SPECTRA: Faster Large Language Model Inference with Optimized Internal and External Speculation

Anonymous ACL submission

Abstract

Inference with modern Large Language Models (LLMs) is both computationally expensive and time-consuming. Speculative decoding has emerged as a promising solution, but existing approaches face key limitations: training-based methods require a draft model that is challenging to obtain and lacks generalizability, while training-free methods offer limited speedup gains. In this work, we present SPECTRA, a novel framework for accelerating LLM inference without the need for additional training or modification to the original LLM. SPECTRA introduces two new techniques for efficiently utilizing internal and external speculation, each outperforming corresponding state-of-the-art (SOTA) methods independently. When combined, these techniques achieve up to a 4.08x speedup across various benchmarks and LLM architectures, significantly surpassing existing training-free approaches. The implementation of SPECTRA is publicly available.

1 Introduction

Generating long sequences with low latency using Large Language Models (LLMs) is a critical requirement. Current LLMs rely on autoregressive decoding (Touvron et al., 2023; Bai et al., 2023; Jiang et al., 2023; OpenAI et al., 2024), which suffers from inefficiency because it generates text one token at a time. This results in generation time scaling linearly with the sequence length and underutilizes the parallel processing capabilities of modern GPUs. A widely studied approach to mitigate this issue is speculative decoding (Chen et al., 2023; Leviathan et al., 2023), which follows a *guess-and-verify* paradigm. In this approach, a smaller LLM (draft model) (Chen et al., 2023; Leviathan et al., 2023; Miao et al., 2024; Sun et al., 2023b; Zhou et al., 2024; Cai et al., 2024) or the original LLM trained in a specialized manner (self-speculative decoding) (Elhoushi et al., 2024; Liu et al., 2024a; Yang et al., 2024; Zhang et al., 2024a; Li et al., 2024b) predicts multiple tokens in advance. The original LLM then verifies these predictions in parallel, improving efficiency. However, these approaches require additional training, which demands substantial computational resources and may degrade the original model's capabilities.

Another line of research focuses on speculating subsequent tokens without requiring additional training. This approach eliminates the need for training new models or modifying the original LLM, making it practical for off-the-shelf deployment. Some methods leverage specialized mechanisms to generate speculative tokens directly from the LLM's predictions (Fu et al., 2024; Ou et al., 2024), while others rely on external information sources to derive these tokens (Yang et al., 2023; He et al., 2024; Li et al., 2024a). However, the speedup gain in these approaches remains limited due to the quality of the speculative guesses.

We introduce SPECTRA (Figure 1a), a speculative decoding method that improves generation speed without requiring any training or modifications to the original LLM. SPECTRA consists of two main components: a core module (SPECTRA-CORE, Figure 1c), which integrates seamlessly into LLMs in a plug-and-play manner, and an optional retrieval module (SPECTRA-RETRIEVAL, Figure 1e) that further enhances performance. The core module SPECTRA-CORE improves speculative decoding by leveraging the token distribution predicted by the LLM to generate high-quality guesses. Specifically, it employs two multi-level N-gram dictionaries that enable bi-directional search for dynamic-length guesses, balancing both quality and quantity. Additionally, SPECTRA optimizes a candidate pool to continuously update the N-gram dictionaries, ensuring broad token coverage. All updates to these resources, along with guess verification, are performed efficiently in a single forward pass. The retrieval module, SPECTRA-RETRIEVAL,

