Unlocking Efficiency in Large Language Model Inference: A Comprehensive Survey of Speculative Decoding

Anonymous ACL submission

Abstract

To mitigate the high inference latency stemming from autoregressive decoding in Large Language Models (LLMs), Speculative Decoding has emerged as a novel decoding paradigm for LLM inference. In each decoding step, this method first drafts several future tokens efficiently and then verifies them in parallel. Unlike autoregressive decoding, Speculative Decoding facilitates the simultaneous decoding of multiple tokens per step, thereby accelerating inference. This paper presents a comprehensive overview and analysis of this promising decoding paradigm. We begin by providing a formal definition and formulation of Speculative Decoding. Then, we organize in-depth discussions on its key facets, such as drafter selection and verification strategies. Furthermore, we present a comparative analysis of leading methods under third-party testing environments. We aim for this work to serve as a catalyst for further research on Speculative Decoding, ultimately contributing to more efficient LLM inference.[1]

1 Introduction

Large Language Models (LLMs) have achieved remarkable proficiency in a range of downstream tasks (OpenAI, 2023; Touvron et al., 2023a,b; Chiang et al., 2023; Jiang et al., 2023). They are progressively evolving as the cornerstone of comprehensive API interfaces (e.g., ChatGPT[2]), offering human life services and guidance through real-time human-machine interactions. However, the inference latency of these sizable models has emerged as a substantial obstacle restricting their broader applications. This latency primarily arises from the token-by-token generation necessitated by autoregressive decoding, resulting in an escalation of the inference latency with both the length of the generated sequence and the model's scale.

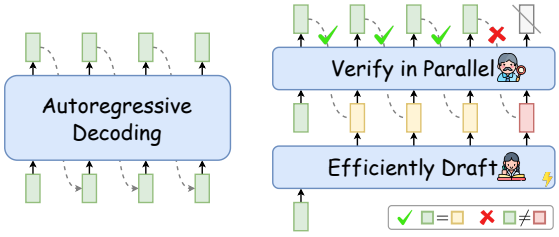

Figure 1: In contrast to autoregressive decoding (*left*) that generates sequentially, Speculative Decoding (*right*) first *efficiently drafts* multiple tokens and then *verifies* them *in parallel* using the target LLM. Drafted tokens after the bifurcation position (*e.g.,*) will be discarded to guarantee the generation quality.

To accelerate LLM inference, an innovative inference paradigm, Speculative Decoding has been introduced (Stern et al., 2018; Xia et al., 2022; Leviathan et al., 2023; Chen et al., 2023a). As shown in Figure 1, in each decoding step, Speculative Decoding first efficiently drafts multiple tokens as speculation of future decoding steps of the target LLM and then utilizes the LLM to verify all drafted tokens in parallel. Only those tokens that meet the LLM's verification criterion are accepted as final outputs to guarantee generation quality.

Speculative Decoding is founded upon two key observations about LLM inference: 1) many easy tokens can be predicted with less computational overhead (e.g., using a smaller model), and 2) LLM inference is highly memory bandwidth bound (Patterson, 2004) with the main latency bottleneck arising from memory reads/writes of LLM parameters rather than arithmetic computations. Drawing on these observations, Speculative Decoding adapts the concept of *speculative execution*[3] to focus LLMs' computational efforts on the validation

[1] Our evaluation scripts and dataset will be released to facilitate future research.

[2] https://chat.openai.com

[3] Speculative execution (Burton, 1985; Hennessy and Patterson, 2012) is an optimization technique used in computer architecture where tasks are performed in advance and subsequently verified for their necessity, thereby circumventing the delays inherent in sequential task execution.

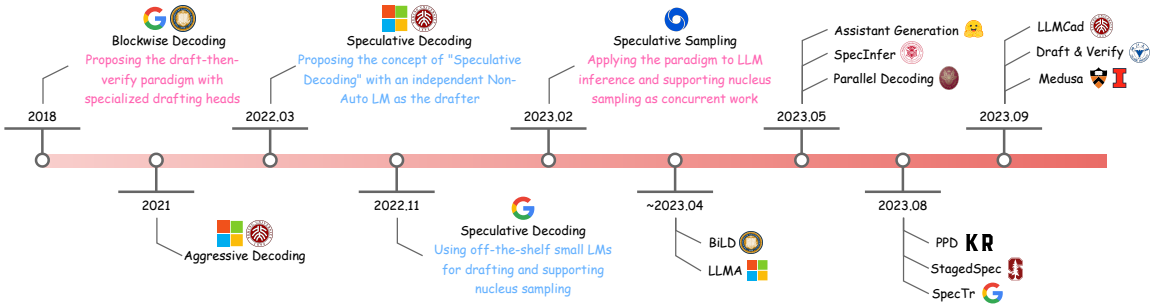

Figure 2: Timeline illustrating the evolution of Speculative Decoding. After 2022, Speculative Decoding was formally introduced as a general decoding paradigm to accelerate LLM inference and garnered widespread attention.

of pre-drafted tokens, substantially diminishing the need for frequent memory operations of LLM parameters, thereby improving inference efficiency.

While Speculative Decoding shows promise, it raises several critical questions that warrant further investigation. For instance, how to design an optimal drafter to strike a balance between speculation accuracy and drafting efficiency (Xia et al., 2022; Li et al., 2024). Additionally, it is essential to assess whether the verification criterion can maintain both generation parallelism and output quality (Miao et al., 2023; Cai et al., 2024). Furthermore, since existing Speculative Decoding methods are evaluated under disparate testing conditions, a unified benchmark is needed to facilitate realistic speedup expectations within the research community.

Amid the rapid expansion of research in Speculative Decoding, this work makes the first attempt to present a survey of this field, aiming to raise awareness among academics about the latest advancements. We provide a systematic categorization of current research and an in-depth analysis of relevant studies. Moreover, we introduce Spec-Bench, a comprehensive benchmark to assess Speculative Decoding methods in diverse application scenarios. Our contributions can be summarized as follows:

(1) ***First survey:*** To our knowledge, we are the first to present a comprehensive survey on Speculative Decoding;
(2) ***Formal definition:*** We furnish a formal definition and formulation of Speculative Decoding, laying the groundwork for future research.
(3) ***New taxonomy:*** We provide a systematic taxonomy for Speculative Decoding, offering an organized categorization of existing work.
(4) ***Spec-Bench:*** We introduce Spec-Bench, an extensive benchmark designed for assessing Speculative Decoding, enabling a comparative evaluation of leading methodologies.

We hope that this work can serve as an essential guide for newcomers and motivate future research.

2 Overview

This paper offers a comprehensive survey of Speculative Decoding. We commence by introducing the early stages of Speculative Decoding research (§3), illustrated by a timeline of its evolution (as shown in Figure 2). This is followed by a formal definition and formulation of Speculative Decoding (§4). Then, we delve into a detailed discussion of leading techniques, including the selection of drafter models (§5), verification strategies (§6), and alignment between the drafter and the target LLM (§7). Moreover, we introduce Spec-Bench, an extensive evaluation benchmark designed for assessing the acceleration effect of Speculative Decoding (§8).

3 Evolution of Speculative Decoding

This section discusses the motivation behind Speculative Decoding (§3.1) and then provides a detailed introduction to early attempts in this field (§3.2).

3.1 Motivation

The widespread adoption of LLMs has established autoregressive decoding as the *de facto* standard to LLM inference (Chowdhery et al., 2023; OpenAI, 2023; Jiang et al., 2024). However, autoregressive decoding is limited by its inference latency, which primarily stems from the memory-bound computation of LLMs (Patterson, 2004; Shazeer, 2019). Specifically, the main latency bottleneck of each decoding step is not due to computational operations but arises from the necessity to transfer all LLM parameters from High-Bandwidth Memory (HBM) to the on-chip cache of modern accelerators like GPUs. This process, which generates only one token per step, leads to the underutilization of these accelerators and results in inefficiencies.

