PARALLELSPEC: PARALLEL DRAFTER FOR EFFICIENT SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative decoding has proven to be an efficient solution to large language model (LLM) inference, where the small drafter predicts future tokens at a low cost, and the target model is leveraged to verify them in parallel. However, most existing works still draft tokens auto-regressively to maintain sequential dependency in language modeling, which we consider a huge computational burden in speculative decoding. We present PARALLELSPEC, an alternative to auto-regressive drafting strategies in state-of-the-art speculative decoding approaches. In contrast to auto-regressive drafting in the speculative stage, we train a parallel drafter to serve as an efficient speculative model. PARALLELSPEC learns to efficiently predict multiple future tokens in parallel using a single model, and it can be integrated into any speculative decoding framework that requires aligning the output distributions of the drafter and the target model with minimal training cost. Experimental results show that PARALLELSPEC accelerates baseline methods in latency up to 62% on text generation benchmarks on Medusa and 9-17% on EAGLE. It also achieves 2.84× overall speedup on the Llama-2-13B model using third-party evaluation criteria.

1 INTRODUCTION

Large language models (LLMs) such as GPT-4 (OpenAI, 2023) and Llama (Touvron et al., 2023) have shown dominant abilities across various domains, such as question answering (Zhuang et al., 2023), code synthesis (Rozière et al., 2023), machine translation (Zhang et al., 2023a) and beyond. However, their auto-regressive nature requires multiple forward passes on models with billions or trillions of parameters, bringing substantial inference latency, thus prohibiting real-time applications. In the pursuit of accelerating LLM inference, various strategies have been explored, including utilizing model sparsity (Liu et al., 2023; Sun et al., 2024; Schuster et al., 2022; Cai et al., 2024a), exploiting redundancy in KV Cache (Cai. et al., 2024; Zhang et al., 2023b; Li et al., 2024a), and distilling model capabilities to smaller models (Gu et al., 2024; Agarwal et al., 2024). While these approaches can lead to faster inference, they often come at the cost of reduced generation quality and do not preserve the generation distributions of the original models.

Speculative decoding (SD) (Leviathan et al., 2023; Chen et al., 2023a) has been proposed as one of the compelling alternatives to auto-regressive generation in a lossless manner. The key motivation behind SD is to utilize a low-cost small model to generate draft tokens efficiently and then use the target model to verify them in parallel to ensure sampling integrity, known as *draft-then-verify* framework. While promising, we observe that draft models in most *draft-then-verify* frameworks still generate token by token, resulting in a low arithmetic intensity during the drafting stage. Moreover, the forward latency of the drafting stage still grows linearly with respect to the draft length, *i.e.*, the number of tokens each draft step generates. As empirically profiled in the right part of Figure 1, the draft latency still accounts for a significant proportion of the overall SD latency.

Researchers have spotted the draft latency issue and proposed several solutions to alleviate it by dynamically determining the draft length either through learnable policy (Huang et al., 2024a), confidence-guided heuristics (Li et al., 2024c), or optimized draft tree structures (Wang et al., 2024a). Nevertheless, these solutions do not tackle the underlying issue of drafting latency scaling linearly with the draft length, but operate only for maximally reducing the compute wasted in drafting tokens that are unlikely to be accepted.

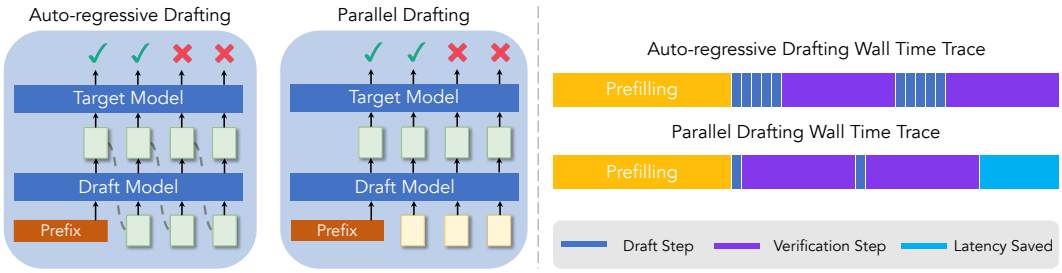

Figure 1: Illustration of PARALLELSPEC. The Left part of the figure has been revised to highlight the difference between the two drafting styles. **Left:** comparison between auto-regressive drafting and our proposed parallel drafting. Blocks in green indicate normal draft tokens. Blocks in yellow denote the mask tokens used to prompt the draft model to generate multiple future tokens in a single forward pass. **Right:** wall time trace diagrams for two drafting styles integrated with EAGLE (Li et al., 2024b) in two rounds of speculative sampling, given the assumption that both drafting styles have the same speculation accuracy on the prefix sequence.

To fundamentally solve the draft latency issue, we propose building a parallel-decoding drafter as the replacement for auto-regressive drafters in popular SD frameworks. Unlike Medusa-style frameworks (Cai et al., 2024b; Ankner et al., 2024) that rely on separate language model heads to decode future tokens, we propose to use a single lightweight model to decode the next k tokens simultaneously. We argue that using a single model for multi-token prediction can effectively leverage parameter sharing to achieve efficient drafter alignment rather than learning several independent language model heads. The latter design would struggle even more with memory and computation in the large vocabulary size (128,000+) of recently introduced language models such as Llama-3 (AI@Meta, 2024). For efficient multi-token alignment training, we introduce a group-wise parallel training strategy that mitigates possible training-inference mismatches by dynamically adjusting the attention mask, positional indexes, and token layout.

Our method still adheres to the *draft-and-verify* framework at inference time: At each draft step, the drafter generates k tokens with a single forward pass, and then they are sent to the target model for parallel verification. Since our method incorporates token-level parallelization in the drafting stage, both stages in the SD pipeline now benefit from this parallelization. Therefore, we name our approach PARALLELSPEC. PARALLELSPEC works as an individual module, ready to replace any drafter in existing SD frameworks that require distilling drafters from their target models. We experiment with the popular speculative decoding framework Medusa (Cai et al., 2024b) and state-of-the-art solution EAGLE (Li et al., 2024b) by replacing their drafters. Experimental results show that PARALLELSPEC is able to bring consistent acceleration improvement in all task domains and different combinations of models. For instance, incorporating PARALLELSPEC into Vicuna-7B Medusa increases the average speedup ratio from 1.42× to 2.31×, leading to a 62.7% relative improvement. This empirically validates the superiority of parallel drafter design. PARALLELSPEC integration with EAGLE also achieves extra speedups across all target model settings, ranging from 2.55× to 2.84×, with a relative improvement ranging from 9% to 17%. In summary, our key contributions are as follows:

- We propose PARALLELSPEC as a parallel multi-token drafter to replace the auto-regressive drafter design in existing SD frameworks that require aligning drafters with their target models.
- We design a group-wise training strategy that allows efficient and accurate parallel drafter training.
- We integrate the proposed method into two popular SD frameworks. Extensive experiments demonstrate the compatibility and performance superiority of PARALLELSPEC.

2 RELATED WORKS

Accelerating Large Language Model (LLM) Inference has attracted considerable research attention from both machine learning system and natural language processing communities and even led

the trend of hardware-software co-design. These research efforts include model compression (Sun et al., 2024; Huang et al., 2024b; Ma et al., 2023), novel architecture design (Gu & Dao, 2023; Peng et al., 2023) and hardware optimization (Dao et al., 2022; Hong et al., 2023). However, some of these methods could lead to generation discrepancies compared to the target model, representing a trade-off between model performance and inference speed. We consider those methods that cannot generate with the target model's original distribution as lossy acceleration methods.