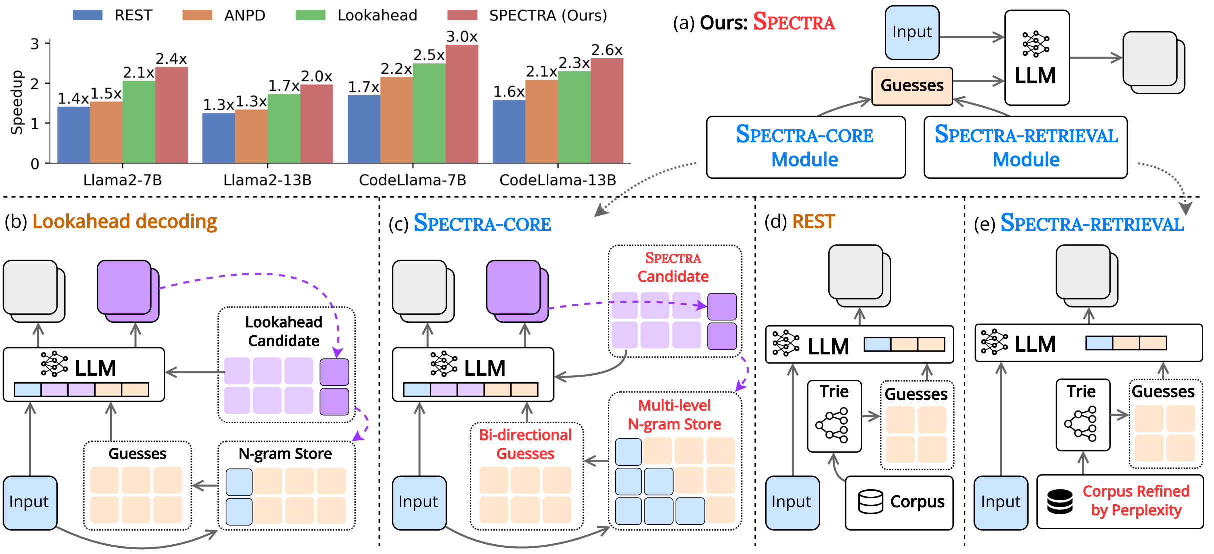

Figure 1: Overview of Spectra and comparison with other non-training SOTA approaches. (a) Overview of SPECTRA. (b) Overview of Lookahead Decoding (Fu et al., 2024). (c) Overview of the SPECTRA-CORE module, which utilizes the knowledge inside LLM for obtaining guesses. (d) Overview of REST (He et al., 2024). (e) Overview of the SPECTRA-RETRIEVAL module, which is designed to be integrated efficiently with SPECTRA-CORE to boost the speedup. The results in the bar chart are measured on HumanEval.

can be integrated to further enhance speedup. Existing approaches that rely on external sources for generating guesses (He et al., 2024) struggle to integrate with other speculative decoding methods, as the search time outweighs the speedup gains. SPECTRA-RETRIEVAL addresses this issue by reducing the search space, selecting only high-quality content from the corpus based on perplexity scores computed by the target LLM. This optimization enables seamless integration with SPECTRA-CORE, maximizing efficiency.

Empirical results on six tasks—including multi-turn conversation, code generation, and mathematical reasoning—across three LLM families (Llama 2 (Touvron et al., 2023), Llama 3 (Dubey et al., 2024), and CodeLlama (Rozière et al., 2024)) with model sizes ranging from 7B to 70B demonstrate that SPECTRA outperforms other non-training speculative decoding methods, achieving speedups of up to **4x**. We publicly release the code and data. The key contributions of this paper are as follows:

- We introduce SPECTRA, which improves speculative decoding by effectively leveraging the LLM's predicted token distribution. SPECTRA is a plug-and-play solution that requires no modifications to the LLM (Section 3.1).
- SPECTRA's retrieval module refines external corpora using perplexity scores computed by the target LLM, providing a general framework that enables speculative decoding approaches relying on external information to be seamlessly integrated with other speculative decoding techniques (Section 3.2).
- Extensive experiments across diverse tasks, LLM architectures, GPU types, and settings demonstrate the efficiency of SPECTRA, outperforming other non-training speculative decoding approaches (Section 5). SPECTRA also integrates with acceleration tools such as FlashAttention and pipeline parallelism (Section 5.2). The code and data are available.

2 Preliminaries

2.1 Autoregressive Decoding in LLMs

Given an input sequence $\mathbf{x} = (x_1, x_2, \ldots, x_s)$ of length s, and a slice of length m as $\mathbf{x}_{1:m} = (x_1, x_2, \ldots, x_m)$, the output of an LLM represents a probability distribution over the next token. The probability of generating the s-th token, conditioned on all preceding tokens, is given by $P_M(x_s \mid \mathbf{x}_{1:s-1})$. The next token x_s is sampled from this distribution using methods such as greedy, top-k, or top-p sampling (see (Kool et al., 2020; Holtzman et al., 2020)). For greedy sampling, the next token is selected as $x_s = \text{argmax}\ P_M(x_s \mid \mathbf{x}_{1:s-1})$. Consequently, the LLM generates an output sequence $(y_1, y_2, \ldots, y_m)$ of length m autoregressively, where each token y_i is computed as

$$y_i = \text{argmax } P_M(y_i \mid y_{1:i-1}, \mathbf{x}).$$

2.2 Speculative Decoding

Speculative decoding follows a *guess-and-verify* approach, where multiple candidate future tokens are speculated and subsequently verified in a single decoding step. With tree attention (Miao et al., 2024), multiple drafts can be verified simultaneously. Let G denote the number of guesses, and define the set of guesses as $\tilde{Y} = \{\tilde{y}^{(1)}, \tilde{y}^{(2)}, \ldots, \tilde{y}^{(G)}\}$, where each guess sequence has length K. The j-th token of the i-th guess is denoted as $\tilde{y}_j^{(i)}$.