3.2 Pioneering *Draft-then-Verify* Efforts

To mitigate the above issue, an intuitive way involves leveraging idle computational resources to enhance parallelism in LLM inference. To this end,

Algorithm 1 Autoregressive Decoding

Require: Language model $\mathcal{M}_q$, input sequence $x_1, \ldots, x_t$, and target sequence length T;
initialize $n \leftarrow t$
while $n < T$ **do**
  Set $q_{n+1} \leftarrow \mathcal{M}_q\left(x \mid x_{<n+1}\right)$
  Sample $x_{n+1} \sim q_{n+1}$
  $n \leftarrow n+1$
end while

Algorithm 2 Speculative Decoding

Require: Target language model $\mathcal{M}_q$, drafter model $\mathcal{M}_p$, input sequence $x_1, \ldots, x_t$, block size K, target sequence length T, drafting strategy DRAFT, verification criterion VERIFY, and correction strategy CORRECT;
initialize $n \leftarrow t$
while $n < T$ **do**
  // Drafting: obtain distributions from $\mathcal{M}_p$ efficiently
  Set $p_1, \ldots, p_K \leftarrow \textsc{Draft}\left(x_{\leq n}, \mathcal{M}_p\right)$
  // Drafting: sample K drafted tokens
  Sample $\widetilde{x}_i \sim p_i, i = 1, \ldots, K$
  // Verification: compute $K+1$ distributions in parallel
  Set $q_i \leftarrow \mathcal{M}_q\left(x \mid x_{\leq n}, \widetilde{x}_{<i}\right), i = 1, \ldots, K+1$
  // Verification: verify each drafted token
  for $i = 1 : K$ **do**
    if $\textsc{Verify}\left(\widetilde{x}_i, p_i, q_i\right)$ **then**
      Set $x_{n+i} \leftarrow \widetilde{x}_i$ and $n \leftarrow n+1$
    else
      $x_{n+i} \leftarrow \textsc{Correct}\left(p_i, q_i\right)$
      and Exit for loop.
    end if
  end for
  If all drafted tokens are accepted, sample next token $x_{n+1} \sim q_{K+1}$ and set $n \leftarrow n+1$.
end while

Stern et al. (2018) introduced Blockwise Decoding, an approach that incorporates extra feedforward neural (FFN) heads atop the Transformer decoder, enabling the simultaneous *drafting* of multiple tokens per step. These tokens are then *verified* by the original LLM *in parallel*, ensuring that the outputs align with those of the original LLM. As a pioneering work proposing the *Draft-then-Verify* paradigm, Blockwise Decoding effectively reduces the number of required LLM calls by increasing generation parallelism, thereby accelerating inference.

To further unleash the potential of this paradigm, Xia et al. (2022) introduced Speculative Decoding (SpecDec), which utilizes an independent drafter, notably a specialized Non-Autoregressive Transformer, to perform the drafting task both accurately and efficiently. Moreover, this method presented an innovative strategy that relaxes the rigid verification criterion, thereby increasing the acceptance rate of drafted tokens. Impressively, SpecDec achieves around 5× speedup over autoregressive decoding with comparable quality, underscoring the substantial potential of Speculative Decoding.

Following SpecDec, Leviathan et al. (2023) and Chen et al. (2023a) made concurrent contributions by proposing Speculative Sampling, expanding this paradigm to encompass the lossless acceleration of nucleus sampling. These methods employed smaller LMs from the same series (e.g., `T5-small`) to speed up the inference of their larger counterparts (e.g., `T5-XXL`). Unlike previous work, these *off-the-shelf* small LMs do not require additional training, enabling the rapid adoption of Speculative Decoding in LLM acceleration. This advancement has elevated Speculative Decoding to the forefront of LLM efficiency research, attracting widespread interest within the NLP community.

To sum up, these pioneering efforts in Speculative Decoding have gradually solidified the *Draft-then-Verify* paradigm, showcasing its promising potential in LLM acceleration. We provide a detailed categorization and discussion of these studies and subsequent research in the following sections.

4 Formulation and Definition

In this section, we first provide a concise overview of standard autoregressive decoding (§4.1). Then, we offer an in-depth exposition of Speculative Decoding (§4.2), which encompasses a formal definition, a comprehensive description of the methodology, and a detailed elaboration of the algorithm.

4.1 Autoregressive Decoding

Transformer-based LLMs typically make generations in an autoregressive manner. Given an input sequence $x_1, \ldots, x_t$, an autoregressive language model $\mathcal{M}_q$ generates the next token according to:

$$x_{t+1} \sim q_{n+1} = \mathcal{M}_q\left(x \mid x_{<t+1}\right), \qquad (1)$$

where q is the conditional probability distribution calculated by $\mathcal{M}_q$ and x_{t+1} denotes the next token sampled from q_{n+1}. We illustrate a detailed process in Algorithm 1.

As discussed in Section 3, while the standard autoregressive decoding offers desirable generation quality, it is bounded by memory bandwidth, resulting in low utilization of modern accelerators. In this process, each memory-bound LLM call (i.e., an LLM forward step) produces merely a single token for the entire sequence, making the whole generation inefficient and time-consuming.

4.2 Speculative Decoding

Following Xia et al. (2022), Leviathan et al. (2023), and Chen et al. (2023a), we here provide a formal

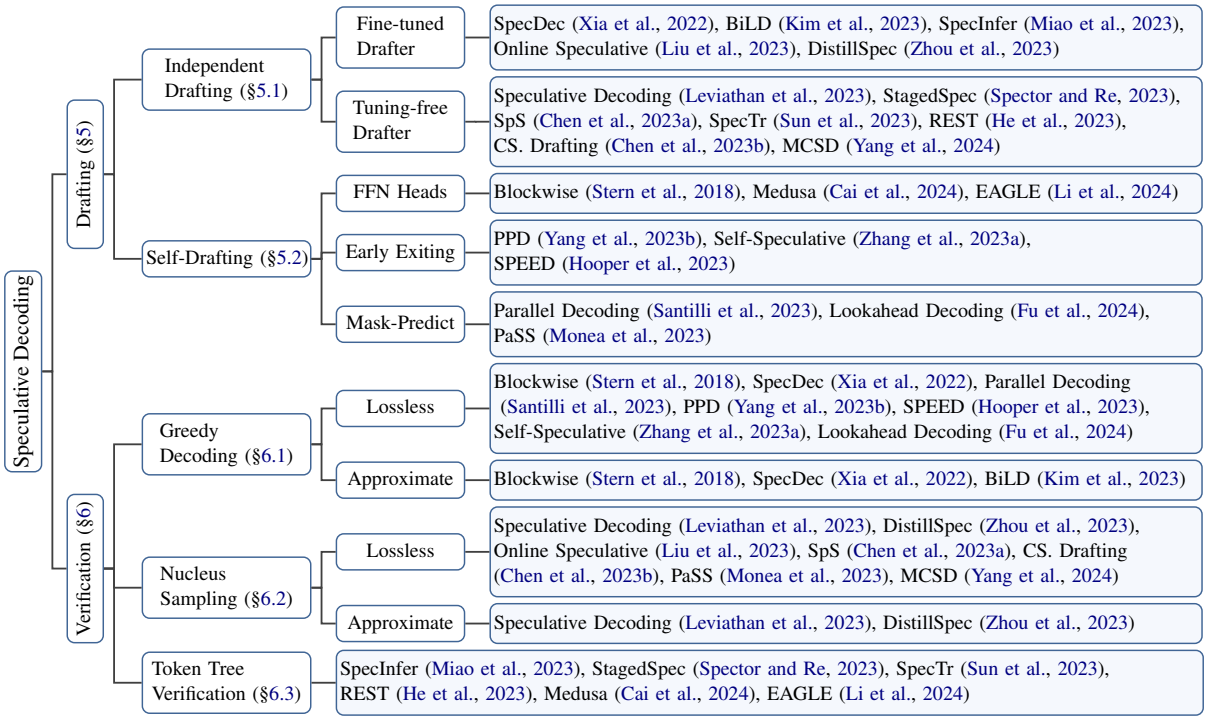

Figure 3: Taxonomy of Speculative Decoding.

definition of Speculative Decoding:

> Speculative Decoding is a *Draft-then-Verify* decoding paradigm in which, at each decoding step, it first *efficiently drafts* multiple future tokens and then *verifies* all these tokens *in parallel* using the target LLM to speed up inference.

We formulate a detailed Speculative Decoding process in Algorithm 2. Subsequently, we delve into the two fundamental substeps integral to this paradigm – *drafting* and *verification*:

Drafting At each decoding step, Speculative Decoding first efficiently drafts multiple future tokens, as a speculation of the target LLM's output. Formally, given an input sequence $x_1, \ldots, x_t$ and the target LLM $\mathcal{M}_q$, this paradigm employs an efficient drafter model $\mathcal{M}_p$ (e.g., a smaller LM) to decode the next K drafted tokens:

$$\begin{aligned} p_1, \ldots, p_K &= \textsc{Draft}\left(x_{\leq t}, \mathcal{M}_p\right), \\ \widetilde{x}_i &\sim p_i, \quad i = 1, \ldots, K, \end{aligned} \tag{2}$$

where $\textsc{Draft}(\cdot)$ denotes various drafting strategies that we will discuss in Section 5, p is the conditional probability distribution calculated by $\mathcal{M}_p$, and $\widetilde{x}_i$ denotes the drafted token sampled from p_i.

Verification Subsequently, these drafted tokens are verified by the target LLM $\mathcal{M}_q$ in parallel. Formally, given the input sequence $x_1, \ldots, x_t$ and the draft $\widetilde{x}_1, \ldots, \widetilde{x}_K$, Speculative Decoding utilizes $\mathcal{M}_q$ to compute $K+1$ probability distributions simultaneously:

$$q_i = \mathcal{M}_q\left(x \mid x_{\leq t}, \widetilde{x}_{<i}\right), i = 1, \ldots, K+1. \tag{3}$$

Then, each drafted token $\widetilde{x}_i$ is verified by a specific criterion $\textsc{Verify}\left(\widetilde{x}_i, p_i, q_i\right)$. Only those tokens that meet the criterion are selected as final outputs, ensuring quality consistent with the target LLM's standards. Otherwise, the first drafted token $\widetilde{x}_c$ that fails the verification will be corrected by the strategy $\textsc{Correct}\left(p_c, q_c\right)$. All drafted tokens after position c will be discarded, to guarantee the high quality of the final outputs. If all tokens pass verification, an additional token x_{t+K+1} will be sampled from q_{K+1} as Eq. (1).

