Rate

To demonstrate the generalization limitation of EAGLE-2 and HASS, we first introduce the metric **position-wise acceptance rate** (**pos-acc**), which measures the probability that a token at position $i$ is accepted given that its preceding token at position $i-1$ is accepted. The strict definition of **pos-acc** is displayed in Appendix A.

We present the empirical **pos-acc** in Figure 1. EAGLE-2's **pos-acc** deteriorates rapidly beyond position $k = 1$, because the draft model is solely trained on predicting the next immediate token. HASS is able to maintain relatively higher **pos-acc** because its draft model is trained on multiple subsequent positions. However, since its single draft model needs to balance between multiple positions, the **pos-acc** drops by about 1 percent at the starting position $k = 1$.

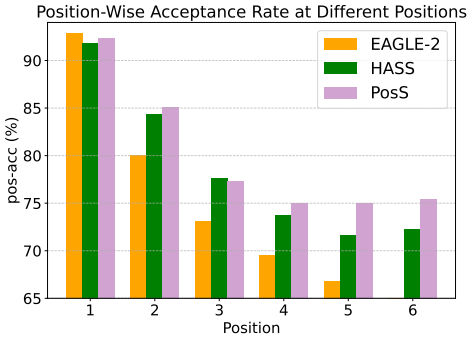


Figure 1: Position-wise acceptance rate (**pos-acc**) of the $i^{th}$ token on MT_Bench dataset by various speculative decoding methods. The **pos-acc** of EAGLE-2 and HASS decays fast as the draft sequence gets longer. Our proposed **PoSS** method keeps a stable and higher **pos-acc** even at the deepest position (draft model prediction depth $L = 6$).

To address this limitation, we introduce Position Specialists (**PoSS**) to preserve early-position acceptance rate while enhancing later position predictions. **PoSS** consists of multiple position-specialized draft layers, called position specialists. Each specialist is trained for certain position(s) and generates draft tokens at its assigned position(s). The number of positions that a specialist is assigned to can be predefined as $n$, and **PoSS**-$n$ means each specialist is responsible for $n$ positions.

Figure 2 illustrates the training of **PoSS** to explain why position specialists are better than a single draft model. Despite being trained, the draft model inevitably generates hidden states (features)

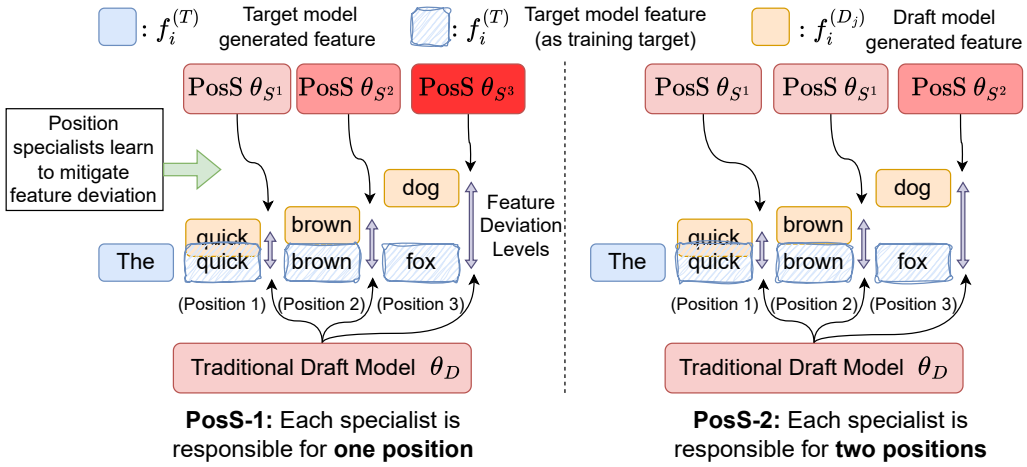


Figure 2: This figure shows a comparison of hidden state (feature) training between **PosS** and previous work. The draft model generated feature inevitably deviates from the target model, and the deviation level increases with error propagation in the draft model(s). In previous work, one draft model learns to mitigate varying levels of feature gaps. In **PosS** method, position specialists can focus on mitigating expected levels of feature deviation.

that deviate from the target model. The deviation $||f_i^{(D)} - f_i^{(T)}||$ increases with position $i$ through error propagation in the autoregressive draft process, as visualized by the purple arrow in Figure 2. While previous approaches like HASS train a single draft model to handle varying levels of input noise – a challenging task for a single-layer architecture, our **PosS** method employs position-specific specialists to handle the narrow and expected levels of feature deviation , enabling more accurate draft sequence prediction by decomposing it into position-specific subtasks.

