CAS-Spec: Cascade Adaptive Self-Speculative Decoding for On-the-Fly Lossless Inference Acceleration of LLMs

Zhiyuan Ning[1,2*] **Jiawei Shao**[1] **Ruge Xu**[2] **Xinfei Guo**[2]
Jun Zhang[3] **Chi Zhang**[1†] **Xuelong Li**[1†]
[1] TeleAI, [2] Shanghai Jiao Tong University, [3] Hong Kong University of Science and Technology,
{telegraphpolehead,schrodinger,xinfei.guo}@sjtu.edu.cn,
{shaojw2,zhangc120}@chinatelecom.cn, eejzhang@ust.hk, xuelong_li@ieee.org

Abstract

Speculative decoding has become a widely adopted as an effective technique for lossless inference acceleration when deploying large language models (LLMs). While on-the-fly self-speculative methods offer seamless integration and broad utility, they often fall short of the speed gains achieved by methods relying on specialized training. Cascading a hierarchy of draft models promises further acceleration and flexibility, but the high cost of training multiple models has limited its practical application. In this paper, we propose a novel Cascade Adaptive Self-Speculative Decoding (CAS-Spec) method which constructs speculative draft models by leveraging dynamically switchable inference acceleration (DSIA) strategies, including layer sparsity and activation quantization. Furthermore, traditional vertical and horizontal cascade algorithms are inefficient when applied to self-speculative decoding methods. We introduce a Dynamic Tree Cascade (DyTC) algorithm that adaptively routes the multi-level draft models and assigns the draft lengths, based on the heuristics of acceptance rates and latency prediction. Our CAS-Spec method achieves state-of-the-art acceleration compared to existing on-the-fly speculative decoding methods, with an average speedup from $1.1\times$ to $2.3\times$ over autoregressive decoding across various LLMs and datasets. DyTC improves the average speedup by 47% and 48% over cascade-based baseline and tree-based baseline algorithms, respectively. CAS-Spec can be easily integrated into most existing LLMs and holds promising potential for further acceleration as self-speculative decoding techniques continue to evolve.

1 Introduction

Large Language Models (LLMs), such as Llama [1], Mixtral [2], and Qwen [3] are rapidly evolving and increasingly adopted in various applications. However, their autoregressive generation process, where tokens are produced sequentially, coupled with their massive parameter sizes, leads to substantial inference latency and high computational cost. To mitigate this bottleneck, speculative decoding [4–6] has emerged as a highly effective and widely adopted approach, achieving significant speedup without compromising the quality of the generated text. It operates by utilizing a smaller, faster "draft" model to generate a sequence of multiple future tokens concurrently, which are then verified by the larger, more accurate "target" model in parallel, thereby reducing the number of expensive forward passes.

[*] Work done during an internship at TeleAI

[†] Corresponding Author.

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

Despite its promise, standard speculative decoding introduces a new burden: it requires training and maintenance of a separate draft model. This not only demands additional training data and computation resources, but also requires careful tuning to ensure compatibility and efficiency aligned with the target model. To address these challenges, self-speculative decoding approaches, self-speculative decoding methods [7–9] have been proposed. These techniques cleverly derive draft predictions from the target model itself, typically by skipping certain blocks in the modules or leveraging model compression techniques, thus eliminating the external training burden. While self-speculation simplifies the deployment pipeline, it often provides limited acceleration. In contrast, cascade speculative decoding [10] introduces a hierarchy of draft models, enabling a multi-stage approach with higher potential speedups and greater flexibility. Yet it requires multiple distinct draft models, making it largely impractical for widespread adoption in real-world systems.

In this paper, we aim to harness the potential of cascade speculation without the crippling overhead of training multiple draft models. We introduce **Cascade Adaptive Self-Speculative Decoding (CAS-Spec)**, a novel framework that brings the performance benefits of cascade speculative decoding without the overhead of training multiple draft models. CAS-Spec dynamically constructs a hierarchy of speculative draft stages using the target model itself by leveraging **Dynamically Switchable Inference Acceleration (DSIA)** strategies. These include techniques such as layer sparsity and activation quantization, enabling the creation of multiple draft models embedded within the target model's inference process. To coordinate this hierarchy at runtime, we further introduce **Dynamic Tree Cascade (DyTC)** algorithm to adaptively route the draft models, construct draft trees as well as controlling the draft lengths. It leverages heuristics based on token acceptance rates and latency prediction to maximize throughput.

Our contributions are summarized as follows:

- We propose CAS-Spec, a novel speculative decoding framework that creates multiple on-the-fly draft stages from a single target model. CAS-Spec achieves lossless inference acceleration without requiring additional draft model training.
- We introduce Dynamic Tree Cascade (DyTC), an adaptive routing algorithm that dynamically manages the draft models and their lengths based on heuristics of acceptance rates and latency predictions. DyTC provides 47% and 48% improvement in average speedup over the cascade-based and tree-based baseline, respectively.
- We demonstrate through extensive evaluations that CAS-Spec achieves state-of-the-art (SOTA) acceleration among on-the-fly speculative decoding methods, delivering speedups ranging from $1.1\times$ to $2.3\times$ over autoregressive decoding across various LLMs and datasets.

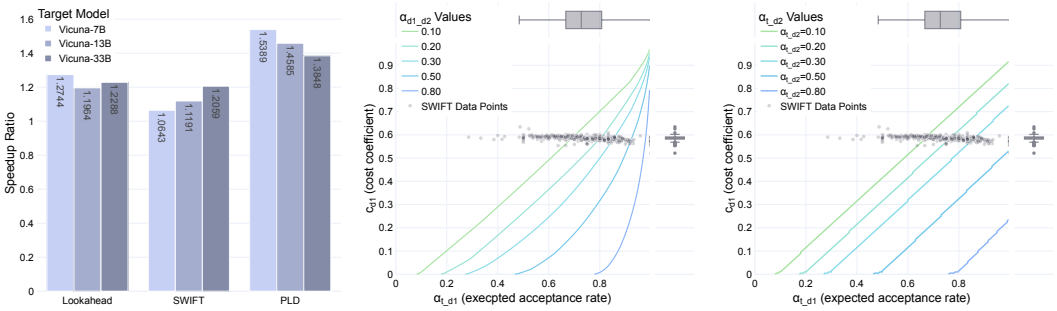
This work presents a practical and efficient approach to significantly accelerate LLM inference, paving the way for wider deployment of powerful language models in latency-sensitive and resource-constrained scenarios.

2 Preliminary

Autoregressive decoding, where each token depends on the previously generated ones, requires sequential execution, limiting the inference speeds of LLMs. Speculative decoding [4–6] offers a general framework to mitigate this issue by predicting multiple future tokens using a faster draft model $\mathcal{M}_d$ and verifying them with the target model $\mathcal{M}_t$ in parallel. To address practical challenges and unlock greater acceleration, various methods have been developed focusing on the nature and utilization of the draft model, leading to approaches like Self-Speculative Decoding and Cascade Speculative Decoding.

Self-Speculative Decoding. Self-speculative decoding (SSD) aims to eliminate the need for external draft models altogether, thereby reducing the overhead of training separate draft models. A common approach of SSD is to derive draft token predictions directly from the target model $\mathcal{M}_t$ itself during the inference process. Several strategies have been explored, including *layer sparsity* [7, 9, 11], *early-exiting* [8, 12, 13], *efficient attention* [14, 15], *Jacobi decoding* [16–18] and *activation quantization* [19]. Other methods such as EAGLE [20–22] and Medusa [23] reuse the target model's hidden states to generate draft tokens efficiently. By leveraging parts of the target model's own information or computation, SSD avoids the burden of separate draft model training and extra memory footprint for maintaining the key-value (KV) cache of the draft LLM.