The drafting and verification substeps will be iterated until the termination condition is met, i.e., the `[EOS]` token is decoded or the sentence reaches the maximal length.

Notably, the acceleration effect of Speculative Decoding primarily hinges on the acceptance rate of drafted tokens at each step. This rate is influenced by several factors, including the draft quality, verification criteria, and the behavior alignment between the drafter and the target LLM. Additionally, the intrinsic efficiency of the drafter itself also contributes to the overall speedup. In subsequent sections, we will delve into these pivotal components of Speculative Decoding, as depicted in Figure 3,

Methods	DRAFT $(x_{\leq t}, \mathcal{M}_p)$	Drafter Type
Parallel Drafting	$p_1, \ldots, p_K = \mathcal{M}_p\left(x \mid x_{\leq t}\right)$	FFN Heads (Stern et al., 2018; Cai et al., 2024), Non-Autoregressive LM (Xia et al., 2022), Mask-Predict (Santilli et al., 2023; Fu et al., 2024)
Autoregressive Drafting	$p_i = \mathcal{M}_p\left(x \mid x_{\leq t}, \widetilde{x}_{<i}\right), i = 1, \ldots, K$	Small LMs (Leviathan et al., 2023; Chen et al., 2023a), Early Exiting (Yang et al., 2023b), Layer Skipping (Zhang et al., 2023a)

Table 1: Summary of formulations for various drafting strategies in Speculative Decoding. We categorize these methods into two distinct groups based on their formulations: *parallel drafting* and *autoregressive drafting*.

to systematically categorize recent research trends within this promising paradigm.

5 Drafting

As a vital component of Speculative Decoding, the drafting process has a crucial impact on the speedup of the paradigm. The impact is determined by two key factors: the speculation accuracy of the drafter $\mathcal{M}_p$, measured by the average number of accepted tokens per step, and the drafting latency (Stern et al., 2018; Xia et al., 2022). How to trade off high speculation accuracy and low drafting latency presents a major challenge in this process. In this section, we classify various drafting strategies into two categories: independent drafting (§5.1) and self-drafting (§5.2), and summarize their formulations DRAFT $(x_{\leq t}, \mathcal{M}_p)$ in Table 1.

5.1 Independent Drafting

To strike a balance between speculation accuracy and efficiency, SpecDec (Xia et al., 2022) first proposed utilizing an independent model for drafting. Specifically, it employed a specialized Non-Autoregressive Transformer that drafts multiple tokens simultaneously per step. This model has a deep-shallow encoder-decoder architecture to run efficiently. Despite its strengths, SpecDec requires training a drafter model from scratch, which demands an increased computational budget.

Considering the available models in existing LLM series (e.g., OPT (Zhang et al., 2022) and LLaMA (Touvron et al., 2023a,b)), a more straightforward and efficient approach is directly employing a small LM from the same series as the drafter to accelerate the inference of its larger counterparts (Leviathan et al., 2023; Chen et al., 2023a; Spector and Re, 2023; Sun et al., 2023; Chen et al., 2023b). For instance, Leviathan et al. (2023) utilized `T5-small` as the drafter, to accelerate the inference of `T5-XXL`. These *off-the-shelf* small LMs do not require additional training or any modification on model architectures, facilitating the quick adoption of Speculative Decoding. Moreover, since models in the same series share tokenizers, pretraining corpora, and similar training processes, they inherently have an alignment in prediction behaviors.

5.2 Self-Drafting

While leveraging an external drafter model offers considerable advantages, this approach necessitates extra effort to either train or identify a drafter model that closely aligns with the target LLM. This challenge is intensified when no smaller counterparts of the LLM are available, e.g., LLaMA-7B (Touvron et al., 2023a,b). Furthermore, integrating two distinct models within a single system introduces additional computational complexity, particularly in distributed settings (Cai et al., 2024).

To address the above issues, numerous studies have suggested leveraging the target LLM itself for efficient drafting (Stern et al., 2018; Santilli et al., 2023; Hooper et al., 2023; Cai et al., 2024; Fu et al., 2024; Monea et al., 2023). Particularly, Blockwise Decoding (Stern et al., 2018) and Medusa (Cai et al., 2024) incorporated FFN heads atop the Transformer decoder, enabling the parallel token generation per step. Compared with external drafters, these lightweight heads reduce extra computational overhead and are friendly to distributed inference. Another line of research has explored the potential of *early exiting* and *layer skipping* within the target LLM for drafting (Yang et al., 2023b; Zhang et al., 2023a; Hooper et al., 2023). For instance, Yang et al. (2023b) introduced additional subprocesses that exit early during the current decoding step, thereby initiating the drafting of future tokens in advance. Similarly, Self-Speculative (Zhang et al., 2023a) proposed to adaptively skip several intermediate layers during inference to draft efficiently.

In contrast to prior work that focused on extending model architectures or altering the inference process, Santilli et al. (2023) introduced a simple drafting strategy that directly appends multiple `[PAD]` tokens to the end of the input prompt to enable parallel generation. However, this approach deviates from LLMs' autoregressive pretraining pattern, leading to suboptimal drafting quality. To tackle this, Fu et al. (2024) proposed transforming

Methods	VERIFY $(\widetilde{x}_i, p_i, q_i)$	CORRECT (p_c, q_c)	Representative Work
Greedy Decoding	$\widetilde{x}_i = \arg\max q_i$	$x_{t+c} \leftarrow \arg\max q_c$	Blockwise Decoding (Stern et al., 2018), SpecDec (Xia et al., 2022)
Nucleus Sampling	$r < \min\left(1, \frac{q_i(\widetilde{x}_i)}{p_i(\widetilde{x}_i)}\right), r \sim U[0,1]$	$x_{t+c} \sim \text{norm}(\max(0, q_c - p_c))$	Speculative Decoding (Leviathan et al., 2023), SpS (Chen et al., 2023a)

Table 2: Summary of formulations for various verification strategies in Speculative Decoding.

low-quality drafts into multiple n-grams to improve the speculation accuracy; Monea et al. (2023) introduced multiple learnable [LA] tokens and finetuned these token embeddings on a small training dataset to enhance the parallel decoding performance.

6 Verification

In each decoding step, the drafted tokens are *verified in parallel* to ensure the outputs align with the target LLM. This process also determines the number of tokens accepted per step, a vital factor impacting the speedup. This section summarizes various verification criteria VERIFY $(\widetilde{x}_i, p_i, q_i)$ (as shown in Table 2), encompassing those supporting greedy decoding (§6.1) and nucleus sampling (§6.2) in LLM inference. Besides, we introduce token tree verification (§6.3), an effective strategy to increase the token acceptance rate.

6.1 Greedy Decoding

Early attempts at Speculative Decoding focused on the verification criterion supporting greedy decoding, which guarantees that the outputs are exactly the same as the greedy decoding results of the target LLM (Stern et al., 2019; Sun et al., 2021; Xia et al., 2022). Formally, given the input sequence $x_1, \ldots, x_t$, the drafted tokens $\widetilde{x}_1, \ldots, \widetilde{x}_K$, and the computed probability distributions $p_1, \ldots, p_K$, $q_1, \ldots, q_K$ as obtained from Eq. (2) and (3), respectively, the verification criterion on the i_{th} drafted token is formulated as

$$\widetilde{x}_i = \arg\max q_i, \tag{4}$$

where $i = 1, \ldots, K$. The first position c that the drafted token $\widetilde{x}_c$ fails the verification denotes the *bifurcation* position. The output token at this position x_{t+c} will be adjusted by the correction strategy, which simply replaces the drafted token with the LLM's top-1 prediction:

$$x_{t+c} \leftarrow \arg\max q_c. \tag{5}$$

The verification criterion of greedy decoding is straightforward and clear. Thus, multiple subsequent studies have adopted this criterion to demonstrate the efficacy of their methodologies (Santilli et al., 2023; Yang et al., 2023b; Hooper et al., 2023; Zhang et al., 2023a; Fu et al., 2024). However, the strict matching requirement of this criterion often results in the rejection of high-quality drafted tokens, simply because they differ from the top-1 predictions of the target LLM, thereby constraining the speedup of the paradigm.

To tackle this problem, multiple studies have proposed various approximate verification criteria (Stern et al., 2018; Xia et al., 2022; Kim et al., 2023). Compared with the lossless criterion, these methods slightly relax the matching requirement to trust the drafts more, leading to higher acceptance of drafted tokens. For instance, SpecDec (Xia et al., 2022) only requires the drafted tokens to fall in top-k candidates of the target LLM; BiLD (Kim et al., 2023) proposed a rollback criterion that only rejects drafted tokens when the number of consecutive mismatch tokens exceeds a fixed threshold.