Speculative Decoding (SD) (Leviathan et al., 2023; Chen et al., 2023a) arises as one of the lossless acceleration methods for LLM inference. It is based on the observation that the latency bottleneck of LLM inference is brought by memory bandwidth instead of arithmetic computation. SD alleviates the bandwidth bottleneck by utilizing a small model to draft multiple tokens and verifying them in parallel with the *target* model, thereby reducing the frequency of language model calls and decreasing the memory access density during decoding. The community has witnessed many improvements in efficiently and accurately drafting tokens. Self-speculative decoding methods (Hooper et al., 2023; Elhoushi et al., 2024; Zhang et al., 2024a; Bhendawade et al., 2024) do not explicitly rely on draft models but use some of the intermediate layers of the target model to draft. Medusa-style methods add independent (Cai et al., 2024b) or sequential (Ankner et al., 2024) decoding heads on the target model to draft tokens. Lookahead decoding and its variants (Fu et al., 2024; Zhao et al., 2024) use n-gram trajectory as drafts. DistillSpec (Zhou et al., 2024) leverages knowledge distillation to closely align distributions between the draft and target models. PEARL (Liu et al., 2024) proposes two-stage verification to alleviate the mutual waiting problem. Apart from efficient draft methods, token tree verification (Miao et al., 2024; Sun et al., 2023) has been widely adopted for verifying top candidate sequences that share common prefixes in parallel. Specialized SD frameworks for long-context generation (Chen et al., 2024a;b), retrieval-augmented generation (He et al., 2024; Wang et al., 2024b; Zhang et al., 2024b) and beyond (Chen et al., 2023b) have been proposed to better fit individual use cases.

Parallel Decoding was first known for its efficiency in machine translation system (Ghazvininejad et al., 2019) and code generation (Gloeckle et al., 2024) as an alternative to auto-regressive generation. However, its usage in SD frameworks remains under-explored. Cai et al. (2024b) and Stern et al. (2018) utilize parallel language model heads to predict multiple tokens at different positions. Santilli et al. (2023) proposed to use fixed-point iterations to replace auto-regressive decoding. Monea et al. (2023); Yi et al. (2024) pioneered the use of parallel decoding in SD, but it is limited to a self-speculative framework and results in different generation sampling. BiTA (Lin et al., 2024) proposed using prompt tuning to train a small number of prompt parameters on a frozen target LM for semi-autoregressive generation. Wu et al. (2024) suggested using trainable linear projection to regress intermediate hidden states of target models, thereby enabling multi-token prediction. However, these methods either fail to effectively learn the draft distribution due to the limited number of learnable parameters or cannot achieve lossless acceleration due to the method design.

3 Background: Speculative Decoding

Notation. Speculative decoding (SD) frameworks maintain two models: the *target* model, denoted as $\mathcal{M}_{\mathrm{T}}$, is the one which we want the SD frameworks to sample from; the *draft* model, denoted as $\mathcal{M}_{\mathrm{D}}$, is the one that proposes candidate tokens which are later being verified by the *target* model. Let $x_{<t}$ be the prompt sequence we are running the SD framework on, $p\left(x_t \mid x_{<t}\right)$, $q\left(x_t \mid x_{<t}\right)$ be the inference distribution of $\mathcal{M}_{\mathrm{T}}$ and $\mathcal{M}_{\mathrm{D}}$ given the prompt $x_{<t}$, respectively. We use the denotations of $p\left(y_t\right)$ and $q\left(y_t\right)$ to indicate $p\left(y_t \mid x, y_{<t}\right)$ and $q\left(y_t \mid x, y_{<t}\right)$ whenever they do not lead to confusion.

Speculative Decoding Procedures. One round of speculative decoding can be divided into the drafting and verification stages, each governed by the corresponding model. The drafting stage auto-regressively calls $\mathcal{M}_{\mathrm{D}}$ to sample γ candidate token distributions $q\left(y_t\right), \ldots, q\left(y_{t+\gamma-1}\right)$. The verification stage calls $\mathcal{M}_{\mathrm{T}}$ once to sample γ distributions from the target model, $p\left(y_t\right), \ldots, p\left(y_{t+\gamma-1}\right)$, given y_{t+i} is the concatenation of $y_{<t}$ and drafted token sequence $x_1, \ldots, x_i$, where $x_k \sim q\left(y_{t+k}\right)$. The verification stage determines whether the token y_{t+i} is accepted via speculative sampling, where its acceptance rate α_i is defined as:

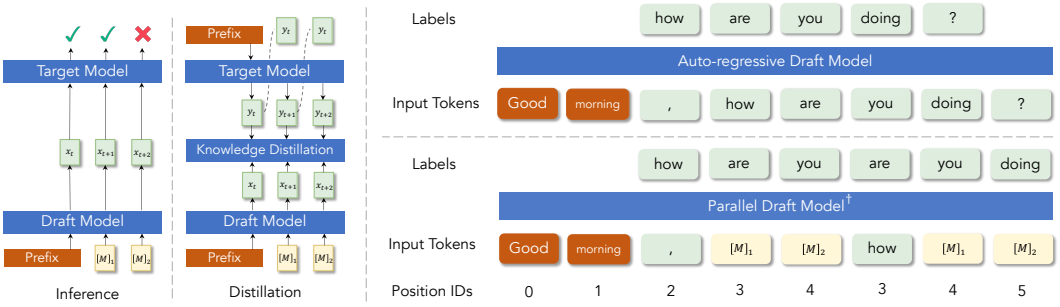

Figure 2: Illustration of parallel drafter inference, training, and the difference between training auto-regressive drafter and parallel one. **Left:** Parallel drafter proposes multiple candidate tokens with a single forward pass. **Middle:** Training the parallel drafter to align with the target model is a process of knowledge distillation (KD). **Right:** The input, labels, and position indices for training a parallel drafter need special treatment. † refers to Figure 3 for the special attention mask design of parallel training.

$$\alpha_i = \begin{cases} 1 & p\left(y_{t+i}\right) \geq q\left(y_{t+i}\right) \\ \frac{p(y_{t+i})}{q(y_{t+i})} & p\left(y_{t+i}\right) < q\left(y_{t+i}\right) \end{cases} \tag{1}$$

If the token y_{t+i} is rejected before γ candidate tokens are all accepted, the remaining draft tokens will be discarded, and y_{t+i} will be resampled from $\max\left(0, p\left(y_{t+i}\right) - q\left(y_{t+i}\right)\right)$. Otherwise, drafted tokens are all accepted and SD samples an extra token from $y_{t+\gamma}$ and appends it to the end of the sequence. Each round of speculative decoding generates at least 1 and at most $\gamma + 1$ tokens, and Leviathan et al. (2023) theoretically proves that the sequence from SD and the sequence from the target model follow the same distribution.

Average Acceptance Length. One important efficiency metric to measure an SD system is the acceptance rate of drafted tokens in each round of speculative decoding. Since each drafting step takes a constant amount of time and each round of conventional SD takes the same drafting steps, the overall efficiency of an SD system will be determined by the average acceptance length τ measured on some prompt sequences.

Token Tree Verification. Prior studies (Miao et al., 2024; Leviathan et al., 2023) suggest that verifying multiple candidate sequences within the same verification step could greatly improve the expected acceptance length. This is achieved by the tree attention mechanism. By properly arranging the top predicted tokens at different token positions and manipulating the attention mask based on a tree structure, we enable the processing of multiple candidate sequences with only one verification step. We refer to the details of token tree verification in Appendix A.3.

4 Methodology

In this section, we describe our parallel drafting method (§4.1), the algorithm that preserves the output distribution of the target model with parallel drafter (§4.2) and its integration into popular speculative sampling methods (§4.3).