Cascade Speculative Decoding. Vanilla draft models are commonly autoregressive LLMs, which means they can also be accelerated by speculative decoding. Further acceleration is promised by employing multiple draft models, typically ordered by decreasing sizes and latency, e.g., $\{\mathcal{M}_{d_1}, \mathcal{M}_{d_2}, ..., \mathcal{M}_{d_n}\}$. Applying this technique recursively leads to similar approaches known as *vertical cascade* [10], *hierarchical speculative decoding* [15] or *multi-level speculative decoding* [24]. In the direction of draft token generation, since rejecting a draft token means rejecting all the following draft tokens, the acceptance of early draft tokens is more important than the later ones. Cascade Speculative Drafting (CS-Drafting) [10] further proposes *horizontal cascade* which generates early draft tokens using a slightly slower draft model with a high acceptance rate, and subsequent draft tokens with progressively faster draft models. By leveraging *vertical cascade* and *horizontal cascade*, CS-Drafting achieves further acceleration over the vanilla speculative decoding.



(a) Spec-Bench Results (b) *vertical cascade* Bound (c) *horizontal cascade* Bound

Figure 1: (a) Comparison of on-the-fly SSD methods (Lookahead, SWIFT) and methods with statistical draft models (e.g. PLD) on Spec-Bench, tested on NVIDIA H100 GPU. (b) Theoretical effective bound of *vertical cascade* for a draft model $\mathcal{M}_{d_1}$ to be beneficial in the cascade speculative decoding compared with vanilla speculative decoding of $\mathcal{M}_{d_2}$ alone. The x-axis is the expected acceptance rate $\alpha(\mathcal{M}_t, \mathcal{M}_{d_1})$ and the y-axis is the cost coefficient $c(\mathcal{M}_t, \mathcal{M}_{d_1})$. The SWIFT data points are from the Spec-Bench results for `Vicuna-7B-v1.3` model. (The acceptance rates of PLD are between 0.1 and 0.5 in this setting.) (c) Theoretical effective bound of *horizontal cascade*, similar to (b). In this case, we consider $\alpha_{t,d2}$, which is commonly similar to $\alpha_{t,d2}$ in practice.

3 Motivation and Analysis

Although self-speculative decoding methods like EAGLE [20–22], BiTA [18] and LayerSkip [8] have shown promising speedups, these methods still require training for several days on a node of 8 common server GPUs like Nvidia A100. While the on-the-fly SSD methods like Lookahead [16] and SWIFT [9] are training-free, they are superior in terms of speed, even falling short of minimal retrieval-based drafting methods like prompt lookup (PLD) [25] on Spec-Bench [26], as shown in Fig. 1a. Since retrieval-based methods like PLD utilize the repeating tokens in the generation process to produce draft tokens, they are inherently lightweight and universally applicable, which weakens the competitiveness of the current training-free SSD methods. To break this trade-off between speed and ease of use, using a combination of multiple training-free SSD methods stands out as a potential solution. Chen et al. [10] present provable improvements over the standard SSD methods by cascading multiple draft models horizontally and vertically. However, not all LLMs have a series of smaller draft models available like the FLAN-T5 [27] family used in CS-Drafting. Requiring multiple draft models is a significant limitation for the practical application of cascade speculative decoding (CSD). We note that the training-free SSD methods can be used to construct a cascade of draft models for speculative decoding. Though it has been proven that a retrieval-based statistical model with negligible cost (e.g. PLD) can almost always gain further speedup against the standard speculative decoding [10], it is not guaranteed that such CSD can be faster than SD with the statistical model alone. It thus leads to the first research question to be addressed:

RQ 1: *Can existing training-free self-speculative decoding methods be used to construct an effective cascade of draft models above retrieval-based methods for on-the-fly speculative decoding?*

Firstly, we establish a theoretical effective bound for an *intermediate draft model* to bring a speedup in CSD. We adopt the same notations as in CS-Drafting for the sake of clarity:

Expected acceptance rate $\alpha(\mathcal{M}_t, \mathcal{M}_d)$: The probability that a draft token from a draft model $\mathcal{M}_d$ is verified as correct by the target model $\mathcal{M}_t$.

Cost coefficient $c(\mathcal{M}_t, \mathcal{M}_d)$: We define $c(\mathcal{M}_t, \mathcal{M}_d)$ as the ratio of inference times, indicating how much faster the draft model $\mathcal{M}_d$ is compared to the target model $\mathcal{M}_t$ for a single forward step.

Expected walltime improvement factor (EWIF) T: It represents the predicted gain in overall execution speed (wall time) assuming that the acceptance of each token is an i.i.d. Bernoulli trial.

As proved by Chen et al. [10], the EWIF of speculative decoding is: $T_{SD} = \frac{\phi'_{(\alpha,k)}(1)}{(ck+1)} = \frac{1-\alpha^{k+1}}{(1-\alpha)(ck+1)}$, where $\phi_{(\alpha,k)}(x) = 1 + (x-1)\frac{1-\alpha^{k+1}x^{k+1}}{(1-\alpha x)}$ is the probability generating function of the probability of generating i tokens p_i and $\alpha = \alpha(\mathcal{M}_t, \mathcal{M}_d)$. The EWIF for a *vertical cascade* with two draft models, $\mathcal{M}_{d_1}$ (slower, higher quality) and $\mathcal{M}_{d_2}$ (faster, lower quality), is [10]:

$$T_{VC(\mathcal{M}_{d_1},\mathcal{M}_{d_2})} = \frac{1-\alpha\phi^n(\alpha)}{(1-\alpha)(1+nc_{d_1}+nkc_{d_2})}, \tag{1}$$

where $\phi(x) = \phi_{(\alpha(\mathcal{M}_{d_1},\mathcal{M}_{d_2}),k)}(x)$, $\alpha = \alpha(\mathcal{M}_t, \mathcal{M}_d)$, and c_{d_1}, c_{d_2} are $c(\mathcal{M}_t, \mathcal{M}_{d_1}), c(\mathcal{M}_t, \mathcal{M}_{d_2})$ respectively. Adapting from Chen et al. [10], Thm. 4.5, the EWIF of *horizontal cascade* for two draft models is:

$$T_{HC(\mathcal{M}_{d_1},\mathcal{M}_{d_2})} = \frac{\frac{1-\alpha_{d_1}^{k_{d_1}+1}}{1-\alpha_{d_1}} + \alpha_{d_1}^{k_{d_1}}\frac{\alpha_{d_2}(1-\alpha_{d_2}^{k_{d_2}})}{1-\alpha_{d_2}}}{1+k_{d_1}c_{d_1}+k_{d_2}c_{d_2}}, \tag{2}$$

where $\alpha_{d_1} = \alpha(\mathcal{M}_t, \mathcal{M}_{d_1})$, $\alpha_{d_2} = \alpha(\mathcal{M}_t, \mathcal{M}_{d_2})$, $c_{d_1} = c(\mathcal{M}_t, \mathcal{M}_{d_1})$, $c_{d_2} = c(\mathcal{M}_t, \mathcal{M}_{d_2})$.

Then we get a theoretical bound by solving the inequalities $T_{VC(\mathcal{M}_{d_1},\mathcal{M}_{d_2})} \geq T_{SD(\mathcal{M}_{d_2})}$ and $T_{HC(\mathcal{M}_{d_1},\mathcal{M}_{d_2})} \geq T_{SD(\mathcal{M}_{d_2})}$, where $T_{SD(\mathcal{M}_{d_2})} = \frac{1-\alpha_{d_2}^{k_0+1}}{(1-\alpha_{d_2})(c_{d_2}k_0+1)}$. The solution of these inequalities are presented in the Appendix B.

However, the solutions of the inequalities are highly dependent on the hyperparameters of the speculative decoding scheduling, such as k_{d_1}, k_{d_2}, and n.