6.2 Nucleus Sampling

Following Stern et al. (2019), subsequent work extended Speculative Decoding to support nucleus sampling (Leviathan et al., 2023; Chen et al., 2023a), accelerating the target LLM's inference without changing its output distribution. Formally, given the initial sequence $x_1, \ldots, x_t$, the drafted tokens $\widetilde{x}_1, \ldots, \widetilde{x}_K$ and the computed distributions $p_1, \ldots, p_K$, $q_1, \ldots, q_K$, the verification criterion on the i_{th} drafted token is

$$r < \min\left(1, \frac{q_i(\widetilde{x}_i)}{p_i(\widetilde{x}_i)}\right), r \sim U[0,1], \tag{6}$$

where r denotes a random number drawn from a uniform distribution $U[0,1]$; $q_i(\widetilde{x}_i)$ and $p_i(\widetilde{x}_i)$ are the probability of $\widetilde{x}_i$ according to $\mathcal{M}_q$ and $\mathcal{M}_p$, respectively; and $i = 1, \ldots, K$. In other words, this criterion accepts the token $\widetilde{x}_i$ if $q_i(\widetilde{x}_i) \geq p_i(\widetilde{x}_i)$, and in case $q_i(\widetilde{x}_i) < p_i(\widetilde{x}_i)$ it rejects the token with probability $1 - \frac{q_i(\widetilde{x}_i)}{p_i(\widetilde{x}_i)}$. The correction strategy resamples the output token at the bifurcation position c from an adjusted distribution:

$$x_{t+c} \sim \text{norm}(\max(0, q_c - p_c)). \tag{7}$$

Leviathan et al. (2023) and Chen et al. (2023a) have theoretically proved that this criterion main-

	Methods	Drafting			Verification			Target LLM	Speedup (reported)
		Approach	Alignment	Tuning-free	Greedy	Nucleus	Token Tree		
Independent-D	SpecDec (Xia et al., 2022)	Non-Auto LM	Seq-KD	✗	✓	✗	✗	Transformer-base (65M)	3.9× ∼ 5.1×
	SpS (Chen et al., 2023a)	Small LM	-	✓	✓	✓	✗	Chinchilla (70B)	1.9× ∼ 2.5×
	SpecInfer (Miao et al., 2023)	Boost-tuned LMs	Col-BT	✗	✓	✓	✓	LLaMA (30B-65B)	2.0× ∼ 2.4×
	DistillSpec (Zhou et al., 2023)	Small LM	KD	✗	✓	✓	✗	T5-XL (3B)	-
	Online Speculative (Liu et al., 2023)	Small LM	Online-KD	✗	✓	✓	✗	Vicuna (7B)	-
	CS. Drafting (Chen et al., 2023b)	Cascaded LMs	-	✓	✓	✓	✗	FLAN-T5-xxl (11B)	-
	REST (He et al., 2023)	Context Retrieval	-	✓	✓	✓	✓	Vicuna (7B-13B)	1.6× ∼ 1.8×
Self-D	Blockwise Decoding (Stern et al., 2018)	FFN Heads	Seq-KD	✗	✓	✗	✗	Transformer-big (213M)	1.7× ∼ 3.0×
	Medusa (Cai et al., 2024)	FFN Heads	Seq-KD	✗	✓	✗	✓	Vicuna (7B-13B)	2.2× ∼ 2.3×
	PPD (Yang et al., 2023b)	Early Exiting	-	✗	✓	✗	✗	Vicuna (13B)	1.1× ∼ 1.5×
	Self-Speculative (Zhang et al., 2023a)	Layer Skipping	-	✓	✓	✗	✗	LLaMA-2 (13B-70B)	1.4× ∼ 1.7×
	Parallel Decoding (Santilli et al., 2023)	Mask-Predict	-	✓	✓	✗	✗	MBart50 (610M)	1.0× ∼ 1.1×
	Lookahead Decoding (Fu et al., 2024)	Mask-P & N-grams	-	✓	✓	✗	✗	LLaMA-2 (7B-70B)	1.5× ∼ 2.3×
	EAGLE (Li et al., 2024)	Auto-regression Head	KD	✗	✓	✓	✓	Vicuna (7B-33B)	2.9× ∼ 3.1×

Table 3: Summary of various Speculative Decoding methods. "*Independent-D*" and "*Self-D*" denote independent drafting and self-drafting, respectively. "*Greedy*", "*Nucleus*", and "*Token Tree*" denote whether the method supports greedy decoding, nucleus sampling, and token tree verification, respectively. We list the most representative target LLMs for each method and the speedups in the original paper (if reported), which is obtained with a batch size of 1.

tains identical output distributions to the target LLM. Thus, it has been widely adopted in subsequent research (Liu et al., 2023; Zhou et al., 2023; Monea et al., 2023; Chen et al., 2023b). In addition to the strict requirement, some work has also explored approximate strategies to improve the token acceptance rate (Leviathan et al., 2023; Zhou et al., 2023). For instance, Leviathan et al. (2023) proposed multiplying $p_i(\widetilde{x}_i)$ in Eq. (6) by a lenience parameter $l \in [0, 1]$ to slightly relax the criterion.

6.3 Token Tree Verification

Contrary to prior verification strategies that focused on a single draft sequence, SpecInfer (Miao et al., 2023) proposed *token tree verification*, an effective strategy enabling the target LLM to verify multiple draft sequences in parallel. As illustrated in Figure 4, this method first merges multiple candidate draft sequences into a *token tree* by sharing prefixes. It then utilizes a specially designed *tree attention mask* to facilitate the LLM verifying the whole structure in parallel. Recent research has explored various approaches to obtain these candidate draft sequences (Miao et al., 2023; Cai et al., 2024; He et al., 2023; Li et al., 2024). For instance, Miao et al. (2023) generated diverse draft sequences from different boost-tuned LMs; Cai et al. (2024) considered the top-k predictions from each FFN head to obtain multiple candidate sequences.

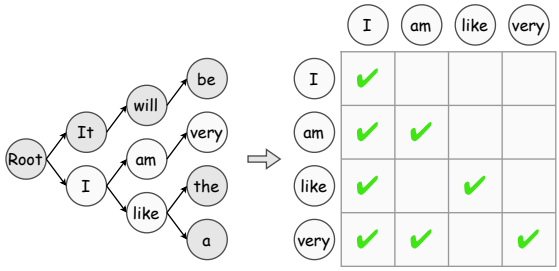

Figure 4: Illustration of the token tree sequences (*left*) and tree attention mask (*right*). For simplicity, we only visualize the attention mask of tokens in white colors.

7 Alignment

As illustrated in Section 5, the speedup of Speculative Decoding primarily depends on the speculation accuracy, which in turn is influenced by the behavior similarity between the drafter and the target LLM. To enhance this, existing research has explored various knowledge distillation (KD) strategies to align the drafter's outputs with those of the target LLM (Stern et al., 2018; Xia et al., 2022; Miao et al., 2023; Liu et al., 2023; Kim et al., 2023; Zhou et al., 2023). Particularly, Blockwise Decoding adopted sequence-level knowledge distillation (Seq-KD) (Kim and Rush, 2016) for alignment, which trained the drafter on the sentences generated by the target LLM. Miao et al. (2023) proposed a collective boost-tuning (Col-BT) strategy, applying Seq-KD to finetune multiple small LMs on the training data and utilizing their aggregated output as drafts to improve the speculation accuracy.

Though Seq-KD is effective, it ignores the probability distributions of the target LLM, leading to performance degradation when adopted for nucleus sampling. To rectify this, recent studies have explored other KD strategies for Speculative Decoding (Zhou et al., 2023; Liu et al., 2023). Notably, DistillSpec (Zhou et al., 2023) conducted a comprehensive comparison of different KD strategies on Speculative Decoding across various downstream tasks. Liu et al. (2023) proposed an online KD strategy that dynamically aligns the drafter with the target LLM on the fly using the query data.

We summarize the main features of existing Speculative Decoding methods in Table 3, including the drafter type or the drafting strategy, the

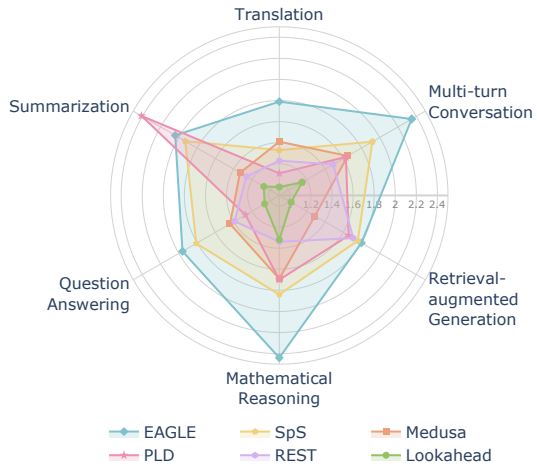

Figure 5: Speedup comparison of various Speculative Decoding methods on Spec-Bench with greedy settings ($T = 0$). Evaluations were conducted on Vicuna-7B with a batch size of 1. We present the mean speedup over 3 runs. The detailed results are shown in Appendix D.

alignment approach, supported verification strategies, and the reported speedup, etc.