Following previous work, **PosS** applies a token-level cross-entropy loss and a feature-level Smooth L1 loss [7], but the gradient at different draft positions updates corresponding specialist layers.

# 4 Experiment

## 4.1 Experiment Setup

**Metrics.** We evaluate the performance of our approach using two key metrics: speed-up ratio and average acceptance length. They are introduced in detail in Appendix B.1.

**Datasets.** We conduct comprehensive experiments on six datasets, following EAGLE-2 [4]. This includes MT-Bench [8] for multi-turn conversation, Alpaca [9] for instruction following, GSM8K [10] for mathematical reasoning, Natural Questions [11] for question answering, CNN/Daily Mail (shortened to CNN/DM) [12] for summarization, and HumanEval [13] for code generation.

**Target Models.** We evaluate our method on two model sizes: Llama-3-8B-Instruct and Llama-2-13B-chat. This allows us to evaluate how our approach performs across model sizes. Llama-3-8B-Instruct serves as our primary model for ablation studies and detailed analysis, while Llama-2-13B demonstrates the scalability of our method to larger models. The implementation details are introduced in Appendix B.2.

## 4.2 Results

We present the average acceptance lengths and speed-up ratio in Table 1. Our methods achieve the highest overall average acceptance length under different settings, demonstrating the effectiveness of position specialists in making accurate draft predictions. When L3 8B serves as the target model, **PosS** achieves consistently higher speed-up ratio over the baselines. When L2 13B is the target

model and generates stronger feature representations, **PosS** is less advantageous, but **PosS**-3 still achieves the highest speed-up ratio.

Due to extra computation overhead, the observed advantage of **PosS** on speed-up ratio is less significant than the average acceptance length. In Appendix D, we discuss the trade-off between the computation time cost and the gain in average acceptance length, and empirically demonstrate **PosS** is beneficial.

Table 1: Average acceptance length $\tau$ and speed-up ratio of all methods. L3 8B represents Llama-3-8B-Instruct, L2 13B represents Llama-2-13B-Chat.

| Average Acceptance Length | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 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 |
| | **PosS**-1 (ours) | **4.54** | 4.78 | 4.82 | **3.65** | **4.06** | 5.39 | 4.54 |
| | **PosS**-2 (ours) | **4.54** | **4.83** | **4.83** | 3.63 | **4.06** | 5.40 | **4.55** |
| | **PosS**-3 (ours) | 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 | **5.40** | **5.31** | 5.47 | 4.55 | 4.71 | 6.47 | 5.32 |
| | **PosS**-1 (ours) | **5.40** | 5.23 | 5.60 | 4.57 | 4.78 | 6.48 | 5.34 |
| | **PosS**-2 (ours) | 5.39 | 5.27 | 5.58 | 4.59 | 4.79 | **6.51** | 5.36 |
| | **PosS**-3 (ours) | **5.40** | 5.28 | **5.62** | **4.60** | **4.80** | **6.51** | **5.37** |
| Speed-up Ratio | | | | | | | | |
| 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 |
| | **PosS**-1 (ours) | 2.96x | 3.00x | 3.19x | **2.49x** | 2.50x | 3.52x | 2.94x |
| | **PosS**-2 (ours) | **2.99x** | **3.14x** | **3.25x** | 2.45x | **2.52x** | 3.52x | **2.98x** |
| | **PosS**-3 (ours) | 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 |
| | **PosS**-1 (ours) | 3.16x | 3.18x | 3.47x | 2.86x | 2.63x | 3.99x | 3.21x |
| | **PosS**-2 (ours) | 3.22x | 3.26x | 3.54x | 2.93x | 2.72x | 4.11x | 3.30x |
| | **PosS**-3 (ours) | **3.28x** | 3.32x | **3.59x** | **2.96x** | **2.74x** | 4.12x | **3.34x** |

### 4.3 Analysis on Position-Wise Acceptance Rate

In Figure 3, we show the pos-acc with a draft depth of 8. The empirical results demonstrate that **PosS largely improves pos-acc by mitigating the feature deviation at each position**. EAGLE-2, with the least position generalization ability, has pos-acc lower than 65% from the $5^{th}$ position on. HASS can only maintain adequate pos-acc at the first four positions, after which performance degrades significantly due to one single draft model. In contrast, all variants of our **PosS** method maintain substantially higher pos-acc until the last position.