Thus, we should compare the EWIF of these methods with optimal hyperparameters to find a tighter bound, i.e.

$$\max_{n,k} T_{VC(\mathcal{M}_{d_1},\mathcal{M}_{d_2})} \geq \max_{k_0} T_{SD(\mathcal{M}_{d_2})}, \quad \max_{k_{d_1},k_{d_2}} T_{HC(\mathcal{M}_{d_1},\mathcal{M}_{d_2})} \geq \max_{k_0} T_{SD(\mathcal{M}_{d_2})} \tag{3}$$

This inequality (Eq. 3) does not readily yield a closed-form expression for c_{d_1}, as the maximization over integer hyperparameters $(k_0, n, k, k_{d1}, k_{d2})$ typically requires numerical evaluation of given model parameters like the expected acceptance rates and cost coefficients. Therefore we conduct a numerical simulation to find the theoretical effective bound for $\mathcal{M}_{d_1}$ to be benificial in the CSD. When $\mathcal{M}_{d_2}$ is a retrieval-based statistical model with negligible cost, we assume $c_{d_2} = 0.01$ and $\alpha(\mathcal{M}_t, \mathcal{M}_{d_2}) = \alpha(\mathcal{M}_{d_1}, \mathcal{M}_{d_2})$. The simulation results of the borderline of c_{d_1} and $\alpha(\mathcal{M}_t, \mathcal{M}_{d_1})$ are shown in Fig. 1b and Fig. 1c.

With the theoretical effective bound established, we can analyze the performance of existing training-free SSD methods in the context of CSD. For instance, SWIFT [9] employs a layer sparsity strategy, the distribution of its acceptance rates and cost coefficients on Spec-Bench are illustrated in Fig. 1b and Fig. 1c. As observed, most of the data points of SWIFT lie above the theoretical effective bound, indicating that naive HC or VC cascade with SWIFT as the intermediate draft model does not guarantee a speedup over using PLD alone for speculative decoding.

While it's feasible to cascade multiple training-free SSD methods for CSD, more effective cascade algorithms are necessary to fully leverage their potential, with tree-based structures being a promising direction. Since the EWIF of CSD depends on the scheduling algorithm of different draft models, it is possible to achieve a better speedup by adaptively routing the draft models and assigning the draft lengths. This leads to our second research question:

RQ 2: *Can we achieve further speedup by adaptively routing the draft models and assigning the draft lengths, with regards to the characteristics of different DSIA strategies?*

This question is explored in Section 4.2, where we introduce the Dynamic Tree Cascade (DyTC) algorithm for online scheduling of CAS-Spec.

4 Cascade Adaptive Self-Speculative Decoding

To address the outlined challenges and leverage the potential of training-free SSD methods, we propose Cascade Adaptive Self-Speculative Decoding (CAS-Spec). CAS-Spec constructs a hierarchy of draft models using various inference acceleration strategies applied to the target model $\mathcal{M}_t$. It then employs a dynamic mechanism to route through this hierarchy and determine draft lengths.

4.1 Construct Draft Models with Dynamically Switchable Inference Acceleration Strategies

Definition 4.1. A **Dynamically Switchable Inference Acceleration** (DSIA) strategy is a technique that modifies the inference process of a model to accelerate the token generation, which can be dynamically switched on or off during inference. These strategies can often be parameterized (e.g., by the degree of sparsity, quantization bit-width).

Each DSIA strategy, potentially with different parameter settings, can be viewed as creating a distinct "virtual" draft model $\mathcal{M}_{d_i}$ derived from $\mathcal{M}_t$. Examples of DSIA strategies include:

- *Layer Sparsity*: Skipping a subset of transformer layers or a subset of attention and FFN blocks in $\mathcal{M}_t$ to generate draft tokens [7, 9, 11].
- *Early-Exiting*: Using predictions from intermediate layers of $\mathcal{M}_t$ as draft tokens [8, 12, 13].
- *Activation Sparsity*: Keeping only a subset of neurons (activations) in each layer to reduce computation and memory movement of weights. It commonly requires batch size to be 1 or small. [3]
- *Activation Quantization*: Using lower precision (e.g., INT4) for activations and (partial) KV cache of the model during draft generation, as explored by QSpec. It requires a weight-only quantized target model for considerable speedup.
- *Efficient Attention*: Using an efficient attention mechanism like StreamingLLM [30] for draft generation. This is explored in the context of SSD by TriForce and MagicDec. Such methods are commonly more performant in long context generation.

To construct a hierarchy of draft models, there are three approaches in general:

- **Mixing-DSIA Cascade**: Using orthogonal DSIA strategies to create a series of draft models. For example, $\mathcal{M}_{d_1}$ could be a layer-sparse model, while $\mathcal{M}_{d_2}$ could have both layer sparsity and activation sparsity.
- **Replacing-DSIA Cascade**: Using conflicted DSIA strategies to create a series of draft models. For example, $\mathcal{M}_{d_1}$ could be a model with FP8 quantized SageAttention2 [31], and $\mathcal{M}_{d_2}$ could be a model with StreamingLLM attention.
- **Scaling-DSIA Cascade**: Using the same DSIA strategy with different parameter settings (e.g., different degrees of sparsity) to create a series of draft models $\{\mathcal{M}_{d_1}, \mathcal{M}_{d_2}, \ldots, \mathcal{M}_{d_n}\}$.

In the spectrum of the trade-off between speed and accuracy, each intermediate draft model $\mathcal{M}_{d_i}$ typically satisfies $\alpha(\mathcal{M}_t, \mathcal{M}_{d_i}) \geq \alpha(\mathcal{M}_t, \mathcal{M}_{d_{i+1}})$ and $c(\mathcal{M}_t, \mathcal{M}_{d_i}) \leq c(\mathcal{M}_t, \mathcal{M}_{d_{i+1}})$. The last model in the hierarchy is expected to be the fastest and least accurate, such as an extremely fast, often non-neural or retrieval-based method.

Definition 4.2. In a hierarchy fo draft models $\{\mathcal{M}_{d_1}, \mathcal{M}_{d_2}, \ldots, \mathcal{M}_{d_n}\}$, the *bottom draft model* $\mathcal{M}_{d_n}$ is a model that serves as the final stage in a cascade of draft models, which means it cannot be further accelerated by speculative decoding.

Prompt Lookup Decoding (PLD) [25] is a prime example. Statistical n-gram models or small, fixed draft heads like those in EAGLE [20–22] and Medusa [23] could also serve this role. However, since EAGLE and Medusa require the hidden states of the target model, their performance will be greatly affected when using the hidden states from DSIA draft models. For simplicity and considering its proven efficacy in CS-Drafting [10], we often consider PLD as a default $\mathcal{M}_{d_n}$.

The $\mathcal{M}_{d_i}$ are inherently training-free if the DSIA strategies are training-free. By choosing a training-free bottom draft model and DSIA strategies, CAS-Spec can be implemented without training multiple draft models and thus more easily integrated into a wide range of LLMs. Among the listed DSIA strategies, layer sparsity does not require special condition to gain speedup. We choose it for easy

[3] There are no existing works on SSD with activation sparsity yet, but it is a promising direction with recent advances in this technique [28, 29]

comparison with other speculative decoding methods. CAS-Spec with other DSIA strategies like activation sparsity and quantization is discussed in the Appendix C.

4.2 Dynamic Tree Cascade (DyTC)

Tree attention allows for more flexible and efficient speculative decoding by enabling the parallel verification of different branches of draft tokens [23, 32]. Similarly, early tokens in a draft token tree should also be prioritized for more accepted tokens.

Proposition 4.3. ***Tree Cascade*** *(TC) assigns the different draft models in the draft token tree to maximize the expected acceptance rate of the early draft tokens.*

Given a set of available DSIA strategies and a bottom draft model, there are many possible configurations for the cascade of draft models, forming a set of candidate draft models.