8 Spec-Bench

With the rapid research progress in Speculative Decoding, there is an increasing demand for comparative analysis of leading methods. However, existing approaches are tested using disparate benchmarks, devices, and environments, making fair comparisons impractical. To address this gap, we introduce Spec-Bench – a comprehensive benchmark for Speculative Decoding covering diverse application scenarios. Based on Spec-Bench, we present a systematic comparison of open-source approaches under third-party testing conditions. Experiments were executed on *the same device and testing environment* to ensure a fair comparison.

8.1 Benchmark Construction

To assess Speculative Decoding methods across various scenarios, Spec-Bench encompasses six distinct subtasks: multi-turn conversation, translation, summarization, question answering, mathematical reasoning, and retrieval-augmented generation. We composed Spec-Bench by randomly selecting 80 instances from each of six widely used datasets, including MT-bench (Zheng et al., 2023), WMT14 DE-EN, CNN/Daily Mail (Nallapati et al., 2016), Natural Questions (Kwiatkowski et al., 2019), GSM8K (Cobbe et al., 2021), and DPR (Karpukhin et al., 2020). For details on Spec-Bench and the specific experimental setup, please refer to Appendix C.

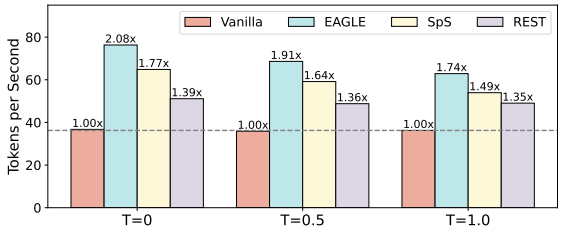

Figure 6: Speedup comparison of various methods on Spec-Bench at different temperatures. The speedup effect diminishes as the sampling temperature increases.

8.2 Comparative Evaluation

Our main evaluations were conducted on Vicuna-7B at FP16 precision, using a single consumer-grade 3090 GPU[4]. As depicted in Figure 5, under greedy settings, EAGLE (Li et al., 2024) achieves the largest speedup ratio (1.8×∼2.4×) over autoregressive decoding across most subtasks, especially the mathematical reasoning subtask (with ∼2.4× speedup). PLD (Saxena, 2023) excels in subtasks with high similarities between input and output, such as summarization (with ∼2.4× speedup). However, its performance diminishes in other subtasks like translation and question answering, with speedup ratios falling between 1.1×∼1.3×.

We also compare the speedups of Speculative Decoding methods at different sampling temperatures. As illustrated in Figure 6, EAGLE consistently outperforms other methods across various settings, achieving a speedup ratio ranging from 1.7× to 2.1×. Besides, it is observed that the acceleration effect of all methods decreases with an increase in sampling temperature. This is attributed to the increased computational complexity of the speculative sampling criterion at higher temperatures, as revealed in prior research (Joao Gante, 2023; Spector and Re, 2023).

9 Conclusion

This paper presents a comprehensive survey of Speculative Decoding, including a formal definition and formulation, a systematic categorization of existing methods, and an in-depth review of leading techniques. Moreover, we introduce Spec-Bench, an extensive evaluation benchmark for Speculative Decoding methods, and present a comparative evaluation of prominent methods. To our knowledge, this is the first survey dedicated to Speculative Decoding. Our aim for this paper is to clarify the current research landscape and provide insights into future research directions.

[4]For comparative analysis on a more powerful A100 GPU, please refer to Appendix E.

Limitations

This paper provides a thorough examination and categorization of current methodologies and emerging trends in Speculative Decoding. We have also conducted a comparative analysis of leading open-source methods to offer researchers deeper insights into the advantages and limitations of different models. Beyond Speculative Decoding, we acknowledge additional efficient NLP strategies such as vLLM (Kwon et al., 2023) and continuous batching (Yu et al., 2022). In the future, we intend to expand the discussion to encompass the integration of Speculative Decoding with these advanced techniques. Moreover, due to the absence of an available implementation of batched Speculative Decoding, our evaluations could not cover this aspect. We plan to undertake subsequent experiments to assess the speedup of Speculative Decoding methods across various batch sizes.

Ethics Statement

The datasets used in our experiment are publicly released and labeled through interaction with humans in English. In this process, user privacy is protected, and no personal information is contained in the dataset. The scientific artifacts that we used are available for research with permissive licenses. And the use of these artifacts in this paper is consistent with their intended use. Therefore, we believe that our research work meets the ethics of ACL.

References

Christopher Bryant, Zheng Yuan, Muhammad Reza Qorib, Hannan Cao, Hwee Tou Ng, and Ted Briscoe. 2023. Grammatical error correction: A survey of the state of the art. Comput. Linguistics, 49(3):643–701.

F. Warren Burton. 1985. Speculative computation, parallelism, and functional programming. IEEE Trans. Computers, 34(12):1190–1193.

Deng Cai, Yan Wang, Lemao Liu, and Shuming Shi. 2022. Recent advances in retrieval-augmented text generation. In SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022, pages 3417–3419. ACM.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. CoRR, abs/2401.10774.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023a. Accelerating large language model decoding with speculative sampling. CoRR, abs/2302.01318.

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

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. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2023. Palm: Scaling language modeling with pathways. J. Mach. Learn. Res., 24:240:1–240:113.

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. 2021. Training verifiers to solve math word problems. CoRR, abs/2110.14168.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. CoRR, abs/2307.08691.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. 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.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. 2024. Break the sequential dependency of llm inference using lookahead decoding.

Tao Ge, Heming Xia, Xin Sun, Si-Qing Chen, and Furu Wei. 2022. Lossless acceleration for seq2seq generation with aggressive decoding. CoRR, abs/2205.10350.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason D. Lee, and Di He. 2023. REST: retrieval-based speculative decoding. CoRR, abs/2311.08252.

John L. Hennessy and David A. Patterson. 2012. Computer Architecture - A Quantitative Approach, 5th Edition. Morgan Kaufmann.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Hasan Genc, Kurt Keutzer, Amir Gholami, and Yakun Sophia Shao. 2023. SPEED: speculative pipelined execution for efficient decoding. CoRR, abs/2310.12072.

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. 2023. Mistral 7b. CoRR, abs/2310.06825.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts.

Joao Gante. 2023. Assisted generation: a new direction toward low-latency text generation.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020, pages 6769–6781. Association for Computational Linguistics.

Sehoon Kim, Karttikeya Mangalam, Jitendra Malik, Michael W. Mahoney, Amir Gholami, and Kurt Keutzer. 2023. Speculative decoding with big little decoder. CoRR, abs/2302.07863.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. arXiv preprint arXiv:1606.07947.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. Transactions of the Association for Computational Linguistics, 7:452–466.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023, pages 611–626. ACM.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA, volume 202 of Proceedings of Machine Learning Research, pages 19274–19286. PMLR.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2023. Contrastive decoding: Open-ended text generation as optimization. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023, pages 12286–12312. Association for Computational Linguistics.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024. Eagle: Speculative sampling requires rethinking feature uncertainty.

Xiaoxuan Liu, Lanxiang Hu, Peter Bailis, Ion Stoica, Zhijie Deng, Alvin Cheung, and Hao Zhang. 2023. Online speculative decoding. CoRR, abs/2310.07177.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Rae Ying Yee Wong, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. 2023. Specinfer: Accelerating generative LLM serving with speculative inference and token tree verification. CoRR, abs/2305.09781.

Giovanni Monea, Armand Joulin, and Edouard Grave. 2023. Pass: Parallel speculative sampling. CoRR, abs/2311.13581.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016, pages 280–290. ACL.

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

David A. Patterson. 2004. Latency lags bandwith. Commun. ACM, 47(10):71–75.

Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodolà. 2023. Accelerating transformer inference for translation via parallel decoding. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023, pages 12336–12355. Association for Computational Linguistics.

Apoorv Saxena. 2023. Prompt lookup decoding.

Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need. CoRR, abs/1911.02150.

Benjamin Spector and Chris Re. 2023. Accelerating LLM inference with staged speculative decoding. CoRR, abs/2308.04623.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, volume 97 of Proceedings of Machine Learning Research, pages 5976–5985. PMLR.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. In Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, pages 10107–10116.

Qidong Su, Christina Giannoula, and Gennady Pekhimenko. 2023. The synergy of speculative decoding and batching in serving large language models. CoRR, abs/2310.18813.

Xin Sun, Tao Ge, Furu Wei, and Houfeng Wang. 2021. Instantaneous grammatical error correction with shallow aggressive decoding. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021, pages 5937–5947. Association for Computational Linguistics.

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix X. Yu. 2023. Spectr: Fast speculative decoding via optimal transport. CoRR, abs/2310.15141.

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. 2023a. Llama: Open and efficient foundation language models. CoRR, abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. CoRR, abs/2307.09288.

Yu Wang, Yuelin Wang, Kai Dang, Jie Liu, and Zhuo Liu. 2021. A comprehensive survey of grammatical error correction. ACM Trans. Intell. Syst. Technol., 12(5):65:1–65:51.

Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. 2022. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation.