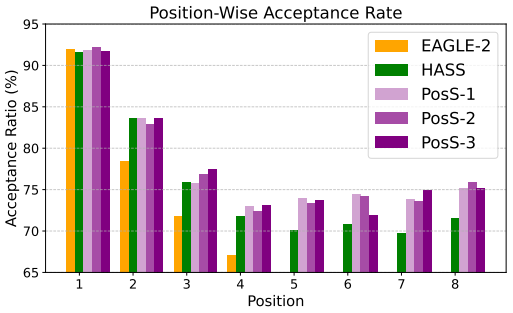


Figure 3: The position-wise acceptance rate of EAGLE-2, HASS, and variants of **PosS**. Experiments are conducted on MT-Bench dataset, with base model Llama-3-8B-Instruct and draft depth=8. Three variations of our method maintain a relatively higher pos-acc even at the $8^{th}$ position.

## 5 Conclusion

This paper proposes **PosS**, a draft model consisting of several position specialists. This method mitigates feature deviation between the draft and target models, and reduces the deviation accumulation across draft positions. Experiments show that **PosS** maintains a high position-wise acceptance rate at large positions, achieving a larger acceptance length and faster generation speed than other methods.

# References


[1] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, 2022.

[2] 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.

[3] 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.

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

[5] 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*, 2024.

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

[7] Ross B. Girshick. Fast R-CNN. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1440–1448. IEEE Computer Society, 2015.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond, 2016.

[13] 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.

[14] Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable, robust, and hardware-aware speculative decoding. *arXiv preprint arXiv:2402.12374*, 2024.

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

# A Position-Wise Acceptance Rate (pos-acc)

In this section, we provide a formal definition of Position-Wise Acceptance Rate. The **pos-acc** at position $i$ is defined as:

$$\textbf{pos-acc}_i = P(A_i \mid A_{i-1}) = \frac{P(A_{i-1} \cap A_i)}{P(A_{i-1})} = \frac{P(A_i)}{P(A_{i-1})}\,, \;\; i > 1 \tag{3}$$

where $A_i$ denotes the event that the token at position $i$ is accepted during the verifying process. Notice that the target model acceptance follows a strict sequential dependency: if $x_i$ is accepted, its preceding tokens $x_{[0:i-1]}$ must also 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 $\tau$ 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:

$$P(A_k) = P(A_1 \cap A_2 \cap \cdots \cap A_k) = \begin{cases} P(A_1) & \text{if } k = 1 \\ P(A_1) \prod_{i=2}^{k} \textbf{pos-acc}_i & \text{if } k > 1 \end{cases} \tag{4}$$

This chain rule decomposition reveals that the overall acceptance length depends on the multiplication of **pos-acc**, and is particularly sensitive to degradation in any single position. Notably, token prediction inherently becomes more challenging at later positions due to the accumulation of prediction errors and the increasing uncertainty in longer draft positions.

Besides, Equation (4) highlights the importance of accuracy at early positions. For example, a lower $P(A_1)$ affects not only the first position but also later positions through the chain rule. As a result, although the **pos-acc** of HASS drops by about 1 percent at position $k = 1$ in Figure 1, it critically impairs the overall acceptance length due to the multiplicative nature of the acceptance probability.

# B Experiment Details

## B.1 Metrics

- **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.
- **Average Acceptance Length** $\tau$: The average acceptance length represents the mean number of tokens accepted in each round of $L$ drafting positions (denoted as prediction length). 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.

## B.2 Implementations

Our implementation is built upon the open-source repositories of EAGLE-2 and HASS. We experiment with EAGLE-2, HASS, and our method with configurations of **PosS**-1, 2, 3, where **PosS**-3 adds the least extra layers and computation overhead.

The training configurations are mostly aligned with HASS, including the loss-related hyperparameters and the learning rate. When Llama-3-8B-Instruct serves as the target model, we notice that the open-sourced checkpoints of EAGLE-2 and HASS are inconsistent in structure[1]. To ensure the fairness of the comparison, we reproduce both methods with the model structure of HASS. While EAGLE-2 is trained for 20 epochs, HASS is trained for 40 epochs. We start training our method **PosS** from the

[1]In the configuration file of EAGLE-2 model, `https://huggingface.co/yuhuili/EAGLE-LLaMA3-Instruct-8B/blob/main/config.json`, the "*bias*" is false. However, this is true in the configuration file of HASS, `https://huggingface.co/HArmonizedSS/HASS-LLaMA3-Instruct-8B/blob/main/config.json`.

reproduced EAGLE-2 model for another 20 epochs to fairly compare with HASS. When Llama-2-13B-Chat serves as the target model, we use the open-sourced checkpoints of EAGLE-2 and HASS. Similarly, we train **PosS** from the EAGLE-2 model checkpoint for another 20 epochs.