In CAS-Spec systems, the challenge is to determine *where* to start generating tokens (in the draft token tree), *which* draft models to use, and *when* to switch between them. Since the dimensions of the search space are large, it is impractical to use global optimization methods, which are commonly used in vanilla speculative decoding to find the optimal draft length. Instead, we propose to use a heuristic-based approach to dynamically adapt the hyperparameters of the scheduling algorithm during inference. This dynamic adaptation is guided by heuristics based on continuously updated estimates of acceptance rates and latency predictions. Inspired by the idea of dynamic draft tree expansion [33], we propose to leverage online optimization to adapt the expansion of *Tree Cascade*:

Proposition 4.4. ***Dynamic Tree Cascade*** *(DyTC) is a dynamic scheduling algorithm that adaptively selects the draft models and their configurations in a tree structure based on the online acceptance rates and latency predictions.*

Acceptance Rate for Draft Configurations. For each potential draft model configuration $\mathcal{M}_{d_i}$ (including DSIA variants of $\mathcal{M}_t$ and vertical cascade combinations), DyTC maintains an online estimate $\hat{\alpha}(\mathcal{M}_t, \mathcal{M}_{d_i})$ of its acceptance rate. This estimate is continuously updated using an Exponential Moving Average (EMA) mechanism:

$$\hat{\alpha}_{new} = \lambda \cdot \hat{\alpha}_{prev} + (1 - \lambda) \cdot \hat{\alpha}_{recent} \tag{4}$$

where $\hat{\alpha}_{prev}$ is the estimate from the previous step, $\hat{\alpha}_{recent}$ is the acceptance rate computed from a local history window of the most recent H generation steps (we use $H = 20$ and $\lambda = 0.7$ in our experiments), and λ controls the balance between stability and responsiveness to changing generation contexts.

Critically, for computing $\hat{\alpha}_{recent}$, we focus on the acceptance of the *first* draft token generated by each configuration, rather than the overall ratio of accepted to drafted tokens. Specifically, if the local history contains outcomes $\{o_1, o_2, \ldots, o_H\}$ where $o_i \in \{0, 1\}$ indicates whether the first drafted token was accepted, then: $\hat{\alpha}_{recent} = \frac{1}{H} \sum_{i=1}^{H} o_i$.

The EMA-based update ensures that $\hat{\alpha}$ adapts to the dynamic nature of text generation, where acceptance probability can vary significantly across different tasks (e.g., translation vs. summarization) and even within a single sequence. For cold starts initialization, a brief calibration phase can be performed at the beginning of generation to gather initial acceptance statistics for each configuration, detailed in Appendix D.

Token-Level Information for Drafted Tokens. While the configuration selection in `FindBestConfigurationForStep` (Algorithm 2) relies on the aggregate acceptance rate estimates described above, the calculation of *accumulated acceptance rate* $\prod_{j=1}^{l_s} \hat{\alpha_j}$ for already-drafted but not-yet-verified tokens *does* incorporate token-level information. Specifically, we consider:

- For neural draft models: the normalized probability (logit) of the drafted token, which is positively correlated with acceptance likelihood [33].
- For non-neural drafts (e.g., PLD): longer length of the n-gram match indicating higher confidence.

This token-level refinement allows DyTC to more accurately estimate the quality of different candidate branches in the draft tree when selecting the next leaf node to expand. However, for *future* tokens (those not yet generated), we cannot access such token-level information without actually running the draft model, which would be prohibitively expensive for all candidate configurations. Therefore, the configuration selection relies on the historical acceptance rate estimates.

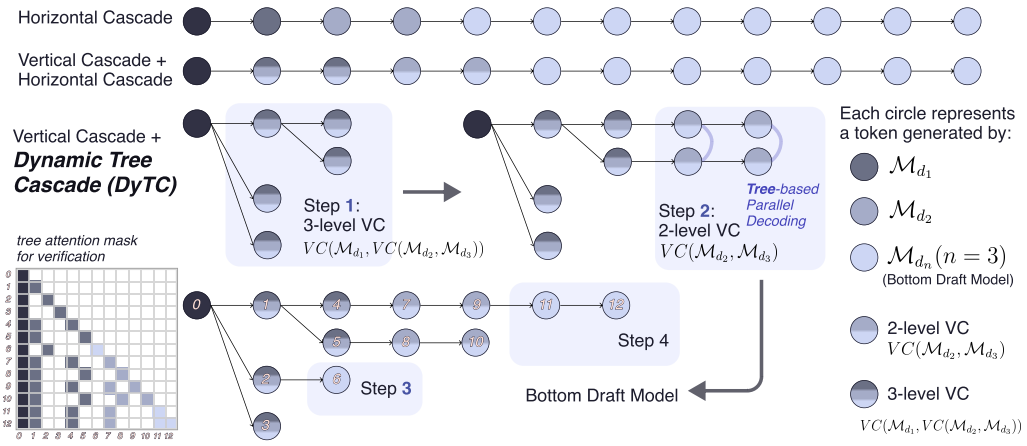


Figure 2: Illustration a example of the Dynamic Tree Cascade (DyTC) algorithm when $n = 3$.

Hardware-Aware Latency Prediction ($\hat{c}$): The cost coefficient $c(\mathcal{M}_t, \mathcal{M}_{d_i})$ depends on the specific DSIA strategy and the hardware platform. DyTC utilizes a latency model to predict these costs. We predicted the roofline latency of the hardware platform with Bayesian linear regression.

DyTC Algorithm. Firstly, we start by generating tokens at the leaf node where the *accumulated acceptance rate* $\prod_{j=1}^{l} \hat{\alpha_j}$ is the highest. Here l is the path length from the root node to the leaf node and $\hat{\alpha_j}$ is the estimated acceptance rate of the j-th node in the path. At each decoding step, DyTC evaluates different cascade configurations. A "configuration" involves a selection of DSIA models, their draft lengths, and their arrangement. For simplicity, we consider a round of *vertical cascade* as a holistic step. For example, at the s-th step of generation, DyTC considers:

- Generate with a single model $\mathcal{M}_{d_i}$, with a draft length of k_s (commonly small for finer control).
- A *vertical cascade*: $VC(\mathcal{M}_{d_i}, (\ldots, \mathcal{M}_{d_n}))$, with an *expected* draft length of k_s (the draft length cannot be strictly controlled since it depends on the acceptance of the low-level draft tokens).
- End the generation of the draft tokens (if the *accumulated acceptance rate* $\prod_{j=1}^{l_s} \hat{\alpha_j}$ is too low).

The decision-making process aims to maximize the overall EWIF, choosing proper configurations $\mathcal{M}_{d_s}$ and k_s. Firstly, we start with a *Greedy Search* that optimize the predicted local speedup $\hat{t_s} = \frac{\hat{\alpha}(1-\hat{\alpha}^{k_s})}{(1-\hat{\alpha})\hat{c}k_s} \prod_{j=1}^{l_s} \hat{\alpha_j}$ of the current step. However, the speedup of CSD may not obey the Greedy Choice Property, that is, choosing the locally optimal solution at each step does not guarantee a globally optimal EWIF. For instance, if there are two draft models: $\mathcal{M}_{d_1}$ with $\hat{\alpha_{d_1}} = 0.9$ and $\hat{c_{d_1}} = 0.4$, and $\mathcal{M}_{d_2}$ with $\hat{\alpha_{d_2}} = 0.8$ and $\hat{c_{d_2}} = 0.3$, the local speedup when $k_s = 1$ is $\hat{t_s}(\mathcal{M}_{d_1}) = \frac{0.9}{0.4} \approx 2.25$ and $\hat{t_s}(\mathcal{M}_{d_2}) = \frac{0.8}{0.3} \approx 2.67$ at the first step. The *Greedy Search* algorithm would select $\mathcal{M}_{d_2}$ at every step, with a suboptimal overall EWIF of 1.554. On the other hand, if we use the *horizontal cascade* of $\mathcal{M}_{d_1}$ and $\mathcal{M}_{d_2}$, we can achieve an overall EWIF of 1.615.