Daliang Xu, Wangsong Yin, Xin Jin, Ying Zhang, Shiyun Wei, Mengwei Xu, and Xuanzhe Liu. 2023. Llmcad: Fast and scalable on-device large language model inference. CoRR, abs/2309.04255.

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

Sen Yang, Shujian Huang, Xinyu Dai, and Jiajun Chen. 2024. Multi-candidate speculative decoding. CoRR, abs/2401.06706.

Seongjun Yang, Gibbeum Lee, Jaewoong Cho, Dimitris S. Papailiopoulos, and Kangwook Lee. 2023b. Predictive pipelined decoding: A compute-latency trade-off for exact LLM decoding. CoRR, abs/2307.05908.

Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. 2022. Orca: A distributed serving system for transformer-based generative models. In 16th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2022, Carlsbad, CA, USA, July 11-13, 2022, pages 521–538. USENIX Association.

Hongyi Yuan, Keming Lu, Fei Huang, Zheng Yuan, and Chang Zhou. 2023. Speculative contrastive decoding. CoRR, abs/2311.08981.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. 2023a. Draft & verify: Lossless large language model acceleration via self-speculative decoding. CoRR, abs/2309.08168.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: open pre-trained transformer language models. CoRR, abs/2205.01068.

Zhihao Zhang, Alan Zhu, Lijie Yang, Yihua Xu, Lanting Li, Phitchaya Mangpo Phothilimthana, and Zhihao Jia. 2023b. Accelerating retrieval-augmented language model serving with speculation. In Submitted to The Twelfth International Conference on Learning Representations. Under review.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. In Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. 2023. Distillspec: Improving speculative decoding via knowledge distillation.

Appendix

A Applications

In addition to serving as a general paradigm, recent work has revealed that some variants of Speculative Decoding demonstrate extraordinary effectiveness in specific tasks. Furthermore, other research has applied this paradigm to address latency issues unique to certain application scenarios, achieving inference acceleration. Below, we will provide a detailed introduction to these promising works.

Recent studies have highlighted Speculative Decoding is particularly well suited for tasks where model inputs and outputs are highly similar (Sun et al., 2021; Ge et al., 2022; Yang et al., 2023a), such as Grammatical Error Correction (Wang et al., 2021; Bryant et al., 2023) and Retrieval-augmented Generation (Cai et al., 2022). These methods introduced a specialized form of Speculative Decoding, where the initial user input or the retrieved context is directly employed as drafts. For instance, SAD (Sun et al., 2021), an early attempt at Speculative Decoding on Grammatical Error Correction, utilized the input sentence with grammatical errors as a draft and leveraged the LLM to verify the whole sentence in parallel, achieving a $9\times\sim12\times$ speedup. Similarly, LLMA (Yang et al., 2023a) selected text spans from the reference as drafts, demonstrating a $2\times\sim3\times$ speedup across various practical application scenarios including Retrieval-augmented Generation, Cache-assisted Generation, and Multi-turn Conversations.

Beyond these works, RaLMSpec (Zhang et al., 2023b) adopted Speculative Decoding to accelerate retrieval-augmented language models (RaLMs). It pointed out that the main latency bottleneck of iterative RaLMs is the frequent retrieval from a vast knowledge base. To accelerate inference, this method proposed to maintain a local cache for speculative retrieval, achieving around $2\times$ speedup with identical model outputs. LLMCad (Xu et al., 2023) applied Speculative Decoding to on-device LLM inference. Concretely, it proposed to generate drafts with a smaller real-time LM that can be hosted in device memory, and only utilize the target LLM for parallel verification. This approach effectively reduces repetitive releasing and loading of model weights, achieving a $9.3\times$ speedup compared to existing inference engines.

B Challenges and Future Directions

How to trade off speculation accuracy and drafting efficiency? As discussed in Sections 5, scaling up the drafter can effectively enhance speculation accuracy, yet it largely reduces the drafting efficiency and even the overall speedup. Therefore, it is essential to strike a balance between speculation accuracy and drafting latency. Among existing strategies, behavior alignment is a promising approach to address this issue, as it improves speculation accuracy without increasing latency. However, despite recent advancements (Miao et al., 2023; Zhou et al., 2023; Liu et al., 2023), there is still considerable room for improvement to align the drafter with the target LLM. For example, given that the drafted tokens after the bifurcation position are all discarded, one potential direction could involve encouraging the drafter to prioritize the generation quality of early-position tokens. Beyond alignment, other factors such as the quality of drafting (Fu et al., 2024) and the determination of speculation length (Su et al., 2023) also influence speculation accuracy and merit further exploration.

How to apply Speculative Decoding in batched inference scenarios? Currently, only a few Speculative Decoding implementations have supported batched inference, such as EAGLE[5] and SpS[6]. However, batched inference is a crucial technique for efficiently managing user inputs in LLM real-time services. The primary challenges in batched Speculative Decoding lie in two aspects: (1) Each decoded sentence in Speculative Decoding varies in decoding steps due to different speculation accuracy. Thus, the inference latency of a batch depends on the slowest sample in the batch; (2) The extra computational complexity introduced by Speculative Decoding, especially in sampling settings, increases with larger batch sizes. How to maintain a promising speedup of Speculative Decoding in batched inference, and combine it with advanced techniques such as continuous batching (Yu et al., 2022), warrants further investigation.

How to integrate Speculative Decoding with other leading techniques? As a general decoding paradigm, Speculative Decoding has already demonstrated its potential in conjunction with other advanced techniques (Yang et al., 2023a; Zhang

[5] https://github.com/SafeAILab/EAGLE

[6] https://github.com/lucidrains/speculative-decoding

Subtask	Dataset	#Samples
Multi-turn Conversation	MT-bench	80
Retrieval-aug. Generation	Natural Questions	80
Summarization	CNN/Daily Mail	80
Translation	WMT14 DE-EN	80
Question Answering	Natural Questions	80
Mathematical Reasoning	GSM8K	80
Overall	-	480

Table 4: Detailed Composition of Spec-Bench. Spec-Bench includes 6 distinct subtasks to encompass diverse application scenarios.

et al., 2023b; Li et al., 2023). For instance, Yuan et al. (2023) combined Speculative Decoding with Contrastive Decoding (Li et al., 2023), which not only speeds up the inference but also substantially improves the generation quality. In addition to the acceleration of text-only LLMs, the application of Speculative Decoding in multimodal inference, such as image synthesis, text-to-speech synthesis, and video generation, is also an intriguing and valuable direction for future research. Another promising research direction is to integrate Speculative Decoding with other efficient methods such as vLLM (Kwon et al., 2023) and Flash-Attention (Dao et al., 2022; Dao, 2023), further boosting the inference efficiency of LLM services.

C Experimental Details

C.1 Details of Spec-Bench

To assess the acceleration performance of Speculative Decoding methods in various scenarios, we developed Spec-Bench, a comprehensive benchmark encompassing six distinct tasks. Spec-Bench integrates MT-bench (Zheng et al., 2023), a multi-turn conversation benchmark previously adopted in research (Cai et al., 2024; Li et al., 2024), to provide a basis for comparison with earlier studies. Additionally, it includes two input-guided tasks: summarization and retrieval-augmented generation (RAG), both of which exhibit a significant overlap between the input prompts and the target outputs. We selected CNN/Daily Mail (Nallapati et al., 2016) and Natural Questions (Kwiatkowski et al., 2019) as the dataset for these two tasks, respectively. Specifically, in the RAG subtask, the top-5 documents retrieved from DPR (Karpukhin et al., 2020) were concatenated with each question to construct the input prompt.

Moreover, Spec-Bench incorporates three further subtasks – translation, question answering, and mathematical reasoning – to provide a thorough evaluation of Speculative Decoding's speedup capabilities in diverse contexts. We utilized WMT14 DE-EN, Natural Questions, and GSM8K (Cobbe et al., 2021) as the primary datasets for these tasks, respectively. We randomly selected 80 instances from each subtask's test set for evaluation.

C.2 Implement Details

We have selected six representative Speculative Decoding methods for our comparative analysis on Spec-Bench. These methods are open-source and free of bugs. Specifically, **SpS** (Chen et al., 2023a) stands as the pioneering work in this field, utilizing a smaller LM from the same model series as the drafter to accelerate LLM inference. **Medusa** (Cai et al., 2024) and **EAGLE** (Li et al., 2024) integrate multiple lightweight heads into the target LLM to facilitate parallel drafting. **Lookahead** (Fu et al., 2024) introduces multiple special tokens to the end of the input prompt for parallel drafting and transforms the drafts into n-gram candidates. **PLD** (Saxena, 2023) is the code implementation[7] of LLMA (Yang et al., 2023a), which selects text spans from the input as drafts. **REST** (He et al., 2023) retrieves relevant drafts from text corpora based on the input prompt.

We conducted our experimental evaluations using the `Vicuna-v1.3` model series (Zheng et al., 2023). For SpS, we employed the Huggingface implementation[8] and utilized the `vicuna-68m-v1.3` model provided by Yang et al. (2024) as the drafter. We followed the default parameters of Lookahead[9] and PLD for our evaluations. The main experiments were conducted using Pytorch 2.0.1 with a single consumer-grade NVIDIA GeForce RTX 3090 GPU (24GB) of 12 CPU cores under CUDA 11.8. Further analysis was performed on a more powerful NVIDIA A100 GPU (80GB) of 64 CPU cores under CUDA 11.4.