4.1 Parallel Drafting

Inference. Let $\mathcal{M}_{\mathrm{D}}^{\theta}$ be the parallel drafter parameterized by θ, and $q^{\theta}\left(x_t, x_{t+1}, \ldots, x_{t+k} \mid x_{<t}\right)$ be the multi-token output distribution of the parallel drafter. Naive draft models do not support predicting multiple tokens in a single forward pass. In order to equip a drafter with such abilities, we propose to use customized `[MASK]` tokens as prompt tokens to produce contextualized token-level representations that are used to enforce multi-token training. The parallel drafter introduces k special tokens in its vocabulary, $\texttt{[MASK]}_1, \ldots, \texttt{[MASK]}_k$. At each drafting step where the drafter is invoked, k special tokens are concatenated after the original input sequence $x_{<t}$. Apart from

the last token representation that is used to decode the next token x_t, the last-layer representations corresponding to `[MASK]` tokens are leveraged to sample future tokens $x_{t+1}, \ldots, x_{t+k}$. The left part of Figure 2 demonstrates how the parallel draft model proposes 3 future tokens with 2 `[MASK]` prompt tokens.

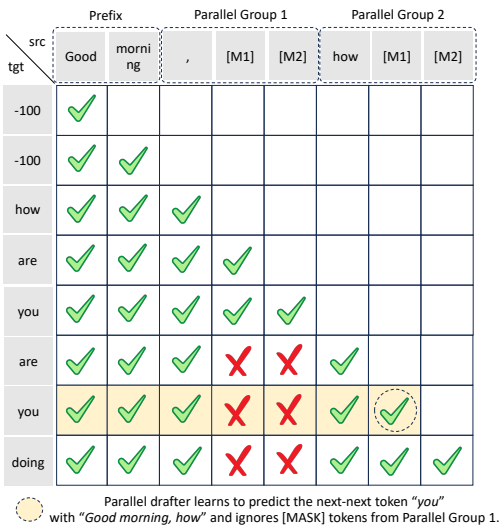

Figure 3: Attention mask illustration of parallel drafter training. ✓ denotes activated attention. ✗ denotes attention suppressed to prevent access across parallel groups. -100 denotes ignored tokens in the target sequence that do not contribute to training loss. Blocks with yellow and the legend illustrate one of the next-next token prediction training objectives.

Group-wise Parallel Training. Given the access to either the ground truth future tokens $y_t, y_{t+1}, \ldots, y_{t+k}$ or their output distributions of the *target* model $p(y_t), p(y_{t+1}), \ldots, p(y_{t+k})$, training a parallel drafter is a process of knowledge distillation (KD) in either online or offline setting (Zhou et al., 2024), which we defer to §4.3 for detailed discussion. This process is not as trivial as training an auto-regressive drafter, where teacher-forcing supervision is enforced via *shift-one-token* in labels as depicted in the upper right of Figure 2. Simply shifting the label position by k tokens would result in discrepancies between training and inference. Therefore, We carefully design a group-wise training paradigm that eliminates training-inference mismatch by manipulating the token layout, attention masks, and position indices, illustrated in the lower right of Figure 2. Specifically, we introduce the concept of *parallel group*, illustrated in Figure 3, where each *parallel group* in the source sequence consists of an input token and several `[MASK]` tokens. At each training step, a customized causal attention mask guarantees that all training token pairs ignore `[MASK]` tokens from previous *parallel group*s to ensure the model behaves the same in training and evaluation.

4.2 Speculative Sampling with Parallel Drafting

In order to preserve the output distribution of the target model as standard speculative sampling does, following Monea et al. (2023), we present a modified version of speculative sampling (Chen et al., 2023a) that drafts with a parallel decoder and verifies with the target model using token tree verification method (Miao et al., 2024), detailed in Algorithm 1.

4.3 Integration with Popular Speculative Decoding Frameworks

ParallelSpec can be inserted into any speculative decoding (SD) framework that requires aligning output distributions of the drafter and the target model. We choose two popular SD frameworks as testbeds, Medusa (Cai et al., 2024b) and EAGLE (Li et al., 2024b).

Medusa leverages an *offline* knowledge distillation method, where cross-entropy loss between Medusa heads and ground truth tokens is used for alignment. Specifically, K extra decoding heads are added to decode the last hidden states of the target model, and the k-th head is used to predict the $(t+k+1)$-th token given a prefix sequence of length t. The final training objective of Medusa is expressed as:

$$\mathcal{L}_{\text{MEDUSA}} = \sum_{k=1}^{K} -\lambda_k \log p_t^{(k)}\left(y_{t+k+1}\right), \tag{2}$$

where λ_k is a coefficient to balance learning difficulties, $p_t^{(k)}$ is the output distribution of the k-th head and y_{t+k+1} is the oracle token.

Algorithm 1 Speculative Sampling with Parallel Draft Models and Token Tree Verification

Given k special tokens $\texttt{[MASK]}_1, \ldots, \texttt{[MASK]}_k$ and minimum target sequence length T.
Given the target model distribution $p(\cdot|\cdot)$, the parallel draft model distribution $q(\cdot|\cdot)$ and initial prefix sequence $x_{<n}$.
Return the generated sequence y.
Initialise $n \leftarrow t$, $y \leftarrow x_{<n}$.
while $n < T$ **do**
  Draft future token tree $\mathcal{N}$ by sampling from the parallel draft model with a single forward pass:

$$\mathcal{N} \sim (q(x|x_{<n}), q(x|x_{<n}, \texttt{[MASK]}_1), \ldots, q(x|x_{<n}, \texttt{[MASK]}_1, \ldots, \texttt{[MASK]}_k))$$

  In parallel, compute a set of logits $\mathcal{O}$ with the target model using $\mathcal{N}$ and tree attention.

$$\mathcal{O} = \text{TreeParallelDecode}(\mathcal{N}, p)$$

  $\mathcal{V} = \emptyset, u$ ▷ u is the root node of $\mathcal{N}$
  while u is not a leaf node **do**
    $\mathcal{H} = \text{Child}(u)$ ▷ $\text{Child}(u)$ returns the child nodes of u
    while $\mathcal{H}$ is not empty **do**
      Sample $r \sim U[0,1]$; $s = \text{Select}(\mathcal{H})$; $\tilde{x}_s = \mathcal{H}(s)$; $t = \text{TreeDepth}(s)$
      if $r < \min\left(1, \dfrac{p(\tilde{x}_s|x_{<n}, \mathcal{V})}{q(\tilde{x}_s|x_{<n}, \ldots, \texttt{[MASK]}_{t-1})}\right)$ **then**
        ✓ accept the draft token x_s at depth t
        $\mathcal{V}$.append(x_s); $u = s$; $n \leftarrow n + 1$
        break
      else
        × reject the draft token x_s. Normalize the residual.
        $p(x|x_{<n}, \ldots, \mathcal{V}) := (p(x|x_{<n}, \ldots, \mathcal{V}) - q(x|x_{<n}, \ldots, \texttt{[MASK]}_{t-1}))_+$
        $\mathcal{H}$.pop(s)
      end if
    end while
    if $\mathcal{H}$ is empty **then**
      break
    end if
  end while
  $x_{\text{next}} \sim p(x|x_{<n}, \ldots, \mathcal{V})$; $n \leftarrow n + 1$; $\mathcal{V}$.append(x_{next})
  $y \leftarrow y + \mathcal{V}$
end while

5 EXPERIMENTS

This section describes the experimental settings (§5.1), including training and evaluation datasets, metrics, and involved baselines. §5.2 reports the main results for PARALLELSPEC compared with baselines. Finally, we discuss the impact of different experiment settings in §5.3.