Dynamic programming could be used to find the optimal global solution, but the computational overhead is prohibitive for online scheduling since the search space grows exponentially with the number of steps. Inspired by the concept of "admissible heuristic" from the A* algorithm[34], we propose to adjust the local optimization target considering not only the estimated speedup brought by the current step, but also the estimated *least future speedup* to address this issue. We define the *least future speedup* as the EWIF of using the Bottom Draft model $\mathcal{M}_{d_n}$ for the following draft step. So the subproblem of the s-th step is to maximize the following objective function:

$$\mathcal{T}_s(\mathcal{M}_{d_s}, k_s) \prod_{j=1}^{l_s} \hat{\alpha_j}, \quad \text{where } \mathcal{T}_s(\mathcal{M}_{d_s}, k_s) = \frac{\frac{\hat{\alpha}(1-\hat{\alpha}^{k_s})}{1-\hat{\alpha}} + \hat{\alpha}^{k_s}\hat{\alpha_{d_n}}}{\hat{c}k_s + \hat{c_{d_n}}} \tag{5}$$

Since $\prod_{j=1}^{l_s} \hat{\alpha_j}$ only depends on the chosen leaf node, we first find the best leaf node with the highest *accumulated acceptance rate*. We stop the generation of the draft tokens if $\frac{\hat{\alpha_{d_n}}}{\hat{c_{d_n}}} \prod_{j=1}^{l_s} \hat{\alpha_j} < t_{min}$,

Algorithm 1: Dynamic Tree Cascade (DyTC) Draft Generation

Input : Initial prefix x_0, Maximum tree size M_{tree_max}
Set of candidate draft model configurations $\mathcal{S}$, Bottom draft model $\mathcal{M}_{d_n}$ with $\hat{\alpha}_{d_n}, \hat{c}_{d_n}$
Minimum overall speedup threshold t_{min}, Maximum draft length per expansion step k_{max}
Top-K token selection K, Top-P tree probability threshold P_{tree}
Output : Generated draft token tree T_r

Initialize T_r with N_{root} representing the last bonus token x_0
Dictionary tracking accumulated acceptance rate $P_{acc}[N_{root}] \leftarrow 1.0$
Dictionary tracking active nodes $D_{active}[N_{root}] \leftarrow True$ `// Mark` N_{root} `as active leaf`
while $T_r.size() < M_{tree\_max}$ **do**
  $N_{leaf} \leftarrow arg\max_{N \in T_r}\{P_{acc}[N] \mid D_{active}[N] = True\}$
  if N_{leaf} *is* null **then**
    break `// No more active leaves to expand`
  $(S^*, k^*) \leftarrow$ `FindBestConfigurationForStep`$(\mathcal{S}, \hat{\alpha}_{d_n}, \hat{c}_{d_n}, k_{max})$
  if S^* *is* null **then**
    $D_{active}[N_{leaf}] \leftarrow False$
    break `// No beneficial configuration, stop expansion`
  $x \leftarrow$ `GetSequenceToNode`(N_{leaf})
  $x_{siblings} \leftarrow$ `GetSiblingTokens`(N_{leaf})
  if S^* *is not* $\mathcal{M}_{d_n}$ ***And*** $x_{siblings}$ *is not empty* **then**
    $x \leftarrow$ concat$(x, x_{siblings})$ `// Tree-based sequence parallelism`
  $\mathbf{y} \in \mathbb{N}^{k^* \times K} \leftarrow$ `GenerateDraftTokens`(S^*, x, k^*)
  $\hat{\alpha}_{S^*} \leftarrow$ current estimate $\hat{\alpha}(S^*)$
  for $i \leftarrow 1$ **to** k^* **do**
    for j *in* arg *top*$_P$ $\mathbf{y}[i, :]$ **do**
      $N_{parent} \leftarrow$ `GetSiblingNode`(N_{leaf}, j) `// returns` N_{leaf} `if` $j = 0$
      $N_{new} \leftarrow T_r$.add_child$(N_{parent}, \mathbf{y}[i, j],$ info from $S^*)$
      if $j == 0$ **then**
        $\mathrm{N}_{first} \leftarrow N_{new}$
      $P_{acc}(N_{new}) \leftarrow P_{acc}(N_{parent}) \times \hat{\alpha}_{S^*}$
      $D_{active}[N_{new}] \leftarrow True$
    $N_{parent} \leftarrow N_{first}$
    if $T_r.size() \geq M_{tree\_max}$ **then**
      return T_r `// Tree size limit reached`
return T_r

where t_{min} is a threshold for the minimum local speedup, or the total tree size exceeds the maximum size m. Then we get the best configuration $\mathcal{M}_{d_s}$ and k_s by solving the optimization problem in Eq. 5:

$$\mathcal{M}_{d_s}, k_s = \underset{\mathcal{M}_{d_s}, k_s}{\arg\max} \mathcal{T}_s(\mathcal{M}_{d_s}, k_s), \quad \text{s.t. } k_s \in [1, k_{\max}] \tag{6}$$

Tree-based Parallel Draft Generation. Draft models constructed with DSIA strategies are themselves an LLM variant, with memory-bounded inference process. Tree attention allows for not only parallel verification of multiple candidate draft paths, but also parallel generation of draft paths. Following the idea of tree-based parallel decoding in SpecInfer [32], we can generate draft tokens for multiple sibling leaf nodes in parallel. Given the TOP-K selected siblings $N_{s_1}, N_{s_2}, \ldots, N_{s_m}$ ($m = K - 1$) of the selected leaf node N_{leaf}, we select TOP-P sibling nodes based on the normalized probability of the drafted candidate tokens. In memory-bounded decoding process, a slightly larger sequence of input tokens each step is acceptable and does not significantly affect the overall latency of draft generation. Thus, we use the same draft length k_s for all selected sibling nodes to simplify the implementation.

The algorithm is summarized in Alg. 1. The detailed algorithms for functions `FindBestLeafNode`, `FindBestConfig`, and `GenerateDraftToks` are presented in Appendix D.

Table 1: Overall speedup compared to Autoregressive Decoding on Spec-Bench. Models: `Vicuna-7B-v1.3`, `Vicuna-13B-v1.3`, and `Vicuna-33B-v1.3`. [36] **Bold** indicates the best performance among training-free methods. Underlined indicates the best overall performance, including methods with training. *CAS-Spec*† denotes CAS-Spec with Kangaroo and PLD

Model	Method	MT-Bench	Trans	Summary	QA	Math	RAG	Overall
7B	Lade	1.386	1.172	1.173	1.253	1.567	1.078	1.274
	PLD	1.563	1.046	**2.276**	1.109	1.603	1.642	1.539
	SWIFT	1.073	1.075	1.096	1.019	1.067	1.055	1.064
	CAS-Spec	**1.598**	**1.103**	2.268	**1.145**	**1.664**	**1.676**	**1.578**
	Kangaroo	1.698	1.307	1.548	1.409	1.658	1.581	1.534
	CAS-Spec†	1.727	1.312	2.327	1.407	1.701	1.695	1.696
13B	Lade	1.281	1.065	1.132	1.128	1.480	1.068	1.196
	PLD	1.418	1.020	**2.104**	1.035	1.577	1.673	1.458
	SWIFT	1.155	1.087	1.196	1.040	1.106	1.142	1.119
	CAS-Spec	**1.562**	**1.134**	2.063	**1.107**	**1.582**	**1.691**	**1.524**
	Kangaroo	1.652	1.244	1.483	1.340	1.652	1.508	1.482
	CAS-Spec†	1.732	1.251	2.337	1.401	1.719	1.689	1.673
33B	Lade	1.295	1.085	1.159	1.165	1.535	1.114	1.229
	PLD	1.431	1.047	**1.891**	1.061	1.523	1.396	1.385
	SWIFT	1.218	**1.187**	1.244	1.152	1.218	1.221	1.206
	CAS-Spec	**1.547**	1.176	1.862	**1.186**	**1.563**	**1.490**	**1.481**

5 Experiments

We conduct comprehensive experiments to evaluate the effectiveness of our proposed Cascade Adaptive Self-Speculative Decoding (CAS-Spec) algorithm. We aim to answer the two research questions posed in Section 3.