D Details of Main Experimental Results

The detailed results of our main analysis are shown in Table 5, including the experimental settings of greedy decoding ($T = 0$) and nucleus sampling ($T = 1$). The findings indicate that EAGLE (Li

[7] https://github.com/apoorvumang/prompt-lookup-decoding

[8] https://huggingface.co/blog/assisted-generation

[9] https://github.com/hao-ai-lab/LookaheadDecoding

	Models	Multi-turn Conversation	Translation	Summarization	Question Answering	Mathematical Reasoning	Retrieval-aug. Generation	#tokens/s	Avg.
$T=0$	Autoregressive Decoding	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	36.74$_{\pm 0.31}$	1.00×
	Lookahead (Fu et al., 2024)	1.15×$_{\pm 0.01}$	0.98×$_{\pm 0.02}$	1.07×$_{\pm 0.02}$	1.06×$_{\pm 0.02}$	1.32×$_{\pm 0.02}$	1.03×$_{\pm 0.02}$	40.64$_{\pm 0.26}$	1.11×
	REST (He et al., 2023)	1.49×$_{\pm 0.02}$	1.23×$_{\pm 0.04}$	1.26×$_{\pm 0.03}$	1.39×$_{\pm 0.04}$	1.34×$_{\pm 0.03}$	1.71×$_{\pm 0.05}$	51.12$_{\pm 0.78}$	1.39×
	PLD (Saxena, 2023)	1.63×$_{\pm 0.02}$	1.11×$_{\pm 0.02}$	**2.41**×$_{\pm 0.04}$	1.27×$_{\pm 0.03}$	1.70×$_{\pm 0.03}$	1.66×$_{\pm 0.04}$	59.42$_{\pm 0.55}$	1.62×
	SpS (Leviathan et al., 2023)	1.92×$_{\pm 0.04}$	1.33×$_{\pm 0.02}$	1.93×$_{\pm 0.01}$	1.81×$_{\pm 0.04}$	1.84×$_{\pm 0.00}$	1.76×$_{\pm 0.01}$	64.85$_{\pm 0.70}$	1.77×
	Medusa (Cai et al., 2024)	1.65×$_{\pm 0.03}$	1.41×$_{\pm 0.02}$	1.33×$_{\pm 0.01}$	1.44×$_{\pm 0.03}$	1.69×$_{\pm 0.01}$	1.29×$_{\pm 0.02}$	54.30$_{\pm 0.34}$	1.48×
	EAGLE (Li et al., 2024)	**2.35**×$_{\pm 0.03}$	**1.79**×$_{\pm 0.03}$	2.04×$_{\pm 0.02}$	**1.96**×$_{\pm 0.03}$	**2.44**×$_{\pm 0.02}$	**1.80**×$_{\pm 0.03}$	**76.30**$_{\pm 0.36}$	**2.08**×
$T=1$	Autoregressive Decoding	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	36.24$_{\pm 0.43}$	1.00×
	REST (He et al., 2023)	1.43×$_{\pm 0.01}$	1.19×$_{\pm 0.02}$	1.24×$_{\pm 0.00}$	1.36×$_{\pm 0.02}$	1.34×$_{\pm 0.02}$	1.61×$_{\pm 0.02}$	49.04$_{\pm 0.30}$	1.35×
	SpS (Leviathan et al., 2023)	1.55×$_{\pm 0.01}$	1.20×$_{\pm 0.01}$	1.57×$_{\pm 0.01}$	1.54×$_{\pm 0.03}$	1.56×$_{\pm 0.03}$	1.52×$_{\pm 0.02}$	53.94$_{\pm 0.43}$	1.49×
	EAGLE (Li et al., 2024)	**1.79**×$_{\pm 0.02}$	**1.61**×$_{\pm 0.03}$	**1.74**×$_{\pm 0.03}$	**1.66**×$_{\pm 0.04}$	**1.95**×$_{\pm 0.06}$	**1.63**×$_{\pm 0.03}$	**62.88**$_{\pm 0.54}$	**1.74**×

Table 5: Speedup comparison of various Speculative Decoding methods on Spec-Bench. The results were obtained using `Vicuna-7B-v1.3` at FP16 precision. Evaluations were conducted on a single NVIDIA 3090 GPU with a batch size of 1. We report the mean speedup ratio over 3 different runs. We show the best results in **boldface**.

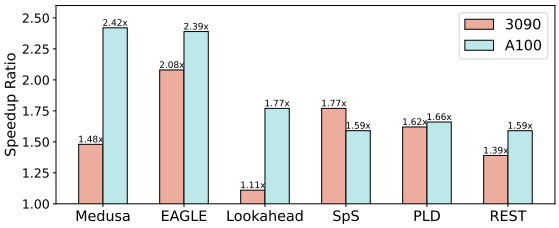

Figure 7: Speedup comparison of various methods on Spec-Bench with different computational devices.

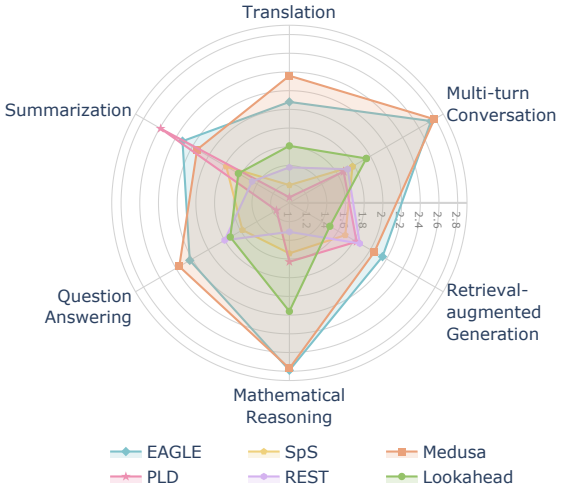

Figure 8: Speedup comparison of various Speculative Decoding methods on a single A100 GPU with greedy settings ($T = 0$). Evaluations were conducted on Spec-Bench using Vicuna-7B at FP16 precision.

et al., 2024) excels across various Spec-Bench subtasks, achieving an overall speedup ranging from 1.6× to 2.4×. PLD (Saxena, 2023) shows notable efficiency in scenarios where the input and output have a significant overlap. For instance, the speedup ratio of PLD increases from 1.27× in the question answering subtask to 1.66× in the retrieval-augmented generation subtask, highlighting its effectiveness when the input includes relevant documents. Notably, most methods achieve a suboptimal speedup on the translation subtask. We suspect that it is due to the potential lack of multilingual data in the pretraining corpora.

E Further Analysis on A100

This section presents a comprehensive analysis of leading Speculative Decoding methods on Spec-Bench, utilizing a single NVIDIA A100 GPU. The discussion delves into the influence of computational hardware, model scale, and computational precision on the performance of Speculative Decoding. All experiments were performed on *the same device* and *environment* to ensure fair comparison.

E.1 Computational Devices

We first discuss the impact of evolving computational devices on Speculative Decoding. As depicted in Figure 7, the acceleration effect of most Speculative Decoding methods is notably enhanced when employed on high-performance GPUs, such as NVIDIA A100s. This enhancement is primarily due to the increased availability of idle computational resources on more advanced computational devices, which Speculative Decoding can leverage to accelerate inference processes. Among the methods evaluated, Medusa (Cai et al., 2024) and Lookahead (Fu et al., 2024) demonstrate the most significant improvements. Specifically, the speedup ratio for Medusa escalates from 1.48× to 2.42×, and for Lookahead, it rises from 1.11× to 1.77×. This finding underscores that Speculative Decoding methods will benefit more from evolving computational hardware, such as H100 GPUs.