In the case of speculative decoding with greedy sampling, given the prompt $\mathbf{x}$, a drafting method generates the draft sequences $\tilde{Y}$. Using these drafts, the LLM computes the true tokens $(y'_1, y'_2, \ldots, y'_K)$ in parallel. These tokens are then verified, and h is defined as the highest number of correctly guessed tokens across all guesses. Consequently, $h + 1$ tokens are generated in a single forward step. Algorithm 2 outlines speculative decoding with greedy sampling, and additional details are provided in Appendix A.

3 SPECTRA DECODING

SPECTRA consists of two modules (SPECTRA-CORE and SPECTRA-RETRIEVAL) that can function independently or together. The core module (SPECTRA-CORE) improves speedup by leveraging the LLM's predicted token distribution to generate high-quality guesses and integrates into LLMs in a plug-and-play manner. The retrieval module (SPECTRA-RETRIEVAL) derives guesses from a refined external information source and is designed to integrate with SPECTRA-CORE to further enhance performance.

3.1 SPECTRA-CORE

SPECTRA-CORE maintains an N-gram storage, which is used to obtain guesses, and a candidate pool, which is used to augment new N-grams in storage. The candidate pool $\mathcal{C}$ contains W sequences, $\{c^{(0)}, c^{(1)}, \ldots, c^{(W-1)}\}$, with each sequence consisting of N tokens. Let $c_j^{(i)}$ represent the j-th token in the i-th sequence. The N-gram storage includes two dictionaries: the forward dictionary $\mathcal{S}_{\text{fwd}}$ and the backward dictionary $\mathcal{S}_{\text{bwd}}$. At each time step, guesses $\mathcal{G}$ are obtained through a bidirectional search using $\mathcal{S}_{\text{fwd}}$ and $\mathcal{S}_{\text{bwd}}$. A single inference pass to the LLM retrieves all necessary distributions, which are used to generate new candidate tokens for $\mathcal{C}$ and verify the guesses $\mathcal{G}$. The dictionaries $\mathcal{S}_{\text{fwd}}$ and $\mathcal{S}_{\text{bwd}}$ are updated with N-grams from the candidate pool. The details of the SPECTRA-CORE decoding process are described in Algorithm 1.

Algorithm 1 SPECTRA-CORE Decoding Process

Require: Sequence $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, model P_M, max N-gram size N, candidate pool size W, max guesses G, max number of new tokens m. Refine threshold τ

Initialize N-gram Forward-dictionary $\mathcal{S}_{fwd} \leftarrow \emptyset$
Initialize N-gram Backward-dictionary $\mathcal{S}_{bwd} \leftarrow \emptyset$
Random $c_j^{(i)}, \forall j \in [0, N-1], \forall i \in [0, W-1]$
$t \leftarrow n + 1$
while $t \leq n + m$ **do**
  {Obtain the guesses}
  $\mathcal{G} \leftarrow \mathcal{S}_{fwd}[\mathbf{x}_{t-1}]$
  $u = \emptyset$
  for $j = 0$ to $N-1$ **do**
    for $k = N-1$ to 1 **do**
      $u_j \leftarrow \mathcal{S}_{bwd}[\mathbf{x}_{t+j-k:t-1} \oplus u_{0:j-1}]$
      break if found value for u_j
    end for
  end for
  $\mathcal{G}$.append(u)
  $\mathcal{G} = \mathcal{G} \oplus \mathcal{G}_{retrieve}$ ▷ Retrieval Integration (Optional)
  $\mathcal{G} \leftarrow \mathcal{G}_{0:G-1}$ ▷ Ensure the max guesses is G
  {Foward in LLM}
  Obtain necessary distributions of P_M in parallel.
  {Verification}
  {**Greedy** verify (Alg. 3) or **Sampling** verify (Alg. 4)}
  $hits \leftarrow$ VerificationFunction$(\mathbf{x}, P_M, \mathcal{G})$
  $\mathbf{x} \leftarrow \mathbf{x} \oplus hits$
  $t \leftarrow t + \text{size}(hits)$
  {Predict Candidates}
  for $i = 0$ to $W-1$ **do**
    $r \sim \text{Uniform}[0, 1]$
    $P_c(c_{N-1}^{(i)}) \leftarrow P_M(c_{N-1}^{(i)} \mid c_{:N-2}^{(i)}, \mathbf{x})$
    if $r > \tau$ **then**
      $c_{N-1}^{(i)} \leftarrow \underset{c \notin \mathcal{S}_{fwd}}{\text{argmax}}\, P_c(c_{N-1}^{(i)})$
    else
      $c_{N-1}^{(i)} \leftarrow \text{argmax}\, P_c(c_{N-1}^{(i)})$
    end if
  end for
  {Update N-gram dictionaries}
  for $i = 0$ to $W-1$ **do**
    for $j = 0$ to $N-2$ **do**
      $\mathcal{S}_{fwd}[c_j^{(i)}]$.append$(c_{j+1:}^{(i)})$
      $\mathcal{S}_{bwd}[c_{0:j}^{(i)}] \leftarrow c_{j+1}^{(i)}$
    end for
  end for
  {Update Candidates}
  $c_j^{(i)} \leftarrow c_{j+1}^{(i)}, \forall j \in [0, N-2], \forall i$
end while
Output: $\mathbf{x}_{n+1:n+m} = (y_1, y_2, \ldots, y_m)$