5.1 Experimental Setup

We evaluate CAS-Spec on a range of widely-used open-source LLMs, including `Llama-2-7B` [35] and `Vicuna-v1.3` [36] family. These models represent different architectures and training objectives, allowing us to assess the generalizability of CAS-Spec. All experiments ensure lossless decoding, meaning that the output is identical to that of standard autoregressive decoding. We choose Spec-Bench [26] and for evaluation. Spec-Bench is a comprehensive benchmark including various tasks such as multi-turn conversations, mathematical reasoning, and summarization. For all datasets, we measure the generation speed for producing 1024 new tokens. To demonstrate the versatility of CAS-Spec, we conduct experiments on server-grade GPU (NVIDIA H100 80GB). The primary metric is **Speedup**, defined as the wall time of autoregressive decoding divided by the wall time of the speculative decoding method.

CAS-Spec Configuration. For CAS-Spec, we construct a hierarchy of draft models using the following DSIA strategies, chosen for their training-free nature and effectiveness:

- **DSIA (Layer Sparsity):** Skip every other Transformer layer in $\mathcal{M}_t$, following SWIFT [9].
- **DSIA (Early Exiting):** Exit after a subset of Transformer layers, then decode with a trained adapter, following Kangaroo [13]. (Kangaroo is a non-training-free method and only provides the adapter weights for 7B and 13B models.)
- **bottom draft model ($\mathcal{M}_{d_n}$):** Prompt Lookup Decoding (PLD) is used as the final, fastest stage, as it has negligible computational cost.

Our CAS-Spec implementation primarily uses a three-level DSIA cascade: $\mathcal{M}_{d_1}$ and $\mathcal{M}_{d_2}$, followed by $\mathcal{M}_{d_n}$ (PLD). The DyTC algorithm dynamically selects between these options and adjusts draft lengths (k_{max} set to 5, t_{min} set to 1.1). Detailed configurations and hyperparameters for CAS-Spec are provided in Appendix E.

5.2 Main Results

Table 1 summarizes the main speedup results. CAS-Spec consistently outperforms all baseline on-the-fly (training-free) speculative decoding methods across all tested models, and datasets. CAS-Spec achieves speedups ranging from $1.10\times$ to $2.27\times$. This significantly surpasses individual training-free methods like PLD and SWIFT. Notably, CAS-Spec's performance is competitive with, and in some cases exceeds, reported numbers for Kangaroo, despite CAS-Spec being entirely training-free. With trained methods like Kangaroo, which leverage a small tuned head for early exiting, we can achieve more substantial gains over both PLD and Kangaroo. The detailed comparison between training-free and not-training-free methods is provided in Appendix F.1. As shown in Figure 3, for `Vicuna-7B-v1.3`, CAS-Spec achieves an average speedup of 47% over CS-Drafting [10](VC+HC) and 48% over tree algorithm in SWIFT [9].

5.3 Discussion

The experimental results robustly demonstrate that CAS-Spec achieves SOTA speedups among on-the-fly speculative decoding methods. **Addressing RQ1**: Our findings confirm that training-free self-speculative methods can be effectively layered to construct a cascade that significantly outperforms a single, strong statistical draft model like PLD under proper cascade scheduling. **Addressing RQ2**: The ablation study on DyTC clearly shows its superiority over static cascade scheduling. The ability to dynamically route through the draft model hierarchy and assign draft lengths based on runtime heuristics (acceptance rates and latency predictions) is crucial for maximizing performance. This adaptability allows CAS-Spec to handle variations in generation difficulty and hardware characteristics more effectively, boosting real-world applications [37, 38].

The training-free nature of CAS-Spec, combined with its high performance, makes it a practical and attractive solution for accelerating LLM inference in diverse deployment scenarios. It can be readily integrated with existing LLMs without the need for costly retraining or maintaining separate draft model weights and KV caches (beyond the DSIA modifications to the target model's inference path).

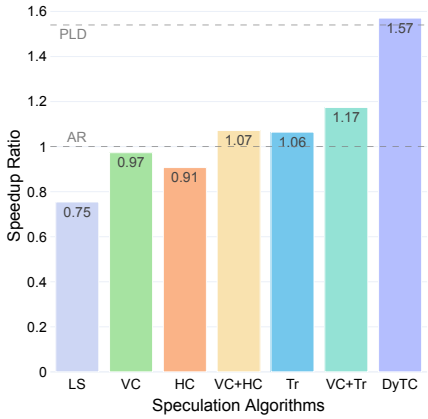


Figure 3: Speedup of different methods relative to baseline. AR (1.0) and PLD (1.54) reference lines are shown. The vertical line separates two groups of methods.

6 Conclusion

In this paper, we introduced Cascade Adaptive Self-Speculative Decoding (CAS-Spec), a novel algorithm for lossless LLM inference acceleration that eliminates the need for training separate draft models. CAS-Spec constructs a hierarchy of speculative draft models by leveraging dynamically switchable inference acceleration (DSIA) strategies, applied to the target model. This approach offers significant flexibility and ease of integration.

A core contribution of our work is the Dynamic Tree Cascade (DyTC) method. DyTC adaptively routes generation through the multi-level draft models and assigns draft lengths based on online heuristics of acceptance rates and latency predictions. This dynamic scheduling allows CAS-Spec to optimize its performance continuously during inference.

Our experiments demonstrate that CAS-Spec achieves state-of-the-art acceleration among on-the-fly speculative decoding methods. The results validated our hypotheses that (1) training-free DSIA strategies can form effective cascades over fast bottom draft models, and (2) dynamic scheduling via DyTC further enhances these gains.

CAS-Spec offers a compelling solution for practical LLM deployment by providing substantial speedups without the overheads associated with training and maintaining external draft models. Its adaptability and ease of integration make it a valuable tool for a wide range of LLMs.

References

[1] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

[2] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

[3] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

[4] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding, 2023.

[5] 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.

[6] Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3909–3925, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.257. URL https://aclanthology.org/2023.findings-emnlp.257/.

[7] 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. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11263–11282, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.607. URL https://aclanthology.org/2024.acl-long.607/.

[8] Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed Aly, Beidi Chen, and Carole-Jean Wu. LayerSkip: Enabling early exit inference and self-speculative decoding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12622–12642, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.681. URL https://aclanthology.org/2024.acl-long.681/.

[9] Heming Xia, Yongqi Li, Jun Zhang, Cunxiao Du, and Wenjie Li. SWIFT: On-the-fly self-speculative decoding for LLM inference acceleration. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=EKJhH5D5wA.

[10] Ziyi Chen, Xiaocong Yang, Jiacheng Lin, Chenkai Sun, Kevin Chen-Chuan Chang, and Jie Huang. Cascade speculative drafting for even faster llm inference. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 86226–86242. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/9cb5b083ba4f5ca6bd05dd307a2fb354-Paper-Conference.pdf.

[11] Anonymous. CLasp: In-context layer skip for self-speculative decoding. In *Submitted to ACL Rolling Review - December 2024*, 2025. URL https://openreview.net/forum?id=tlYBPa0z8z. under review.

[12] Jiahao Liu, Qifan Wang, Jingang Wang, and Xunliang Cai. Speculative decoding via early-exiting for faster LLM inference with Thompson sampling control mechanism. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3027–3043, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.179. URL https://aclanthology.org/2024.findings-acl.179/.