We illustrate the comparison of various Speculative Decoding methods evaluated with a single A100 GPU in Figure 8. The detailed experimental results are shown in Table 6. The results indicate

	Models	Multi-turn Conversation	Translation	Summarization	Question Answering	Mathematical Reasoning	Retrieval-aug. Generation	#tokens/s	Avg.
Vicuna-7B	Autoregressive Decoding	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	40.24$_{\pm 0.30}$	1.00×
	Lookahead (Fu et al., 2024)	1.95×$_{\pm 0.01}$	1.61×$_{\pm 0.05}$	1.63×$_{\pm 0.02}$	1.73×$_{\pm 0.04}$	2.16×$_{\pm 0.04}$	1.50×$_{\pm 0.00}$	71.20$_{\pm 1.30}$	1.77×
	REST (He et al., 2023)	1.72×$_{\pm 0.06}$	1.38×$_{\pm 0.05}$	1.46×$_{\pm 0.04}$	1.80×$_{\pm 0.04}$	1.31×$_{\pm 0.03}$	1.87×$_{\pm 0.06}$	63.81$_{\pm 1.00}$	1.59×
	PLD (Saxena, 2023)	1.67×$_{\pm 0.03}$	1.06×$_{\pm 0.03}$	**2.59**×$_{\pm 0.06}$	1.16×$_{\pm 0.03}$	1.63×$_{\pm 0.03}$	1.83×$_{\pm 0.02}$	66.61$_{\pm 1.15}$	1.66×
	SpS (Leviathan et al., 2023)	1.78×$_{\pm 0.03}$	1.19×$_{\pm 0.02}$	1.78×$_{\pm 0.03}$	1.58×$_{\pm 0.03}$	1.54×$_{\pm 0.02}$	1.69×$_{\pm 0.02}$	64.07$_{\pm 0.41}$	1.59×
	Medusa (Cai et al., 2024)	**2.79**×$_{\pm 0.07}$	**2.36**×$_{\pm 0.07}$	2.14×$_{\pm 0.04}$	**2.36**×$_{\pm 0.08}$	2.77×$_{\pm 0.08}$	2.05×$_{\pm 0.01}$	**97.27**$_{\pm 2.04}$	**2.42**×
	EAGLE (Li et al., 2024)	2.75×$_{\pm 0.05}$	2.08×$_{\pm 0.05}$	2.32×$_{\pm 0.05}$	2.23×$_{\pm 0.03}$	**2.79**×$_{\pm 0.04}$	**2.15**×$_{\pm 0.01}$	96.23$_{\pm 1.15}$	2.39×
Vicuna-13B	Autoregressive Decoding	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	31.38$_{\pm 0.22}$	1.00×
	Lookahead (Fu et al., 2024)	1.57×$_{\pm 0.01}$	1.34×$_{\pm 0.01}$	1.39×$_{\pm 0.00}$	1.40×$_{\pm 0.01}$	1.82×$_{\pm 0.02}$	1.32×$_{\pm 0.01}$	46.42$_{\pm 0.12}$	1.48×
	REST (He et al., 2023)	1.68×$_{\pm 0.01}$	1.31×$_{\pm 0.05}$	1.51×$_{\pm 0.01}$	1.67×$_{\pm 0.02}$	1.29×$_{\pm 0.00}$	1.96×$_{\pm 0.01}$	48.89$_{\pm 0.26}$	1.56×
	PLD (Saxena, 2023)	1.53×$_{\pm 0.02}$	1.08×$_{\pm 0.01}$	2.25×$_{\pm 0.00}$	1.09×$_{\pm 0.02}$	1.65×$_{\pm 0.03}$	1.72×$_{\pm 0.00}$	48.42$_{\pm 0.17}$	1.54×
	SpS (Leviathan et al., 2023)	1.73×$_{\pm 0.02}$	1.25×$_{\pm 0.02}$	1.76×$_{\pm 0.00}$	1.53×$_{\pm 0.01}$	1.68×$_{\pm 0.00}$	1.73×$_{\pm 0.00}$	50.48$_{\pm 0.28}$	1.61×
	Medusa (Cai et al., 2024)	2.39×$_{\pm 0.02}$	2.12×$_{\pm 0.02}$	1.92×$_{\pm 0.00}$	2.07×$_{\pm 0.02}$	2.49×$_{\pm 0.02}$	1.88×$_{\pm 0.00}$	67.64$_{\pm 0.07}$	2.16×
	EAGLE (Li et al., 2024)	**2.88**×$_{\pm 0.05}$	**2.24**×$_{\pm 0.04}$	**2.52**×$_{\pm 0.03}$	**2.24**×$_{\pm 0.04}$	**2.90**×$_{\pm 0.03}$	**2.34**×$_{\pm 0.01}$	**79.35**$_{\pm 1.18}$	**2.53**×
Vicuna-33B	Autoregressive Decoding	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	1.00×$_{\pm 0.00}$	16.34$_{\pm 0.01}$	1.00×
	Lookahead (Fu et al., 2024)	1.46×$_{\pm 0.00}$	1.21×$_{\pm 0.00}$	1.32×$_{\pm 0.00}$	1.29×$_{\pm 0.00}$	1.71×$_{\pm 0.00}$	1.28×$_{\pm 0.00}$	22.58$_{\pm 0.08}$	1.38×
	REST (He et al., 2023)	1.71×$_{\pm 0.01}$	1.39×$_{\pm 0.00}$	1.57×$_{\pm 0.00}$	1.69×$_{\pm 0.01}$	1.34×$_{\pm 0.01}$	1.89×$_{\pm 0.00}$	25.98$_{\pm 0.07}$	1.59×
	PLD (Saxena, 2023)	1.45×$_{\pm 0.00}$	1.06×$_{\pm 0.00}$	1.98×$_{\pm 0.00}$	1.07×$_{\pm 0.00}$	1.54×$_{\pm 0.00}$	1.43×$_{\pm 0.00}$	23.07$_{\pm 0.01}$	1.41×
	SpS (Leviathan et al., 2023)	1.79×$_{\pm 0.00}$	1.31×$_{\pm 0.00}$	1.80×$_{\pm 0.00}$	1.57×$_{\pm 0.00}$	1.73×$_{\pm 0.00}$	1.69×$_{\pm 0.00}$	26.89$_{\pm 0.03}$	1.65×
	Medusa (Cai et al., 2024)	2.22×$_{\pm 0.00}$	1.95×$_{\pm 0.00}$	1.85×$_{\pm 0.00}$	1.87×$_{\pm 0.01}$	2.32×$_{\pm 0.01}$	1.84×$_{\pm 0.00}$	32.92$_{\pm 0.06}$	2.01×
	EAGLE (Li et al., 2024)	**2.81**×$_{\pm 0.00}$	**2.14**×$_{\pm 0.00}$	**2.53**×$_{\pm 0.00}$	**2.19**×$_{\pm 0.00}$	**3.01**×$_{\pm 0.00}$	**2.31**×$_{\pm 0.00}$	**40.91**$_{\pm 0.03}$	**2.50**×

Table 6: Speedup comparison of Speculative Decoding methods across various model scales on Spec-Bench. The results were obtained using `Vicuna-v1.3` at FP16 precision with greedy settings ($T = 0$). Evaluations were conducted on a single NVIDIA A100 GPU with a batch size of 1. We report the mean speedup over 3 different runs.

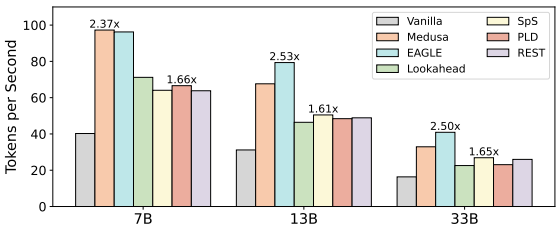

Figure 9: Speedup comparison of various methods on Spec-Bench at different model scales.

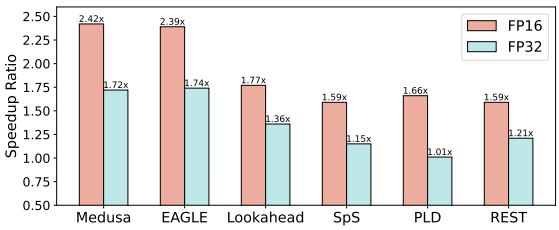

Figure 10: Speedup comparison of various methods on Spec-Bench with different computational precision.

that Medusa (Cai et al., 2024) and EAGLE (Li et al., 2024) excel in this experimental setting, achieving an overall speedup of 2.4×. These two methods perform particularly well on the multi-turn conversation and mathematical reasoning subtasks, with a ∼2.8× speedup.

E.2 Model Scale

We present the speedup comparison of Speculative Decoding methods across various model scales in Figure 9. The detailed experimental results are shown in Table 6. Among all the evaluated methods, EAGLE (Li et al., 2024) maintains a high speedup ratio over autoregressive decoding across all model scales, achieving a speedup ratio ranging from 2.4× to 2.5×. While Medusa (Cai et al., 2024) demonstrates superior acceleration performance on Vicuna-7B, its speedup ratio degrades from 2.4× to 2.0× as the model scale increases.

E.3 Computational Precision

It is noteworthy that most Speculative Decoding approaches are predominantly evaluated using FP16 precision (Fu et al., 2024; Cai et al., 2024; Li et al., 2024; He et al., 2023). However, it is critical to underscore that the outputs generated by Speculative Decoding in FP16 precision may not consistently align with those derived from autoregressive decoding. This divergence stems from the accumulation of floating-point errors inherent in FP16 computations, which can result in discrepancies between the outputs of the two decoding methods, particularly in the context of longer sequences. In FP32 precision, the outputs of Speculative Decoding are guaranteed to be exactly the same as autoregressive decoding.

We compare the speedup performance of Speculative Decoding methods with FP16/FP32 precision in Figure 10. The experimental results reveal a noticeable reduction in speedup for all methods under FP32 precision. Specifically, PLD (Saxena, 2023) achieves merely 1.01× speedup in FP32 precision,

and the acceleration effect of EAGLE (Li et al., 2024) also diminishes, with its speedup falling from 2.39× to 1.74×. To furnish the research community with a comprehensive understanding of the acceleration impact, we advocate for future studies to report speedup metrics across both precision settings.