To integrate PARALLELSPEC into Medusa, we introduce a Transformer model specialized in multi-token prediction to replace K Medusa heads as the new draft model. The draft model shares the embedding layer and the language model head with the target model to minimize memory overhead, and they remain frozen to preserve the target model's output distribution. K trainable [MASK] tokens are added to the embedding layer of the draft model to facilitate parallel training. During the training, each *parallel group* is trained with a similar objective denoted in Equation 2, except that the log term now denotes the output distribution of the parallel drafter at position k, and we need to consider the non-mask token at the beginning of each *parallel group*:

$$\mathcal{L}_{\text{MEDUSA-Parallel}} = -\log q\left(y_{t+1}|x_{<t}\right) - \sum_{k=1}^{K} \lambda_k \log q\left(y_{t+k+1}|x_{<t}, \texttt{[MASK]}_1, \ldots, \texttt{[MASK]}_k\right). \quad (3)$$

EAGLE utilizes an *online* knowledge distillation method that directly regresses the last-layer hidden states at the feature level. Specifically, we denote the last-layer hidden states of the target model at t-th position as f_t, the embedding of t-th token as e_t, and the oracle token as y_t. EAGLE proposed to use a fully connected layer and an auto-regression head $\pi\left(\tilde{f}_t|f_{<t}, e_{<t}\right)$ as the draft model to predict the next feature that is used to decode draft tokens. The training objective of EAGLE on each token position is a linear combination of the regression loss $\mathcal{L}_{\text{reg}}$ and the cross-entropy loss $\mathcal{L}_{\text{cls}}$ between draft tokens and oracle tokens:

$$\begin{aligned} \mathcal{L}_{\text{reg}} &= \text{SmoothL1}\left(f_{t+1}, \pi\left(\tilde{f}_t|f_{<t}, e_{<t}\right)\right), \\ \mathcal{L}_{\text{cls}} &= -\log \mu\left(y_{t+1}\right), \end{aligned} \quad (4)$$

where μ denotes the language model head distribution conditioned on drafted feature $\tilde{f}_t$.

Integrating PARALLELSPEC into EAGLE is more intuitive, as it only requires minor modifications to turn the auto-regression head into a parallel head with the method outlined in §4.1, without extra adaptation. For a parallel group of size K starting at token position t, the training loss of EAGLE-Parallel is the sum of EAGLE losses (defined in Equation 4) over K tokens within the same group. At inference time, EAGLE integration needs an additional effort. Each drafting step uses only the embedding of `[MASK]` tokens with the target features of `[MASK]` tokens left empty, *i.e.*:

$$\tilde{f}_t, \ldots, \tilde{f}_{t+K+1} = \pi\left(\left[f_{<t}, 0, \ldots 0\right]; \left[e_{<t}, e_{[MASK_1]}, \ldots, e_{[MASK_K]}\right]\right). \quad (5)$$

This is because these introduced `[MASK]` tokens are not in the original vocabulary of the target models; therefore, there is no way to produce target features for these `[MASK]` tokens. We only use the trainable embeddings of `[MASK]` as a signal for the drafter to predict tokens at different future time steps.

5.1 SETTINGS

Datasets, Tasks, and Training. Following the setup of SpecBench (Xia et al., 2024), we conduct evaluations on six types of text generation tasks, including MT-bench (Zheng et al., 2023) for multi-turn conversation, CNN/Daily Mail (Nallapati et al., 2016) for text summarization, Natural Questions (Karpukhin et al., 2020) for retrieval-augmented generation and question answering, WMT14 DE-EN (Bojar et al., 2014) for machine translation, GSM8K (Cobbe et al., 2021) for mathematical reasoning. As PARALLELSPEC falls into the category of speculative decoding methods that require an extra alignment stage, supervised fine-tuning (i.e., SFT) data are needed for aligning distributional similarity between the drafter and the target model. To ensure a fair comparison with baselines in this category, we follow Li et al. (2024b) to use 68,000 ShareGPT (Tay et al., 2023) conversations as training data without self-distillation. For the self-distillation setting where multi-turn conversations are distilled from the target model given the prompts from the dataset, we refer to §5.3. Due to the group-wise parallel training strategy, the training sequences for PARALLELSPEC will become longer than the ones of conventional auto-regressive drafter. Training PARALLELSPEC on 7B models takes 13 hours on 8 A100-PCIE-40GB GPUs for 40 epochs. The size of *parallel group* is set to 5, *i.e.*, the number of `[MASK]` tokens $k = 4$ unless stated otherwise. We refer to Appendix A.2 for details such as computing environment, other hyper-parameters, *etc*.

Baselines. We select seven competitive speculative decoding methods as baselines. Some of them work as plugin modules like Speculative Sampling (SpS) (Chen et al., 2023a), Prompt Lookup Decoding (PLD) (Saxena, 2023; Yang et al., 2023) and Lookahead Decoding (Fu et al., 2024), which

[1] While we name this model Medusa + PARALLELSPEC, we clarify that only Medusa-style loss is used in the Medusa PARALLELSPEC integration, and no more Medusa heads are utilized as drafter.