[13] Fangcheng Liu, Yehui Tang, Zhenhua Liu, Yunsheng Ni, Duyu Tang, Kai Han, and Yunhe Wang. Kangaroo: Lossless self-speculative decoding for accelerating LLMs via double early exiting. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=lT3oc04mDp.

[14] 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. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=CS2JWaziYr.

[15] Hanshi Sun, Zhuoming Chen, Xinyu Yang, Yuandong Tian, and Beidi Chen. Triforce: Lossless acceleration of long sequence generation with hierarchical speculative decoding. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=HVK6nl3i97.

[16] Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of llm inference using lookahead decoding. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[17] Siqi Kou, Lanxiang Hu, Zhezhi He, Zhijie Deng, and Hao Zhang. Cllms: consistency large language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[18] Feng Lin, Hanling Yi, Yifan Yang, Hongbin Li, Xiaotian Yu, Guangming Lu, and Rong Xiao. Bita: Bi-directional tuning for lossless acceleration in large language models. *Expert Systems with Applications*, 279:127305, 2025. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2025.127305. URL https://www.sciencedirect.com/science/article/pii/S0957417425009273.

[19] Juntao Zhao, Wenhao Lu, Sheng Wang, Lingpeng Kong, and Chuan Wu. Qspec: Speculative decoding with complementary quantization schemes, 2025. URL https://arxiv.org/abs/2410.11305.

[20] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: speculative sampling requires rethinking feature uncertainty. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[21] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-2: Faster inference of language models with dynamic draft trees. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7421–7432, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.422. URL https://aclanthology.org/2024.emnlp-main.422/.

[22] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test, 2025. URL https://arxiv.org/abs/2503.01840.

[23] 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. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[24] Evangelos Georganas, Dhiraj Kalamkar, Alexander Kozlov, and Alexander Heinecke. Ml-specqd: Multi-level speculative decoding with quantized drafts, 2025. URL https://arxiv.org/abs/2503.13565.

[25] Apoorv Saxena. Prompt lookup decoding. GitHub repository, 2023. URL https://github.com/apoorvumang/prompt-lookup-decoding. Accessed: 2024-01-03.

[26] 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, editors, *Findings of the Association for Computational Linguistics ACL 2024*, pages 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.

[27] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models, 2022. URL https://arxiv.org/abs/2210.11416.

[28] Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. Powerinfer: Fast large language model serving with a consumer-grade gpu. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*, SOSP '24, page 590–606, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400712517. doi: 10.1145/3694715.3695964. URL https://doi.org/10.1145/3694715.3695964.

[29] James Liu, Pragaash Ponnusamy, Tianle Cai, Han Guo, Yoon Kim, and Ben Athiwaratkun. Training-free activation sparsity in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=dGVZwyq5tV.

[30] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NG7sS51zVF.

[31] Jintao Zhang, Haofeng Huang, Pengle Zhang, Jia Wei, Jun Zhu, and Jianfei Chen. Sageattention2: Efficient attention with thorough outlier smoothing and per-thread int4 quantization. In *International Conference on Machine Learning (ICML)*, 2025.

[32] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. 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*, ASPLOS '24, page 932–949, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703867. doi: 10.1145/3620666.3651335. URL https://doi.org/10.1145/3620666.3651335.

[33] Yunfan Xiong, Ruoyu Zhang, Yanzeng Li, Tianhao Wu, and Lei Zou. Dyspec: Faster speculative decoding with dynamic token tree structure, 2024. URL https://arxiv.org/abs/2410.11744.

[34] Peter Hart, Nils Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. doi: 10.1109/tssc.1968.300136. URL https://doi.org/10.1109/tssc.1968.300136.

[35] 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.

[36] 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. URL https://lmsys.org/blog/2023-03-30-vicuna/.

[37] Jiawei Shao and Xuelong Li. Ai flow at the network edge. *IEEE Network*, pages 1–1, 2025. doi: 10.1109/MNET.2025.3541208.

[38] Hongjun An, Wenhan Hu, Sida Huang, Siqi Huang, Ruanjun Li, Yuanzhi Liang, Jiawei Shao, Yiliang Song, Zihan Wang, Cheng Yuan, Chi Zhang, Hongyuan Zhang, Wenhao Zhuang, and Xuelong Li. Ai flow: Perspectives, scenarios, and approaches, 2025. URL https://arxiv.org/abs/2506.12479.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We include a clear statement of the main claims made in the abstract and introduction. The analysis, experiment results and implications all support the claims made and our contributions. The scope of this paper is stated as well.

   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: We comprehensively discuss the limitations of our work in the paper in Appendix A.

   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: This paper includes only some computations, which are quiet simple in mathematics. We found that the provided equations and explanations may fit the readers well in understanding.

   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: We have illustrate all the important experimental settings in Experiments. Other information, such as training and evaluation details, is included in the Appendix E. Our results can be fully reproduced by following all the information.

   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: We will submit the code of the full project following the NeurIPS code and data submission guidelines. Meanwhile, the training dataset and evaluation benchmarks we used in this paper are all open-access, and related settings and processing codes are also included in the submission.

   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: We have specified all the training and test details in the Appendix E. The related codes are also submitted.

   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: [Yes]

   Justification: We do conduct extensive experiments that support the main claims of the paper and report all the results in Section 5 and Appendix F.

   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: We implement experiments on H100 80GB for different backbone LLMs. We have displayed the detailed information of setting in Section 5.

   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: We have reviewed the code to confirm with the NeurIPS Code 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: [NA]

Justification: There is no direct societal impact from our work. Our contribution focuses solely on improving the inference efficiency of large language models through a decoding speedup technique. The content or accuracy of the generated responses—including any potential misinformation or ethical concerns—is entirely determined by the underlying backbone model, which our method does not modify or influence.

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 does not pose 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: Part of our training code and the used data and benchmark datasets are from open-access projects and works. We have cited the responding works and authors.

    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: [Yes]

    Justification: We have included a tutorial of our released

    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.

A Limitations and Future Work

While CAS-Spec demonstrates promising performance, its effectiveness is inherently tied to the quality and efficiency of the chosen DSIA strategies. The online estimation for DyTC, though lightweight, introduces some computational overhead and require a short warm-up period for optimal performance. Moreover, since dynamic scheduling depends on the heuristics of the generation of a certain token sequence, its improvement is lessened when the batch size is large. Other limitations include the incompatibility of CAS-Spec with current SOTA speculative decoding methods like EAGLE3 due to their reliance on hidden states of the verify model.

Future research could explore integrating more sophisticated or newly developed training-free SSD techniques as DSIA components within the CAS-Spec framework. Further refinement of the DyTC algorithm, potentially incorporating more advanced online learning for scheduling, could yield additional improvements. Investigating the application of CAS-Spec to even larger models or different modalities, and exploring hardware co-design to optimize DSIA strategy execution, are also promising avenues for future work. As the field of self-speculative decoding continues to evolve, CAS-Spec provides a flexible and powerful framework to leverage these advancements.

B Theoretical Analysis

For certain hyperparameters $(k_0, n, k, k_{d1}, k_{d2})$, we can derive the theoretical bounds for the cost coefficient of $\mathcal{M}_{d_1}$, such that the speedup of the cascade is greater than the speedup of standard speculative decoding with $\mathcal{M}_{d_2}$ alone.

For *vertical cascade*, the solution of the inequality $T_{VC(\mathcal{M}_{d_1},\mathcal{M}_{d_2})} \geq T_{SD(\mathcal{M}_{d_2})}$ gives the bound:

$$c_{d_1} \leq \frac{1}{n}\left[\left(\frac{1-\alpha_{d_1}(\phi(\alpha_{d_1}))^n}{1-\alpha_{d_1}}\right)\left(\frac{(1-\alpha_{d_2})(c_{d_2}k_0+1)}{1-\alpha_{d_2}^{k_0+1}}\right)-(1+nkc_{d_2})\right]$$

where $\phi(\alpha_{d_1}) = \frac{1-\alpha_{d_1}^{k_{d_1}+1}}{(1-\alpha_{d_1})(1+k_{d_1}c_{d_1})}$.