Bi-directional Search for Guesses. At each step, SPECTRA generates G guess sequences $\mathcal{G} = \{\tilde{y}^{(0)}, \tilde{y}^{(1)}, \ldots, \tilde{y}^{(G)}\}$. Unlike previous work (Fu et al., 2024), which enforces uniform guess lengths, SPECTRA supports variable-length guesses, im-

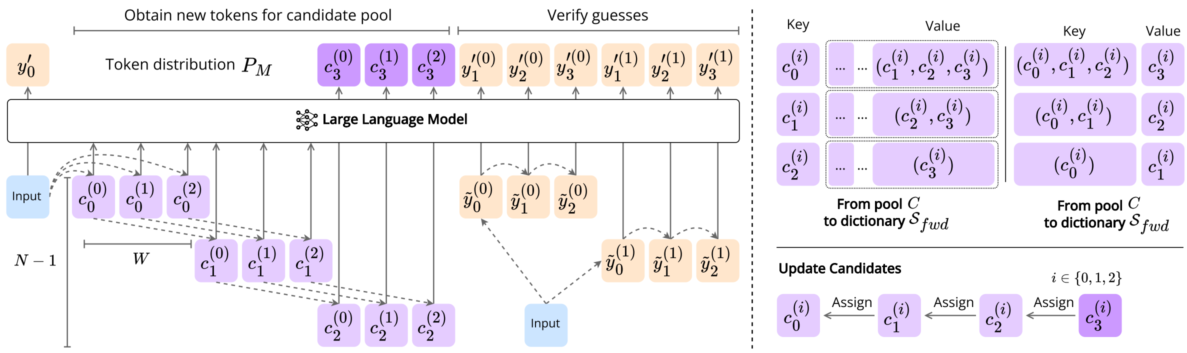

Figure 2: Details of how SPECTRA handles internal knowledge. The dashed arrow indicates interactions between the tokens, which are realized by the attention mask in the LLM.

proving both flexibility and efficiency. The forward dictionary $\mathcal{S}_{fwd}$ maps a token to a *list of sequences*, while the backward dictionary $\mathcal{S}_{bwd}$ maps a sequence to a single token. At time step t, the set of guesses is obtained through a bidirectional search (Alg. 1, lines 7–17). This search operates in two directions: (1) the forward direction, which prioritizes the quantity of guesses, and (2) the backward direction, which prioritizes the quality of guesses. In the forward direction, the last generated token x_{t-1} is used to search $\mathcal{S}_{fwd}$ for guess sequences (Alg. 1, line 7). In the backward direction, a high-quality guess is constructed by iteratively predicting one token at a time using $\mathcal{S}_{bwd}$, repeating the process until a desired sequence length N is reached (Alg. 1, lines 8–14).

Verification. The verification step ensures the output distribution is preserved by validating the guesses (Alg. 1, lines 22–24). For greedy sampling, the process is detailed in Appendix H (Alg. 3). In general speculative decoding, verification involves sending draft tokens to the LLM to obtain outputs and progressively checking if the LLM-generated token matches the draft token. Following prior work (Fu et al., 2024), we verify multiple guesses in parallel, accepting the guess with the largest number of correctly predicted tokens. For advanced sampling methods, we adopt sampling verification from (Miao et al., 2024; Fu et al., 2024), whose correctness has been proven. Details on sample verification are provided in Appendix H (Algorithm 4), and its performance and speedups are verified in Appendix F.

Predict & Verify in One Forward Pass. All distributions required for predicting candidates and verifying guesses are obtained in a single forward pass to the LLM, leveraging parallel processing (Figure 2). This is achieved using a specially designed attention mask that specifies the allowed interactions between tokens. For instance, the token $c_2^{(1)}$ attends only to $c_1^{(1)}$, $c_0^{(1)}$, and the input.