During inference, tree-drafting [14] strategy is applied to generate multiple draft paths in one draft phase, where the *width*, *depth*, and *total tokens* are key controlling factors. We set the draft tree *width* to 10 and the number of draft *total tokens* to 60 for all experiments. We choose the draft tree *depth* that leads to the best performance, which we conduct an analysis experiment in Appendix C to search. Table 3 and 5 suggest that the 8B target model setting achieves the best performance at *depth*=6, and the 13B target model reaches the best performance at *depth*=7.

All experiments are conducted on NVIDIA A100 GPUs with 80GB of memory.

# C Analysis on Different Drafting Hyperparameters

Many factors influence the average acceptance length and speed-up ratio. Besides the prediction accuracy of draft models and computational overhead, the structure of draft trees also matters. We examine two key hyperparameters that affect the performance: depth and total tokens.

We take Llama-2-13B-chat as the base model, and conduct experiments with depths from 6 to 9, and total tokens selected from {60, 80}. We evaluate the models on all six datasets and take the average of them. Table 4 presents the average acceptance length, and Table 5 presents the speed-up ratio.

Interestingly, despite the consistent rise of average acceptance length as the number of total tokens increases from 60 to 80, the speed-up ratio shows a sharp drop. This indicates the target model takes significantly more time to verify. This phenomenon results from the inner structure of the A100 GPU device that we use for experiments, which is also observed by OPT-Tree [15].

Table 2: Average acceptance length under different hyperparameters. Experiments use Llama-3-8B-Instruct as the base model. We average the results on all six datasets. The largest average acceptance length within each column is bolded.

| Temperature | Depth | 6 | | 7 | | 8 | | 9 | |
|---|---|---|---|---|---|---|---|---|---|
| | Total Tokens | 60 | 80 | 60 | 80 | 60 | 80 | 60 | 80 |
| T=0 | HASS | 4.39 | 4.49 | 4.49 | 4.62 | 4.54 | 4.67 | 4.59 | 4.73 |
| | PosS-1 | 4.54 | 4.64 | 4.65 | 4.78 | **4.74** | 4.89 | **4.79** | 4.94 |
| | PosS-2 | **4.55** | **4.67** | **4.68** | **4.81** | **4.74** | **4.90** | **4.79** | **4.96** |
| | PosS-3 | 4.50 | 4.62 | 4.61 | 4.75 | 4.69 | 4.83 | 4.73 | 4.89 |
| T=1 | HASS | 4.16 | 4.24 | 4.22 | 4.34 | 4.26 | 4.39 | 4.30 | 4.41 |
| | PosS-1 | **4.28** | **4.37** | 4.35 | 4.48 | **4.44** | **4.58** | 4.47 | 4.58 |
| | PosS-2 | 4.27 | **4.37** | **4.37** | **4.53** | 4.43 | 4.57 | **4.48** | **4.64** |
| | PosS-3 | **4.28** | 4.35 | 4.30 | 4.49 | 4.40 | 4.53 | 4.43 | 4.53 |

Table 3: Speed-up ratio under different hyperparameters. Experiments use Llama-3-8B-Instruct as the base model. We average the results on all six datasets. The largest number within each row is bolded to show the upper bound of each method.

| Temperature | Depth | 6 | | 7 | | 8 | | 9 | |
|---|---|---|---|---|---|---|---|---|---|
| | Total Tokens | 60 | 80 | 60 | 80 | 60 | 80 | 60 | 80 |
| T=0 | HASS | **2.89x** | 2.83x | 2.84x | 2.78x | 2.76x | 2.71x | 2.67x | 2.65x |
| | PosS-1 | **2.94x** | 2.90x | 2.90x | 2.85x | 2.83x | 2.80x | 2.76x | 2.72x |
| | PosS-2 | **2.98x** | 2.92x | 2.93x | 2.87x | 2.84x | 2.81x | 2.77x | 2.74x |
| | PosS-3 | **2.95x** | 2.89x | 2.89x | 2.84x | 2.83x | 2.78x | 2.73x | 2.71x |
| T=1 | HASS | **2.63x** | 2.54x | 2.56x | 2.50x | 2.47x | 2.44x | 2.41x | 2.35x |
| | PosS-1 | **2.73x** | 2.65x | 2.66x | 2.59x | 2.60x | 2.55x | 2.53x | 2.48x |
| | PosS-2 | **2.66x** | 2.60x | 2.63x | 2.57x | 2.55x | 2.51x | 2.48x | 2.45x |
| | PosS-3 | **2.67x** | 2.59x | 2.60x | 2.56x | 2.55x | 2.47x | 2.48x | 2.41x |