Model	Method	Multi-turn Conversation	Translation	Summarization	Question Answering	Mathematical Reasoning	Retrieval-aug. Generation	Avg.	τ (tokens)
Temperature = 0.0									
V 7B	SpS	1.67×$_{\pm 0.04}$	1.13×$_{\pm 0.02}$	1.71×$_{\pm 0.01}$	1.49×$_{\pm 0.04}$	1.50×$_{\pm 0.03}$	1.67×$_{\pm 0.02}$	1.53×$_{\pm 0.03}$	2.27
	PLD	1.61×$_{\pm 0.02}$	1.02×$_{\pm 0.01}$	2.57×$_{\pm 0.02}$	1.14×$_{\pm 0.02}$	1.61×$_{\pm 0.01}$	1.82×$_{\pm 0.06}$	1.62×$_{\pm 0.01}$	1.75
	Hydra	2.50×$_{\pm 0.02}$	1.94×$_{\pm 0.03}$	1.89×$_{\pm 0.04}$	2.02×$_{\pm 0.04}$	2.53×$_{\pm 0.02}$	1.86×$_{\pm 0.07}$	2.13×$_{\pm 0.04}$	3.26
	Lookahead	1.48×$_{\pm 0.02}$	1.15×$_{\pm 0.02}$	1.36×$_{\pm 0.02}$	1.27×$_{\pm 0.02}$	1.59×$_{\pm 0.03}$	1.23×$_{\pm 0.03}$	1.35×$_{\pm 0.02}$	1.64
	Medusa	1.60×$_{\pm 0.01}$	1.39×$_{\pm 0.01}$	1.22×$_{\pm 0.03}$	1.37×$_{\pm 0.00}$	1.68×$_{\pm 0.01}$	1.20×$_{\pm 0.06}$	1.42×$_{\pm 0.01}$	2.39
	+ParallelSpec [1]	2.63×$_{\pm 0.02}$	1.97×$_{\pm 0.03}$	2.32×$_{\pm 0.04}$	2.20×$_{\pm 0.02}$	2.78×$_{\pm 0.03}$	1.98×$_{\pm 0.03}$	2.31×$_{\pm 0.02}$	3.31
	Medusa†	1.87×$_{\pm 0.01}$	1.56×$_{\pm 0.01}$	1.49×$_{\pm 0.02}$	1.56×$_{\pm 0.02}$	1.85×$_{\pm 0.04}$	1.42×$_{\pm 0.02}$	1.63×$_{\pm 0.02}$	2.31
	EAGLE-2	2.68×$_{\pm 0.05}$	1.78×$_{\pm 0.04}$	2.23×$_{\pm 0.03}$	2.04×$_{\pm 0.04}$	2.69×$_{\pm 0.04}$	2.02×$_{\pm 0.03}$	2.24×$_{\pm 0.04}$	4.34
	EAGLE	2.57×$_{\pm 0.02}$	1.85×$_{\pm 0.04}$	2.17×$_{\pm 0.05}$	2.03×$_{\pm 0.04}$	2.57×$_{\pm 0.05}$	1.92×$_{\pm 0.04}$	2.18×$_{\pm 0.04}$	**3.58**
	+ParallelSpec	**3.01**×$_{\pm 0.04}$	**2.09**×$_{\pm 0.01}$	**2.62**×$_{\pm 0.06}$	**2.40**×$_{\pm 0.03}$	**2.84**×$_{\pm 0.05}$	**2.36**×$_{\pm 0.02}$	**2.55**×$_{\pm 0.03}$	3.52
L2 7B	SpS	1.33×$_{\pm 0.03}$	1.25×$_{\pm 0.03}$	1.21×$_{\pm 0.02}$	1.30×$_{\pm 0.01}$	1.34×$_{\pm 0.02}$	1.43×$_{\pm 0.03}$	1.31×$_{\pm 0.02}$	1.67
	PLD	1.42×$_{\pm 0.01}$	1.17×$_{\pm 0.02}$	1.44×$_{\pm 0.02}$	1.07×$_{\pm 0.02}$	1.31×$_{\pm 0.02}$	1.57×$_{\pm 0.01}$	1.33×$_{\pm 0.01}$	1.42
	Lookahead	1.46×$_{\pm 0.05}$	1.36×$_{\pm 0.04}$	1.34×$_{\pm 0.04}$	1.32×$_{\pm 0.03}$	1.47×$_{\pm 0.04}$	1.37×$_{\pm 0.03}$	1.39×$_{\pm 0.04}$	1.60
	EAGLE	2.61×$_{\pm 0.02}$	2.38×$_{\pm 0.02}$	2.25×$_{\pm 0.02}$	2.30×$_{\pm 0.05}$	2.66×$_{\pm 0.06}$	2.23×$_{\pm 0.01}$	2.40×$_{\pm 0.02}$	**3.55**
	+ParallelSpec	**2.95**×$_{\pm 0.03}$	**2.67**×$_{\pm 0.01}$	**2.64**×$_{\pm 0.03}$	**2.76**×$_{\pm 0.04}$	**2.88**×$_{\pm 0.02}$	**2.52**×$_{\pm 0.03}$	**2.74**×$_{\pm 0.03}$	3.49
V 13B	SpS	1.69×$_{\pm 0.01}$	1.16×$_{\pm 0.01}$	1.78×$_{\pm 0.00}$	1.45×$_{\pm 0.01}$	1.60×$_{\pm 0.01}$	1.76×$_{\pm 0.04}$	1.57×$_{\pm 0.01}$	2.18
	Medusa	2.06×$_{\pm 0.01}$	1.77×$_{\pm 0.02}$	1.66×$_{\pm 0.04}$	1.74×$_{\pm 0.01}$	2.12×$_{\pm 0.01}$	1.62×$_{\pm 0.05}$	1.84×$_{\pm 0.02}$	2.39
	+ParallelSpec	2.76×$_{\pm 0.04}$	2.37×$_{\pm 0.05}$	2.16×$_{\pm 0.02}$	2.33×$_{\pm 0.03}$	2.62×$_{\pm 0.03}$	2.27×$_{\pm 0.04}$	2.52×$_{\pm 0.05}$	3.34
	EAGLE	2.78×$_{\pm 0.02}$	2.03×$_{\pm 0.03}$	2.41×$_{\pm 0.02}$	2.11×$_{\pm 0.03}$	2.78×$_{\pm 0.04}$	2.20×$_{\pm 0.03}$	2.39×$_{\pm 0.02}$	**3.64**
	+ParallelSpec	**3.03**×$_{\pm 0.04}$	**2.30**×$_{\pm 0.03}$	**2.65**×$_{\pm 0.02}$	**2.36**×$_{\pm 0.03}$	**3.04**×$_{\pm 0.05}$	**2.46**×$_{\pm 0.04}$	**2.64**×$_{\pm 0.02}$	3.56
L2 13B	SpS	1.38×$_{\pm 0.03}$	1.30×$_{\pm 0.03}$	1.26×$_{\pm 0.04}$	1.36×$_{\pm 0.03}$	1.41×$_{\pm 0.02}$	1.47×$_{\pm 0.06}$	1.36×$_{\pm 0.03}$	1.66
	EAGLE	2.80×$_{\pm 0.01}$	2.60×$_{\pm 0.02}$	2.53×$_{\pm 0.06}$	2.42×$_{\pm 0.03}$	2.85×$_{\pm 0.03}$	2.39×$_{\pm 0.12}$	2.60×$_{\pm 0.03}$	**3.66**
	+ParallelSpec	**3.02**×$_{\pm 0.02}$	**2.81**×$_{\pm 0.03}$	**2.77**×$_{\pm 0.07}$	**2.68**×$_{\pm 0.03}$	**3.00**×$_{\pm 0.02}$	**2.74**×$_{\pm 0.04}$	**2.84**×$_{\pm 0.04}$	3.60
Temperature = 1.0									
V 7B	SpS	1.35×$_{\pm 0.00}$	1.01×$_{\pm 0.00}$	1.39×$_{\pm 0.02}$	1.25×$_{\pm 0.01}$	1.29×$_{\pm 0.02}$	1.38×$_{\pm 0.05}$	1.28×$_{\pm 0.01}$	1.82
	PLD	1.56×$_{\pm 0.01}$	0.98×$_{\pm 0.01}$	2.49×$_{\pm 0.01}$	1.12×$_{\pm 0.00}$	1.56×$_{\pm 0.01}$	1.73×$_{\pm 0.01}$	1.57×$_{\pm 0.00}$	1.70
	Lookahead	1.43×$_{\pm 0.00}$	1.10×$_{\pm 0.01}$	1.32×$_{\pm 0.00}$	1.21×$_{\pm 0.01}$	1.53×$_{\pm 0.01}$	1.16×$_{\pm 0.00}$	1.29×$_{\pm 0.00}$	1.64
	EAGLE	2.10×$_{\pm 0.01}$	1.59×$_{\pm 0.02}$	1.83×$_{\pm 0.05}$	1.70×$_{\pm 0.02}$	2.04×$_{\pm 0.02}$	1.78×$_{\pm 0.06}$	1.84×$_{\pm 0.01}$	**3.18**
	+ParallelSpec	**2.32**×$_{\pm 0.02}$	**1.78**×$_{\pm 0.03}$	**2.06**×$_{\pm 0.04}$	**1.89**×$_{\pm 0.02}$	**2.10**×$_{\pm 0.01}$	**1.96**×$_{\pm 0.02}$	**2.02**×$_{\pm 0.03}$	3.09
L2 7B	SpS	1.11×$_{\pm 0.00}$	1.07×$_{\pm 0.01}$	1.04×$_{\pm 0.02}$	1.09×$_{\pm 0.01}$	1.13×$_{\pm 0.01}$	1.15×$_{\pm 0.01}$	1.10×$_{\pm 0.01}$	1.47
	EAGLE	2.19×$_{\pm 0.02}$	1.92×$_{\pm 0.05}$	1.91×$_{\pm 0.03}$	1.93×$_{\pm 0.05}$	2.31×$_{\pm 0.05}$	1.87×$_{\pm 0.07}$	2.02×$_{\pm 0.04}$	**3.30**
	+ParallelSpec	**2.47**×$_{\pm 0.03}$	**2.15**×$_{\pm 0.02}$	**2.08**×$_{\pm 0.04}$	**2.11**×$_{\pm 0.03}$	**2.42**×$_{\pm 0.01}$	**2.06**×$_{\pm 0.03}$	**2.22**×$_{\pm 0.02}$	3.25

Table 1: Speedup ratios and average acceptance lengths τ of different methods tested on an A100-PCIE-40GB GPU using third-party benchmark toolkit SpecBench (Xia et al., 2024). V: Vicuna-v1.3. L2: LLaMA2-Chat. We report the mean and standard deviation of speedup ratios on 3 different runs. Best metrics for each model are marked in **boldface**. † denotes additional evaluation that runs on an RTX-4090 GPU.

do not need additional training. For SpS methods, we use `vicuna-68m`[2] and `llama-68m`[3] as the drafter for Vicuna and Llama target models, respectively. SpS implementation strictly follows Spec-Bench (Xia et al., 2024) and Huggingface (Wolf et al., 2019) assisted_generation setup, where the number of draft tokens per step γ is updated with heuristic rules. For PLD, we follow the default settings of n-gram size $=3$ and number of lookup tokens $=10$. For Lookahead Decoding, we use the official recommended configuration of level $=5$, window size $=7$, and n-gram size $=7$. The remaining methods, including Medusa (Cai et al., 2024b), Hydra (Ankner et al., 2024), and EAGLE (Li et al., 2024b), that require extra training while preserving the output distributions, are the main peer works for comparison. We use their official drafter checkpoints to report results.