Predict Tokens for Candidate Pool. We predict the next candidate tokens $c_{N-1}^{(i)}$ for the candidate pool using the distribution obtained from the forward pass (Alg. 1, lines 26–34). A straightforward approach is to select tokens with the highest probability in the token distribution. However, we observe that when searching for guesses in the forward dictionary $\mathcal{S}_{fwd}$, it is crucial for the search token to exist in the dictionary; otherwise, no guesses can be retrieved. To address this, we introduce a randomness-based mechanism to increase the coverage of $\mathcal{S}_{fwd}$. Specifically, we probabilistically encourage the selection of unseen tokens in $\mathcal{S}_{fwd}$ using a hyperparameter $\tau \in [0, 1]$. Let r be a random draw from $[0, 1]$. If $r > \tau$, we select tokens with the highest probability that are not in $\mathcal{S}_{fwd}$; otherwise, we choose tokens with the highest probability regardless of their presence in $\mathcal{S}_{fwd}$. Although $c_{N-1}^{(i)}$ does not immediately affect the coverage of $\mathcal{S}_{fwd}$, it contributes to coverage expansion in subsequent time steps through our candidate updating mechanism. At the end of each time step, all candidate sequences are shifted left by one token: $c_j^{(i)} \leftarrow c_{j+1}^{(i)}$, leaving $c_{N-1}^{(i)}$ empty and ready for prediction in the next time step (Alg. 1, line 43).

Update N-gram Dictionaries. At the end of each time step, candidate tokens from the pool $\mathcal{C}$ are used to update the N-gram dictionaries $\mathcal{S}_{fwd}$ and $\mathcal{S}_{bwd}$. While previous work (Fu et al., 2024) only adds the full N-gram $(c_0^{(i)}, c_1^{(i)}, \ldots, c_N^{(i)})$, we observe that subsequences within N-grams often appear later in the generation process. By including these subsequences in the N-gram storage, we improve both the quality of guesses and the coverage of the dictionaries. Specifically, we add subsequences to $\mathcal{S}_{fwd}$ using the first token as the key, and update $\mathcal{S}_{bwd}$ by mapping the preceding part of the sequence to the last token (Alg. 1, lines 35–41).

3.2 SPECTRA-RETRIEVAL

SPECTRA-RETRIEVAL leverages an external knowledge source to generate guesses. This involves processing a text corpus and indexing it into a structure that supports fast prefix search, such as a trie. At each time step, the last generated tokens are used as input to this structure to retrieve guesses for speculative decoding. However, we observe that using random texts from the corpus without selection can limit the speedup gain. To address this, we propose a method to identify and select high-quality, relevant texts from the corpus tailored to the specific LLM. This improves the speedup gain and enables seamless integration with other speculative decoding approaches, including SPECTRA-CORE.

Corpus Refinement by Perplexity. Given a text sequence $u = (u_0, u_1, \ldots, u_t)$, perplexity quantifies the average uncertainty of the model when predicting the next token, conditioned on the preceding tokens. The perplexity is calculated as $\mathrm{PPL}(u) = \exp\left\{-\frac{1}{t}\sum_{i=1}^{t}\log P_M(u_i \mid u_{<i})\right\}$

A lower perplexity indicates that the model assigns higher probabilities to the sequence, suggesting that the sequence is well-aligned with the model's predictions and can produce high-quality guesses for speculative decoding. To optimize the retrieval process, we select texts with the lowest perplexity from the corpus to form a smaller, high-quality subset, which is then used to construct the Trie structure for generating guesses.

Integration with SPECTRA-CORE. Our experiments (Section 5.2, Table 2) demonstrate that naively integrating guesses from external sources (e.g., REST (He et al., 2024)) into other speculative methods (e.g., Lookahead (Fu et al., 2024)) can lead to a noticeable drop in speedup. This occurs because the forward pass in the LLM can only handle a limited number of guesses, and exceeding this limit increases memory usage and slows down generation. With a limited guess budget, guesses from external sources can only account for a fraction of the total guesses, causing the search time in the indexing structure (e.g., a trie) to outweigh the speedup gain. To address this, it is crucial to limit the size of the external knowledge while maintaining the quality of the guesses. By refining the corpus using perplexity, SPECTRA-RETRIEVAL seamlessly integrates with SPECTRA-CORE, further boosting the speedup gain. Specifically, we integrate SPECTRA-RETRIEVAL into SPECTRA-CORE by including its guesses ($\mathcal{G}_{retrieve}$) in the set of SPECTRA-CORE's guesses during the guess generation step (Alg. 1, line 16).

4 Experiments

Models. We evaluate LLaMA-2-Chat 7B, 13B, 70B (Touvron et al., 2023), CodeLlama 7B, 13B (Rozière et al., 2024), and LLaMA-3-Instruct 8B, 70B (Dubey et al., 2024).