For *horizontal cascade*, the solution of the inequality $T_{HC(\mathcal{M}_{d_1},\mathcal{M}_{d_2})} \geq T_{SD(\mathcal{M}_{d_2})}$ gives the bound:

$$c_{d_1} \leq \frac{1}{k_{d_1}}\left[\left(\frac{1-\alpha_{d_1}^{k_{d_1}+1}}{1-\alpha_{d_1}}+\alpha_{d_1}^{k_{d_1}}\frac{\alpha_{d_2}(1-\alpha_{d_2}^{k_{d_2}})}{1-\alpha_{d_2}}\right)\left(\frac{(1-\alpha_{d_2})(c_{d_2}k_{d_2}+1)}{1-\alpha_{d_2}^{k_{d_2}+1}}\right)-(1+k_{d_2}c_{d_2})\right]$$

C DSIA

Other than the layer sparsity, we also consider the activation quantization and activation sparsity as two candidates of CAS-Spec framework. For activation quantization, we refer to the implementation of QSpec. While for activation sparsity, we adopt the TEAL's method of this DSIA strategy.

Furthermore, since the activation quantization and activation sparsity are both orthogonal to the layer sparsity, we can construct draft models with Mixing-DSIA Cascade. For instance, we can construct $\mathcal{M}_{d_1}$ as the W4A4 quantized model, and $\mathcal{M}_{d_2}$ as the W4A4 model with layer sparsity and activation sparsity.

However, using these two DSIA strategies inherently requires a weight-only quantized draft model with batch size of 1. This configuration is suitable for edge inference, but not general enough for large-scale inference scenarios. Therefore, we do not include the experiments on these two DSIA strategies in our CAS-Spec framework by now.

D Dynamic Tree Cascade Algorithm

Maintaining Estimates for Inactive Configurations. For draft configurations not currently selected in a given step, their acceptance rate estimates are preserved and do not decay. When a previously unused configuration becomes active again (e.g., due to changing generation context), its estimate is updated based on actual verification outcomes. In cold-start scenarios or for configurations that have

never been used, we initialize estimates using either: (1) brief offline profiling on representative tasks, or (2) heuristic priors based on the DSIA strategy's aggressiveness (e.g., higher layer sparsity implies lower expected acceptance rate).

Note that for vertical cascade configurations like $VC(\mathcal{M}_{d_i}, \mathcal{M}_{d_n})$, we maintain a single acceptance rate estimate corresponding to the highest-level draft model $\mathcal{M}_{d_i}$, since all draft tokens are ultimately verified by this model before proceeding to $\mathcal{M}_t$.

The detailed functions mentioned in the Algorithm 1 are as follows:

Algorithm 2: Dynamic Tree Cascade (DyTC) Functions

Function `FindBestConfigurationForStep`($\mathcal{S}_{candidates}, \hat{\alpha}_{d_n}, \hat{c}_{d_n}, k_{max}$)**:**
  $\mathcal{M}_{best\_choice} \leftarrow$ null; $k_{best\_choice} \leftarrow 0$; $max\_obj\_val \leftarrow -\infty$
  foreach *configuration* $S \in \mathcal{S}_{candidates}$ **do**
    $\hat{\alpha}_S \leftarrow$ current estimate $\hat{\alpha}(S)$
    $\hat{c}_S \leftarrow$ current estimate $\hat{c}(S)$
    for $k \leftarrow 1$ **to** k_{max} **do**
      if $\hat{c}_S k + \hat{c}_{d_n} \approx 0$ **then**
        continue
      float $E_{accepted}$; **if** $\hat{\alpha}_S \approx 1.0$ **then**
        $E_{accepted} \leftarrow k$
      else
        $E_{accepted} \leftarrow (\hat{\alpha}_S(1 - \hat{\alpha}_S^k))/(1 - \hat{\alpha}_S)$
      float $\mathcal{T}_{val} \leftarrow (E_{accepted} + \hat{\alpha}_S^k \hat{\alpha}_{d_n})/(\hat{c}_S k + \hat{c}_{d_n})$
      if $\mathcal{T}_{val} > max\_obj\_val$ **then**
        $max\_obj\_val \leftarrow \mathcal{T}_{val}$
        $\mathcal{M}_{best\_choice} \leftarrow S$
        $k_{best\_choice} \leftarrow k$
  if $max\_obj\_val \leq 0$ **then**
    return (null, 0)
  return $(\mathcal{M}_{best\_choice}, k_{best\_choice})$

E Experimental Details

In this section, we provide the details of our experimental setup, specifically for CAS-Spec configuration in the main experiment. Our draft models are constructed using mainly layer sparsity based on SWIFT, which is a training-free and on-the-fly DSIA strategy. The hierarchy is got by Scaling-DSIA Cascade, which tunes the layer sparsity to get different draft models. We also consider the Prompt Lookup Decoding (PLD) as the bottom draft model. The configuration of CAS-Spec is as follows: $\mathcal{M}_{d_1}$ has around 0.4 layer sparsity, while $\mathcal{M}_{d_2}$ has around 0.6 layer sparsity. Thus it leads to multiple candidate $\mathcal{M}_{d_s}$ for each step:

- basic model: $\mathcal{M}_{d_1}(LS = 0.4)$, $\mathcal{M}_{d_2}(LS = 0.6)$, $\mathcal{M}_{d_3}$ (PLD)
- 2-Level VC: $VC\left(\mathcal{M}_{d_1}, \mathcal{M}_{d_3}\right)$, $VC\left(\mathcal{M}_{d_2}, \mathcal{M}_{d_3}\right)$
- 3-Level VC: $VC\left(\mathcal{M}_{d_1}, VC(\mathcal{M}_{d_2}, \mathcal{M}_{d_3})\right)$

However, due to the insufficient gap between the layer sparsity of $\mathcal{M}_{d_1}$ and $\mathcal{M}_{d_2}$, the 3-Level VC configuration turns out to be less efficient and hence rarely used in the DyTC algorithm. Therefore, we present the results of 2-Level VC in the main experiment.

F Additional Experimental Results

Figure 3 shows the speedup of different methods on `Vicuna-7B-v1.3` model. `LS` refers to the layer sparsity only drafting without tree attention. `VC,HC,VC+HC` are the framework of using the vertical

cascade and horizontal cascade with PLD, based on the implementation of CS-Drafting. `Tr` is the standard SWIFT implementation with Tree Attention. `Tr+VC` follows the same implementation of CS-Drafting, but with the tree attention. `DyTC` is the version with Dynamic Tree Cascade algorithm. Compared to `VC+HC` configuration, `DyTC` achieves a 73% improvement in average speedup. Compared to `Tr` (SWIFT), `DyTC` achieves a 47% improvement in average speedup.

F.1 Comparison with Trained Methods

Table 2: Comparison with trained methods on `Vicuna-7B-v1.3` model.

Method	Training-Free	#Mean accepted tokens	Speedup
PLD	Yes	1.75	1.54x
SWIFT	Yes	3.01	1.06x
CAS-Spec (SWIFT,PLD)	Yes	3.43	1.58x
Speculative Decoding (Vicuna 68m)	No	2.27	1.44x
Medusa	No	2.39	1.69x
EAGLE	No	3.57	2.05x
EAGLE2	No	4.36	2.21x

In comparison with trained speculative decoding methods, our CAS-Spec framework with SWIFT and PLD achieves higher speedup than vanilla speculative decoding methods with Vicuna 68m, while being training-free. The gap between training-free and not-training-free methods is significant, as trained methods like Medusa and EAGLE achieve higher acceptance rates and speedups, leveraging the full model's capabilities.