Models. We conduct experiments on the Vicuna series (7B, 13B) (Zheng et al., 2023) and the Llama-2-Chat series (7B, 13B) (Touvron et al., 2023). We chose these models as they are highly representative, and most prior methods built their drafters upon these models, allowing a fair comparison. We provide results for more recent target models in §5.3. All parallel drafters are constructed with a single Transformer layer, with hyper-parameters identical to those of layers in their target models. This results in a 202M drafter for 7B models and a 317M one for 13B models. We keep the same draft token tree structure with the selected two baseline methods in ParallelSpec.

Metrics. Similar to other speculative decoding methods, we primarily focus on the end-to-end wall-time speedup ratio compared to naive auto-regressive decoding. We also report the average acceptance length τ in each round of speculative decoding.

[2] https://huggingface.co/double7/vicuna-68m
[3] https://huggingface.co/JackFram/llama-68m

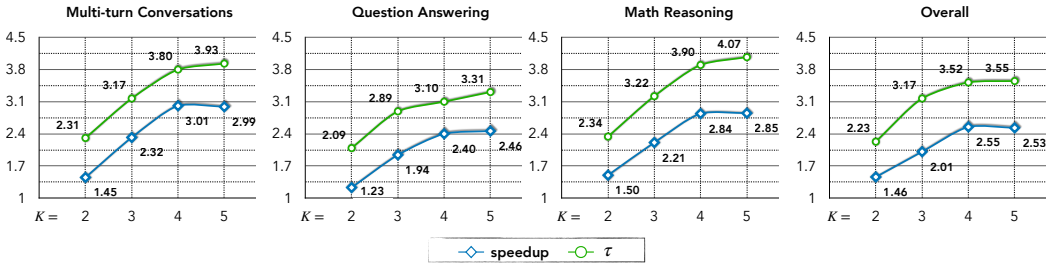

Figure 4: Ablations on speedup ratio and average acceptance length τ with respect to the number of [MASK] tokens K on all three test datasets.

Model & Method	Multi-turn Conversation	Translation	Summarization	Question Answering	Mathematical Reasoning	Retrieval-aug. Generation	Avg.	τ (tokens)
LLaMA 2 7B w/ EAGLE	2.61×	2.38×	2.25×	2.30×	2.66×	2.23×	2.40×	3.55
+PARALLELSPEC	2.95× +13.0%	2.67× +12.2%	2.64× +17.3%	2.76× +20.0%	2.88× +8.3%	2.52× +13.0%	2.74× +14.2%	3.49
+PARALLELSPEC +self-distillation	**3.02**× +15.7%	**2.77**× +17.3%	**2.71**× +20.4%	**2.87**× +24.8%	**2.98**× +12.0%	**2.56**× +14.8%	**2.82**× +17.5%	**3.78**

Table 2: Ablations on alignment training data.

5.2 MAIN RESULTS

Table 1 gives a performance overview of PARALLELSPEC and established prior approaches on different types of text generation benchmarks. We refer to Appendix A.1 for qualitative case studies. In general, PARALLELSPEC integration brings a consistent acceleration improvement in all domains, two selected methods, and different decoding temperatures. Based on the results, we have the following key observations:

PARALLELSPEC significantly accelerates Medusa frameworks by a large margin. For example, integrating PARALLELSPEC into Vicuna-7B Medusa boosts the average speedup ratio from 1.42× to 2.31×, resulting in a 62.7% relative improvement. This even outperforms EAGLE, which can be attributed to the parallel drafter design. It not only increases the average acceptance tokens from 2.39 to 3.31 but also significantly reduces drafter runtime overhead. Similar improvements are also observed on Vicuna-13B Medusa, showing PARALLELSPEC is not sensitive to the target model size.

Equipping EAGLE with PARALLELSPEC also achieves considerable speedups. For instance, the average speedup ratio on Vicuna-7B increased from 2.18× to 2.55×; on Llama2-7B-Chat, it rose from 2.40× to 2.74×; and on LLaMA-13B-Chat, it went from 2.60× to 2.84×. It is worth noting that incorporating PARALLELSPEC into EAGLE causes a slight drop in average acceptance length. This is fully expected, as we no longer preserve the sequence dependency during the draft stage, and the parallel decoding leads to a small decline in drafting accuracy. In addition, we also conduct experiments on EAGLE-PARALLELSPEC with temperature decoding settings and observe substantial overall speedup up to 2.22×. Still, the speculative decoding efficiency at high temperature degrades compared with greedy decoding, echoing conclusions in previous studies (Xia et al., 2024).

5.3 ABLATION STUDIES

Training Data. Prior works often assume the availability of training data that aligns with the output distribution of target models, which is not usually the case. Thus, directly using ShareGPT conversations as SFT data for drafter training may suffer from the domain shift. In order to bypass this, we follow Cai et al. (2024b) to employ the self-distillation technique to build the training dataset that matches the target model. Specifically, we employ vLLM (Kwon et al., 2023) to obtain distilled multi-turn conversations by feeding the prompts from ShareGPT in a greedy decoding setting. Table 2 demonstrates that the self-distillation technique can further improve the speedup ratio, with the average acceptance tokens greatly increased to 3.78.

Size of Parallel Group. The size of the *parallel group* is an important hyper-parameter of our method, and it determines how many tokens the parallel drafter will predict at each step. It equals

Model & Method	Multi-turn Conversation	Translation	Summarization	Question Answering	Mathematical Reasoning	Retrieval-aug. Generation	Avg.	τ (tokens)
LLaMA 3 8B w/ EAGLE	2.66×	2.42×	2.33×	2.32×	2.75×	2.32×	2.47×	**3.42**
+PARALLELSPEC	**3.03**×$_{+13.9\%}$	**2.63**×$_{+8.7\%}$	**2.61**×$_{+12.0\%}$	**2.83**×$_{+22.0\%}$	**2.92**×$_{+6.2\%}$	**2.45**×$_{+5.6\%}$	**2.75**×$_{+11.3\%}$	3.36

Table 3: Ablations on recent advanced target models.

the number of [MASK] tokens K plus one. To investigate its impact on the speedup performance, we conduct ablation studies to re-train parallel drafter with different K and report the speedup ratio and average acceptance length in Figure 4. We notice the increase in the average acceptance length τ resulting from larger *parallel group* size is steady for small K but saturates after $K = 4$, leading the speedup ratio to no longer improve beyond that point. We believe this could be attributed to the difficulty of predicting distant future tokens using parallel decoding. The benefit of increasing the *parallel group* size cannot counterbalance the overhead of predicting distant tokens, resulting in a speedup ratio sweet spot around $K = 4$.

Advanced Target Model. Table 3 reflects that more advanced models like LLaMA3-8B-Instruct (AI@Meta, 2024) can still benefit from the design of PARALLELSPEC. However, the relative improvements on LLaMA3-Instruct series are slightly lower than those on LLaMA2-Chat and Vicuna, possibly because of the larger misalignment between ShareGPT SFT data and LLaMA3-Instruct.

6 CONCLUSION

In this paper, we introduce PARALLELSPEC, a powerful speculative decoding solution that could be inserted into popular speculative decoding frameworks. It proposes to use a single lightweight model and several trainable [MASK] tokens to facilitate fast multi-token prediction as drafters, thereby mitigating the issue of drafting latency scaling linearly with the draft length. Compared with the Medusa-style multi-head multi-token draft strategy, PARALLELSPEC demonstrates significant advantages in drafting accuracy, latency, and parameter efficiency. Extensive experiments on various benchmarks and different target models demonstrate the compatibility and performance superiority of PARALLELSPEC.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

AI@Meta. Llama 3 model card. 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. Hydra: Sequentially-dependent draft heads for medusa decoding, 2024.