Tasks. We conduct comprehensive evaluations on various generation tasks. MT-Bench (Zheng et al., 2023) for multi-turn conversation; GSM8K (Cobbe et al., 2021) for mathematical reasoning; HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021) and ClassEval (Du et al., 2023) for code generation.

Metrics. SPECTRA does not modify the original LLM and the acceptance conditions, making it a lossless acceleration method. Therefore, the generation quality remains the same as the original LLM. We only evaluate the acceleration performance using the following metrics.

- **Speedup Ratio:** The speedup ratio relative to autoregressive decoding.
- **Compression ratio τ:** The ratio of the total number of autoregressive steps to the number of Spectra decoding steps needed to produce the same sequence length.

Baselines. We use standard autoregressive decoding as the baseline (speed-up ratio = 1.00x). We further compare SPECTRA with leading non-training speculative decoding approaches, namely Adaptive N-gram (Ou et al., 2024), REST (He et al., 2024), and Lookahead (Fu et al., 2024). For details regarding implementation settings of both SPECTRA and these baselines, please refer to Appendix B.

5 Results

5.1 Main Results

Overall Performance. The top portion of Table 1 presents speedup ratios under greedy decoding. SPECTRA consistently achieves the highest acceleration, with speedups up to 4.08× for Llama-3-8B on MBPP. For 7B models, SPECTRA often exceeds 3× acceleration, highlighting the effectiveness of multi-token compression. For 13B models, speedups are slightly lower (1.6×–3×). Overall, the model architecture and dataset characteristics significantly influence the speedup gains of speculative decoding methods. While some approaches excel in specific scenarios—such as tasks with repetitive patterns or predictable token distributions (e.g., repeated variable names or class definitions), they often struggle in diverse or open-ended contexts. In contrast, SPECTRA demonstrates robustness across a wide range of models and datasets, consistently achieving the highest speedup ratios.

Compression Ratio. Table 1 also reports each method's compression rate, a measure agnostic to specific hardware configurations. Across every dataset and LLM tested, SPECTRA delivers the highest average compression ratio. Each of SPECTRA's draft-and-verify iterations typically yields 2.1–4.8 tokens, substantially outpacing alternative approaches and nearly doubling the acceptance length achieved by REST.

Acceleration in Sampling Decoding. The lower section of Table 1 reports the performance of SPECTRA under sampling-based decoding with a temperature of 1.0. The results highlight how SPECTRA continues to accelerate generation relative to baselines, offering roughly 1.15–2.77× speedups over standard autoregressive decoding. These gains are more modest than in greedy decoding, reflecting the lower acceptance rate under the sampling-based verification phase, which is consistent with earlier findings (Fu et al., 2024; Leviathan et al., 2023).

5.2 Analysis

Ablation Study. We performed a detailed component-wise analysis to evaluate the contribution of each module to the overall performance (Table 2). On LLaMA2-7B-chat, removing components impacts GSM8K speedups differently. Using only SPECTRA-CORE, excluding multi-level n-grams reduces the speedup from 2.04× to 1.95×, omitting backward dictionary guesses lowers it to 1.94×, and removing forward dictionary guesses drops it further to 1.50×. For SPECTRA-RETRIEVAL, skipping perplexity-based filtering decreases the speedup from 1.18× to 1.16×. The full SPECTRA framework achieves a 2.14× speedup on GSM8K, underscoring the importance of integrating all components to maximize acceptance rates and performance. A similar trend holds for the MTBench dataset. Additionally, we compared SPECTRA with a naive combination of Lookahead and REST, where guess sequences from REST are appended to Lookahead. This approach performs significantly worse than SPECTRA, underscoring that a straightforward merger of two techniques is inadequate without our carefully optimized integration strategy and components.

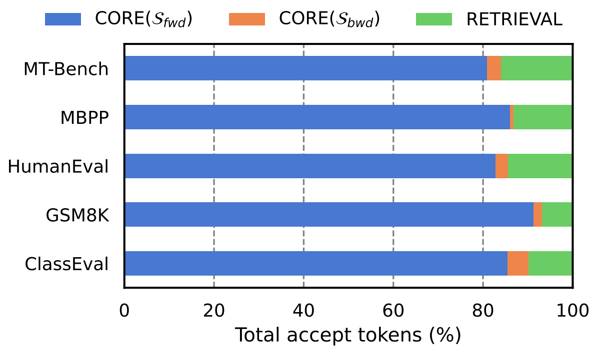

Figure 3: Acceptance rates of Llama2-7B-chat for different guess sources (from SPECTRA-CORE forward dictionary, backward dictionary, SPECTRA-RETRIEVAL). The acceptance rate is the fraction of guessed tokens that pass verification.