Table 4: Average acceptance length under different hyperparameters. Experiments use Llama-2-13B-chat as the base model. We average the results on all six datasets. The largest average acceptance length within each column is bolded.

| Temperature | Depth | 6 | | 7 | | 8 | | 9 | |
|---|---|---|---|---|---|---|---|---|---|
| | Total Tokens | 60 | 80 | 60 | 80 | 60 | 80 | 60 | 80 |
| T=0 | HASS | 4.68 | 5.20 | 5.32 | 5.45 | 5.46 | 5.62 | 5.57 | 5.75 |
| | PosS-1 | 5.09 | 5.20 | 5.34 | 5.48 | 5.52 | 5.66 | 5.63 | 5.79 |
| | PosS-2 | **5.13** | **5.22** | 5.36 | 5.49 | 5.53 | 5.68 | 5.65 | 5.82 |
| | PosS-3 | **5.13** | 5.21 | **5.37** | **5.51** | **5.55** | **5.70** | **5.66** | **5.83** |
| T=1 | HASS | **5.00** | 5.06 | 5.14 | 5.29 | 5.24 | 5.45 | 5.35 | 5.52 |
| | PosS-1 | 4.99 | **5.11** | **5.24** | 5.31 | **5.34** | 5.49 | 5.43 | 5.52 |
| | PosS-2 | 4.96 | **5.11** | 5.16 | **5.32** | 5.30 | 5.49 | **5.44** | 5.61 |
| | PosS-3 | 4.99 | **5.11** | 5.21 | 5.31 | 5.33 | **5.50** | 5.43 | **5.62** |

Table 5: Speed-up ratio under different hyperparameters. Experiments use Llama-2-13B-chat as the base model. We average the results on all six datasets. The largest number within each row is bolded to show the upper bound of each method.

| Temperature | Depth | 6 | | 7 | | 8 | | 9 | |
|---|---|---|---|---|---|---|---|---|---|
| | Total Tokens | 60 | 80 | 60 | 80 | 60 | 80 | 60 | 80 |
| T=0 | HASS | 3.28x | 3.02x | **3.33x** | 3.08x | 3.31x | 3.09x | 3.28x | 3.09x |
| | PosS-1 | 3.16x | 2.93x | **3.21x** | 3.08x | **3.21x** | 3.09x | 3.20x | 3.09x |
| | PosS-2 | 3.26x | 3.00x | 3.30x | 3.06x | **3.31x** | 3.09x | 3.27x | 3.07x |
| | PosS-3 | 3.29x | 3.00x | 3.34x | 3.09x | **3.35x** | 3.11x | 3.30x | 3.10x |
| T=1 | HASS | 3.24x | 2.94x | **3.25x** | 3.00x | 3.20x | 3.01x | 3.18x | 2.98x |
| | PosS-1 | 3.13x | 2.93x | **3.17x** | 2.95x | 3.14x | 2.97x | 3.10x | 2.92x |
| | PosS-2 | 3.17x | 2.94x | **3.19x** | 2.98x | 3.18x | 2.99x | 3.17x | 2.98x |
| | PosS-3 | 3.24x | 2.97x | **3.26x** | 3.00x | **3.26x** | 3.02x | 3.18x | 3.01x |

# D Computation Overhead of Position Specialists

## D.1 The Computational Overhead

While **PoSS** generates drafts closer to the target model and achieves longer acceptance length, we point out two types of additional computation overhead that **PoSS** introduces.

First, the GPU memory usage increases linearly with the number of position specialists. Fortunately, this additional cost is negligible compared to the target model size since each specialist costs only one transformer layer (around 218M parameters per specialist for an 8B target model).

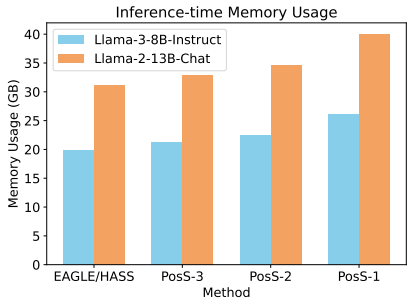


Figure 4: The Inference-time GPU memory usage of different speculative decoding methods. The memory usage is measured on the MT-bench test dataset. **PoSS** methods require slightly more GPU memory than EAGLE-2, the baseline method.