Nikhil Bhendawade, Irina Belousova, Qichen Fu, Henry Mason, Mohammad Rastegari, and Mahyar Najibi. Speculative streaming: Fast llm inference without auxiliary models. *arXiv preprint arXiv: 2402.11131*, 2024.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Ale s Tamchyna. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pp. 12–58, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics.

Ruisi Cai, Yuandong Tian, Zhangyang Wang, and Beidi Chen. Lococo: Dropping in convolutions for long context compression. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024a.

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 *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024b.

Zefan Cai., Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Wen Xiao. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv: 2406.02069*, 2024.

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*, 2023a.

Jian Chen, Vashisth Tiwari, Ranajoy Sadhukhan, Zhuoming Chen, Jinyuan Shi, Ian En-Hsu Yen, and Beidi Chen. Magicdec: Breaking the latency-throughput tradeoff for long context generation with speculative decoding. *arXiv preprint arXiv: 2408.11049*, 2024a.

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable, robust, and hardware-aware speculative decoding. *CoRR*, abs/2402.12374, 2024b.

Ziyi Chen, Xiaocong Yang, Jiacheng Lin, Chenkai Sun, Kevin Chen-Chuan Chang, and Jie Huang. Cascade speculative drafting for even faster llm inference. *arXiv preprint arXiv: 2312.11462*, 2023b.

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. *arXiv preprint arXiv: 2110.14168*, 2021.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed A Aly, Beidi Chen, and Carole-Jean Wu. Layerskip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv: 2404.16710*, 2024.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of LLM inference using lookahead decoding. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 6112–6121, Hong Kong, China, November 2019. Association for Computational Linguistics.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv: 2404.19737*, 2024.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *CoRR*, abs/2312.00752, 2023.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason D. Lee, and Di He. REST: retrieval-based speculative decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pp. 1582–1595. Association for Computational Linguistics, 2024.

Ke Hong, Guohao Dai, Jiaming Xu, Qiuli Mao, Xiuhong Li, Jun Liu, Kangdi Chen, Yuhan Dong, and Yu Wang. Flashdecoding++: Faster large language model inference on gpus. *arXiv preprint arXiv: 2311.01282*, 2023.

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

Kaixuan Huang, Xudong Guo, and Mengdi Wang. Specdec++: Boosting speculative decoding via adaptive candidate lengths. *CoRR*, abs/2405.19715, 2024a.

Wei Huang, Yangdong Liu, Haotong Qin, Ying Li, Shiming Zhang, Xianglong Liu, Michele Magno, and Xiaojuan Qi. Billm: Pushing the limit of post-training quantization for llms. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024b.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, Online, November 2020. Association for Computational Linguistics.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 2023.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *arXiv preprint arXiv: 2404.14469*, 2024a.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: speculative sampling requires rethinking feature uncertainty. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-2: faster inference of language models with dynamic draft trees. *CoRR*, abs/2406.16858, 2024c.

Feng Lin, Hanling Yi, Hongbin Li, Yifan Yang, Xiaotian Yu, Guangming Lu, and Rong Xiao. Bita: Bi-directional tuning for lossless acceleration in large language models. *CoRR*, abs/2401.12522, 2024.

Tianyu Liu, Yun Li, Qitan Lv, Kai Liu, Jianchen Zhu, and Winston Hu. Parallel speculative decoding with adaptive draft length. *arXiv preprint arXiv: 2408.11850*, 2024.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*, 2023.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

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 2024, La Jolla, CA, USA, 27 April 2024- 1 May 2024*, pp. 932–949. ACM, 2024.

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

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 280–290, Berlin, Germany, August 2016. Association for Computational Linguistics.

OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023.

Bo Peng, Eric Alcaide, Quentin Gregory Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Nguyen Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartłomiej Koptyra, Hayden Lau, Jiaju Lin, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Johan S. Wind, Stanisław Woźniak, Zhenyuan Zhang, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. RWKV: Reinventing RNNs for the transformer era. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code. *arXiv preprint arXiv: 2308.12950*, 2023.

Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodola. Accelerating transformer inference for translation via parallel decoding. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12336–12355, Toronto, Canada, July 2023. Association for Computational Linguistics.

Apoorv Saxena. Prompt lookup decoding, November 2023. URL https://github.com/apoorvumang/prompt-lookup-decoding/.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Benjamin Spector and Chris Re. Accelerating llm inference with staged speculative decoding. *arXiv preprint arXiv:2308.04623*, 2023.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 10107–10116, 2018.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix X. Yu. Spectr: Fast speculative decoding via optimal transport. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Steven Tay et al. Sharegpt, 2023. URL https://sharegpt.com/.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023.

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. *CoRR*, abs/2406.17276, 2024a. doi: 10.48550/ARXIV.2406.17276.

Zilong Wang, Zifeng Wang, Long Le, Huaixiu Steven Zheng, Swaroop Mishra, Vincent Perot, Yuwei Zhang, Anush Mattapalli, Ankur Taly, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. Speculative rag: Enhancing retrieval augmented generation through drafting. *arXiv preprint arXiv: 2407.08223*, 2024b.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv: 1910.03771*, 2019.

Pengfei Wu, Jiahao Liu, Zhuocheng Gong, Qifan Wang, Jinpeng Li, Jingang Wang, Xunliang Cai, and Dongyan Zhao. Parallel decoding via hidden transfer for lossless large language model acceleration. *arXiv preprint arXiv: 2404.12022*, 2024.

Heming Xia, 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 *Findings of the Association for Computational Linguistics ACL 2024*, pp. 7655–7671, Bangkok, Thailand and virtual meeting, August 2024. Association for Computational Linguistics.

Nan Yang, Tao Ge, Liang Wang, Binxing Jiao, Daxin Jiang, Linjun Yang, Rangan Majumder, and Furu Wei. Inference with reference: Lossless acceleration of large language models. *arXiv preprint arXiv: 2304.04487*, 2023.

Hanling Yi, Feng Lin, Hongbin Li, Ning Peiyang, Xiaotian Yu, and Rong Xiao. Generation meets verification: Accelerating large language model inference with smart parallel auto-correct decoding. In *Findings of the Association for Computational Linguistics ACL 2024*, pp. 5285–5299, Bangkok, Thailand and virtual meeting, August 2024. Association for Computational Linguistics.

Biao Zhang, Barry Haddow, and Alexandra Birch. Prompting large language model for machine translation: A case study. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 41092–41110. PMLR, 2023a.

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 *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11263–11282, Bangkok, Thailand, August 2024a. Association for Computational Linguistics.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark W. Barrett, Zhangyang Wang, and Beidi Chen. H2O: heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023b.

Zhihao Zhang, Alan Zhu, Lijie Yang, Yihua Xu, Lanting Li, Phitchaya Mangpo Phothilimthana, and Zhihao Jia. Accelerating iterative retrieval-augmented language model serving with speculation. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024b.