Priority for Source of Guesses. Since verifying too many candidate tokens at once can strain GPU resources and reduce speedups (Fu et al., 2024; Li et al., 2024b), SPECTRA limits the total number of guesses processed in each step (Appendix B). In order to assess the individual contributions of our two modules—SPECTRA-CORE and SPECTRA-RETRIEVAL—we temporarily remove the limit on the number of guess sequences in the verification branch and monitor the acceptance rates (Figure 3). We find that guesses generated by the SPECTRA-CORE module (via both forward and backward dictionaries) are accepted at a higher rate than those obtained from the external knowledge source via the SPECTRA-RETRIEVAL module. As a result, SPECTRA gives priority to internal guesses from SPECTRA-CORE over external guesses from SPECTRA-RETRIEVAL, as in Algorithm 1.

FlashAttention. Figure 4 shows that enabling FlashAttention consistently boosts the speedup of all methods, albeit to varying degrees. Notably, we observe an additional 0.24× speedup gain for SPECTRA on both GSM8K and MTBench. This is because FlashAttention better exploits the paral-

Model	Method	Classeval Speedup	Classeval τ	GSM8K Speedup	GSM8K τ	Humaneval Speedup	Humaneval τ	MBPP Speedup	MBPP τ	MTBench Speedup	MTBench τ	AVG Speedup
Greedy (temperature=0)												
CL-13B	ANPD	1.94	2.52	2.81	3.72	2.08	2.50	2.71	3.58	2.61	3.41	2.43
	Lookahead	2.25	3.61	2.80	4.24	2.30	3.16	2.91	4.44	2.59	4.04	2.57
	REST	1.28	2.14	0.93	1.54	1.58	2.31	0.85	1.40	0.94	1.53	1.12
	SPECTRA (Ours)	**2.38**	**4.06**	**2.91**	**4.65**	**2.63**	**3.95**	**3.29**	**4.46**	**2.65**	**4.40**	**2.77**
CL-7B	ANPD	2.30	2.68	3.21	3.75	2.16	2.47	3.16	3.78	3.35	3.83	2.84
	Lookahead	2.59	3.66	2.99	3.83	2.50	3.05	2.90	3.67	3.23	4.27	2.84
	REST	1.45	2.22	0.91	1.39	1.70	2.34	0.96	1.45	1.02	1.44	1.21
	SPECTRA (Ours)	**2.70**	**4.10**	**3.33**	**4.59**	**2.96**	**3.90**	**3.56**	**4.45**	**3.70**	**4.52**	**3.25**
L2-13B	ANPD	1.36	1.78	1.47	1.72	1.34	1.61	1.12	1.32	1.17	1.37	1.29
	Lookahead	1.81	2.76	1.46	1.87	1.73	2.32	1.38	1.69	1.51	2.04	1.58
	REST	1.22	2.01	0.94	1.46	1.25	1.94	0.95	1.44	1.14	1.90	1.10
	SPECTRA (Ours)	**2.00**	**3.24**	**1.83**	**2.62**	**1.96**	**2.91**	**1.63**	**2.24**	**1.75**	**2.60**	**1.83**
L2-70B	ANPD	1.82	1.90	1.63	1.61	1.86	1.87	1.17	1.20	1.34	1.30	1.56
	Lookahead	2.65	2.87	1.86	2.02	2.57	2.67	1.49	1.54	1.94	2.00	2.10
	SPECTRA (Ours)	**3.10**	**3.40**	**2.52**	**2.69**	**3.22**	**3.37**	**1.86**	**1.93**	**2.43**	**2.51**	**2.62**
L2-7B	ANPD	1.62	1.95	1.52	1.68	1.54	1.67	1.19	1.33	1.30	1.37	1.43
	Lookahead	2.19	2.94	1.66	1.93	2.06	2.42	1.46	1.69	1.73	2.05	1.82
	REST	1.36	2.12	1.01	1.47	1.41	2.04	1.01	1.46	1.25	1.90	1.21
	SPECTRA (Ours)	**2.40**	**3.43**	**2.11**	**2.64**	**2.40**	**3.05**	**1.77**	**2.16**	**2.02**	**2.59**	**2.14**
L3-70B	ANPD	1.54	1.67	1.50	1.47	1.83	1.88	1.46	1.41	1.23	1.23	1.51
	Lookahead	2.40	2.62	1.54	1.58	2.56	2.70	1.43	1.45	1.76	1.86	1.94
	SPECTRA (Ours)	**2.67**	**2.91**	**2.10**	**2.14**	**2.84**	**3.02**	**1.94**	**1.94**	**2.06**	**2.13**	**2.32**
L3-8B	ANPD	2.11	2.49	3.86	4.57	1.83	2.09	3.36	3.58	1.14	1.23	2.46
	Lookahead	2.59	3.44	3.71	4.61	2.49	2.89	3.79	4.65	1.53	1.85	2.82
	SPECTRA (Ours)	**2.83**	**3.49**	**3.89**	**4.77**	**2.57**	**3.02**	**4.08**	**4.76**	**1.69**	**2.10**	**3.01**
Sampling (temperature=1.0)												
CL-13B	ANPD	1.15	1.46	1.07	1.31	1.05	1.30	1.00	1.24	2.31	2.89	1.31
	Lookahead	1.38	2.00	1.08	1.43	1.29	1.75	1.02	1.34	2.33	3.48	1.42
	REST	1.14	1.87	0.82	1.35	1.27	1.96	0.84	1.39	0.93	1.50	1.00
	SPECTRA (Ours)	**1.68**	**2.22**	**1.20**	**1.75**	**1.65**	**2.12**	**1.15**	**1.70**	**2.37**	**3.80**	**1.61**
CL-7B	ANPD	1.29	1.50	1.16	1.30	1.10	1.32	1.12	1.27	2.77	3.05	1.49
	Lookahead	1.54	2.03	1.19	1.41	1.43	1.81	1.19	1.43	2.72	3.50	1.61
	REST	1.23	1.86	0.88	1.33	1.33	1.98	0.91	1.40	0.97	1.44	1.06
	SPECTRA (Ours)	**1.81**	**2.25**	**1.35**	**1.73**	**1.68**	**2.12**	**1.33**	**1.72**	**2.78**	**3.94**	**1.79**
L2-13B	ANPD	1.20	1.52	1.24	1.46	1.17	1.40	1.03	1.22	1.17	1.35	1.16
	Lookahead	1.52	2.22	1.32	1.69	1.48	2.00	1.18	1.48	1.49	2.01	1.40
	REST	1.18	1.96	0.93	1.45	1.19	1.88	0.92	1.44	1.12	1.88	1.07
	SPECTRA (Ours)	**1.70**	**2.75**	**1.55**	**2.23**	**1.69**	**2.59**	**1.34**	**1.89**	**1.74**	**2.57**	**1.60**
L2-7B	ANPD	1.31	1.51	1.34	1.48	1.28	1.46	1.10	1.22	1.25	1.36	1.26
	Lookahead	1.78	2.30	1.51	1.76	1.72	2.09	1.25	1.49	1.68	2.02	1.59
	REST	1.26	2.03	0.99	1.46	1.27	1.93	0.96	1.41	1.21	1.88	1.14
	SPECTRA (Ours)	**1.97**	**2.83**	**1.78**	**2.28**	**2.04**	**2.75**	**1.47**	**1.84**	**1.97**	**2.54**	**1.85**
L3-8B	ANPD	1.25	1.37	1.97	2.18	1.43	1.65	1.89	2.07	1.15	1.21	1.54
	Lookahead	1.48	1.78	2.07	2.41	1.79	2.21	1.99	2.40	1.57	1.81	1.78
	SPECTRA (Ours)	**1.94**	**2.84**	**2.27**	**2.78**	**1.92**	**2.51**	**2.19**	**2.78**	**1.70**	**2.05**	**2.01**