Second, the switching of position specialists brings a little extra latency. Although each **PoSS** specialist use the same structure with the single draft model of EAGLE-2 – theoretically implying equivalent computation time per draft phase – practical implementation of position specialists costs slightly more computation overhead for two reasons: (1) Non-shared KV cache across layers: Each position specialist computes key-value cache for draft tokens generated by its preceding specialist in addition to previously verified tokens. (2) Parameter switching overhead: Frequent parameter switching between specialists may introduce additional latency due to hardware-level parameter loading.

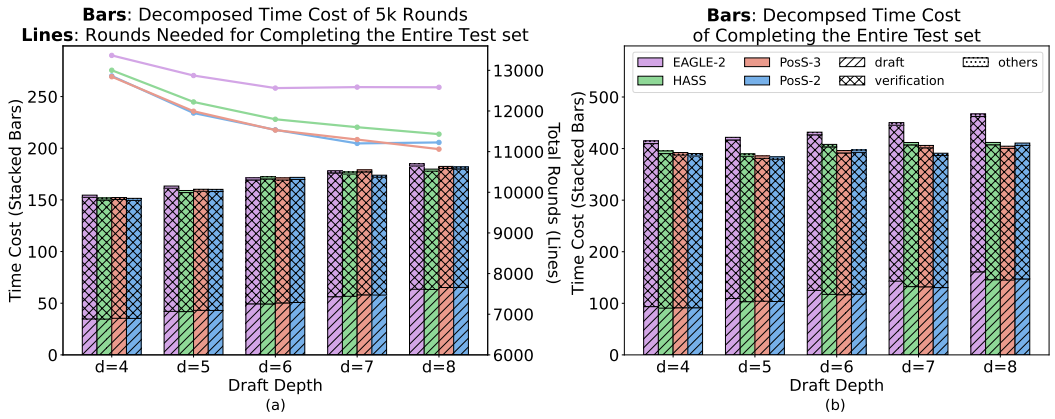


Figure 5: Computation time of different phases on MT-Bench dataset on different models across varying draft depths. The bar plots present the decomposition of time spent on each phase of speculative decoding, where subfigure (a) measures the time spent on 5k rounds and subfigure (b) measures the time to complete an entire test set. The line plot presents the number of rounds needed to complete a dataset. The lower the metrics are, the better the method is.

### D.2 Memory Cost for Extra Position Specialists

Figure 4 visualizes the memory usage of all methods mentioned in this paper. Here, EAGLE-2 and HASS cost the same GPU memory, and they are de facto **PosS**-$\infty$. From left to right, the draft layers in the methods are 1, 2, 3, and 6. In both target model settings, **PosS**-3 and **PosS**-2 increase few extra memory usage. **PosS**-1, despite using 6 times draft layers than EAGLE-2, costs acceptable extra memory usage.

### D.3 Computational Efficiency Tradeoffs for Extra Position Specialists

In this section, we study the positive effect of fewer draft-verification rounds and the negative effect of extra computation overhead per round brought by **PosS** through a comprehensive empirical analysis. We demonstrate that **PosS** only brings minimal overhead compared to the overall computation time, and this overhead is largely outweighed by the increased average acceptance length, which reduces the overall drafting rounds needed.

Each complete round of speculative generation involves two primary phases: the **draft phase** and the **verification phase**. In this experiment, 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.

**Position specialists bring minimal overhead to overall computation time.** We present in Figure 5(a) the sum of per-round computation time over 5,000 rounds across varying draft depths (bar chart), decomposed into draft phases and verification phases. Empirical results show that **PosS** has lower per-round time than EAGLE-2. Compared to HASS, **PosS** only brings a negligible fraction of time in the draft phase with positional specialists, and keeps similar verification phase costs.

**PosS achieves lowest overall computation time with reduced round counts.** The line chart in Figure 5(a) illustrates the total round counts needed for test set generation. **PosS**-2 and **PosS**-3 consistently require fewer rounds than baseline methods, benefitting from accurate draft token prediction from position specialists. The total time cost for decoding is primarily determined by both the **per-round time cost** and the **total round counts**. As shown in Figure 5(b), **PosS**-2 and **PosS**-3 achieve lower overall time costs compared to EAGLE-2 and HASS. This confirms that the efficiency gains from **reduced round costs substantially outweigh the marginal per-round overhead** brought by position specialists.