Weilin Zhao, Yuxiang Huang, Xu Han, Wang Xu, Chaojun Xiao, Xinrong Zhang, Yewei Fang, Kaihuo Zhang, Zhiyuan Liu, and Maosong Sun. Ouroboros: Generating longer drafts phrase by phrase for faster speculative decoding. *arXiv preprint arXiv: 2402.13720*, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. Distillspec: Improving speculative decoding via knowledge distillation. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. Toolqa: A dataset for LLM question answering with external tools. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

A APPENDIX

A.1 CASE STUDY

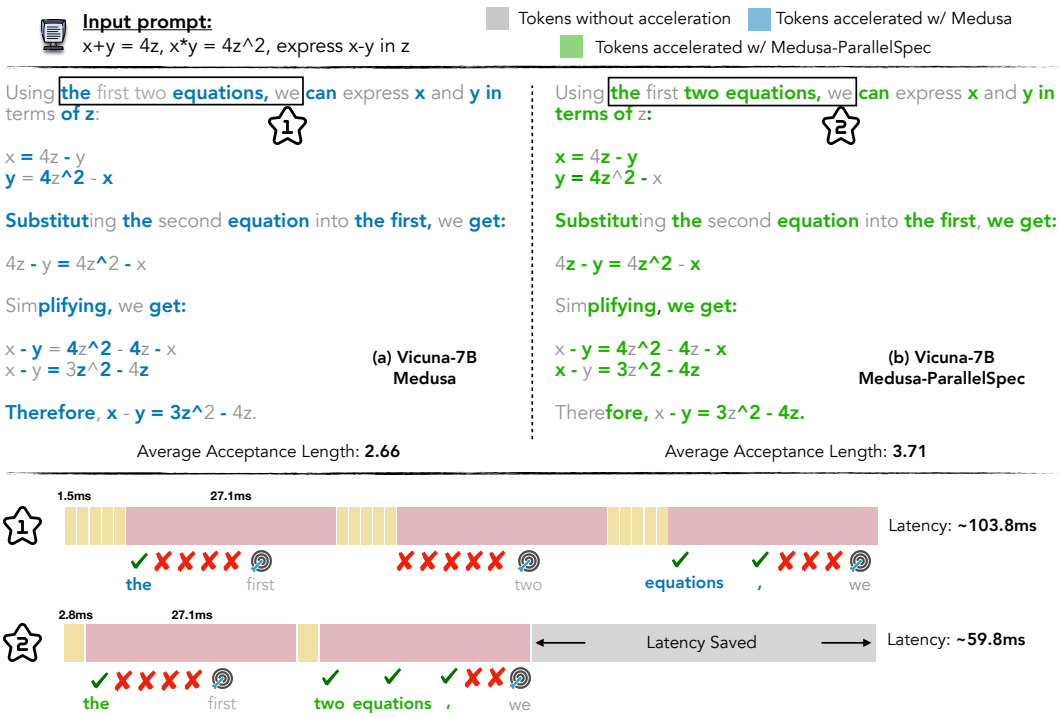

Figure 5: **Upper:** Visualization of accelerated tokens in generation from (a) Vicuna-7B Medusa and (b) Vicuna-7B Medusa-PARALLELSPEC given an input prompt from GSM8K (Cobbe et al., 2021). **Lower:** Simulated wall-time trace of two different methods generating the text in the highlighted box. We only consider the forward pass latency of draft and verification while ignoring the negligible post-processing overhead. ✓: accepted draft tokens. ✗: rejected draft tokens. : tokens without speculative acceleration.

We provide an illustration of two different speculative decoding methods running on the same prompt from GSM8K in Figure 5. The side-by-side comparison indicates that Vicuna-7B Medusa equipped with PARALLELSPEC not only achieves a higher average acceptance length (3.71 vs. 2.66) but also shows a significant advantage in end-to-end latency, thanks to the one-pass decoding nature of parallel-decoding drafter. The lower part of Figure 5 even reveals that PARALLELSPEC nearly cut half of the time cost when decoding the same text span compared with the baseline method.

A.2 EXPERIMENTAL DETAILS

Hyper Parameter	Medusa-PARALLELSPEC	EAGLE-PARALLELSPEC
Global Batch Size	32	32
Learning Rate	5×10^{-4}	5×10^{-5}
Optimizer	AdamW ($\beta_1 = 0.9, \beta_2 = 0.999$)	AdamW($\beta_1 = 0.9, \beta_2 = 0.95$)
Weight Decay	0.0	0.0
Epochs	20	40

Table 4: Hyper parameters of PARALLELSPEC variants for 7B models.

All training was conducted using a node with 8 NVIDIA A100-PCIE-40GB GPUs, 2 AMD EPYC 7282 CPUs, and 512GB RAM. Evaluations were conducted using one GPU of the above node. PyTorch 2.2.0 with CUDA 12.1 version was used in all experiments. To avoid the discrepancies brought by computing environment differences, we re-ran all baseline methods and our method 3 times and reported the mean speedup ratio. We list hyper-parameters used in PARALLELSPEC in Table 4. One might notice that the baseline results have around 10% differences in terms of speedup ratios compared with the original Spec-Bench (Xia et al., 2024). First, we refer you to the latest benchmark page[4] for updated results. We also appreciate your understanding that, due to budget and practical constraints, we do not have access to the same computational resources as the Spec-Bench team. Therefore, we were not able to reproduce their speedup results. However, since all the improvements in this paper were obtained using the same hardware environment, our comparisons are relatively fair. We also notice that in the Spec-Bench paper, Table 5 and Table 6 also reported entirely different sets of speedup ratios when the only difference is the GPU used for benchmarking, indicating the hardware specifications are one of the major factors that impact reported speedup.

A.3 TOKEN TREE VERIFICATION

Initially proposed in SpecInfer (Miao et al., 2024), and also explored in several follow-up works (Cai et al., 2024b; Li et al., 2024b; Spector & Re, 2023), tree-based structure for token verification is proved to be useful. Following existing studies, PARALLELSPEC leverages tree attention to realize this process. Specifically, to guarantee that each token only accesses its predecessors, we use an attention mask that exclusively permits attention flow from the current token back to its antecedent tokens. The positional indices for positional encoding are adjusted in line with this structure. A conceptual view of this process is visualized in Figure 6, with the draft tree structure in the figure being adopted in all experiments. As the main contribution of this paper is not a novel token tree verification strategy, we adopt the same static token trees used in Medusa-1 (Cai et al., 2024b) and EAGLE (Li et al., 2024b). Starting from "Root", every node is expanded with k tokens with top-k highest probabilities. k is designed based on manually crafted rules, and is dynamically changing as the tree depth grows.

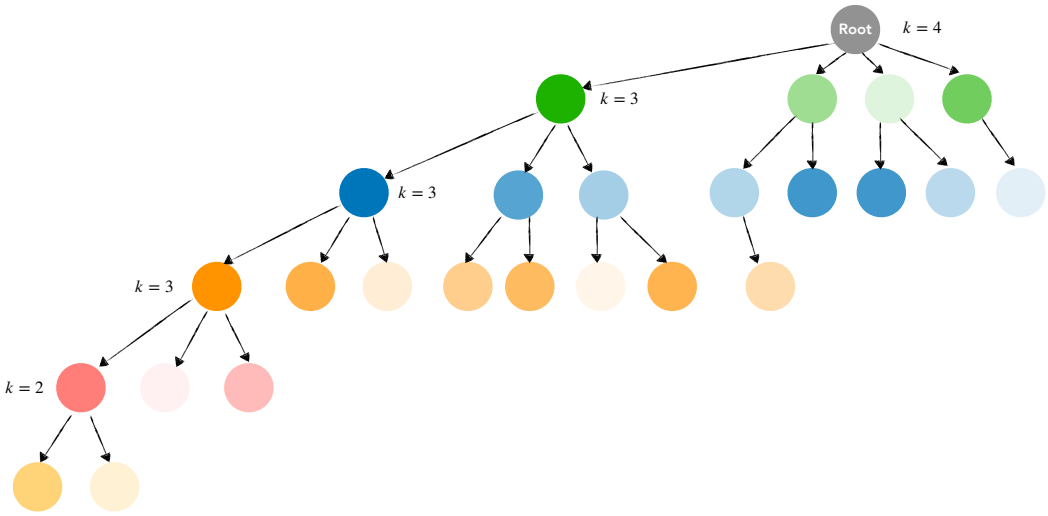

Figure 6: An example of the tree-structured verification process. Each circle represents a token, and the shading of the color indicates the probability of each token in the distribution. Tokens with the highest probability are selected and gradually expanded with dynamically designed k.

[4] https://github.com/hemingkx/Spec-Bench/blob/main/Leaderboard.md