Table 1: Overall performance of speculative decoding methods across multiple tasks. “CL-xB” denotes CodeLlama with xB parameters, “L2-xB” denotes LLaMA-2-Chat of size xB, and “L3-xB” denotes LLaMA-3-Instruct of size xB. We report the speedup ratio (vs. autoregressive) and the compression ratio τ.

lel structure of speculative decoding by reducing attention overheads, especially when verifying multiple guessed tokens in parallel. Although smaller gains are also seen for other methods, SPECTRA benefits the most, as it presents the longest verification branches and thus stands to profit significantly from more efficient attention implementations.

Other Analysis. Detailed throughputs from Table 1 are provided in Appendix D. Evaluations of

Method	GSM8K Speedup	GSM8K τ	MTBench Speedup	MTBench τ
REST	1.01	1.47	1.25	1.90
Lookahead	1.66	1.93	1.73	2.05
Lookahead + REST	1.08	1.47	1.27	1.90
SPECTRA's ablation				
CORE Module	2.04	2.50	1.92	2.35
- *w/o Forward Dict*	1.50	1.68	1.20	1.37
- *w/o Backward Dict*	1.94	2.21	1.74	2.12
- *w/o Sub-Ngram*	1.95	2.34	1.75	2.18
RETRIEVAL Module	1.18	1.31	1.24	1.50
- *w/o PPL refine*	1.16	1.29	1.20	1.45
SPECTRA (ours)	**2.14**	**2.64**	**2.02**	**2.59**

Table 2: Ablation study of SPECTRA's components (greedy decoding, LLaMA2-7B-Chat).

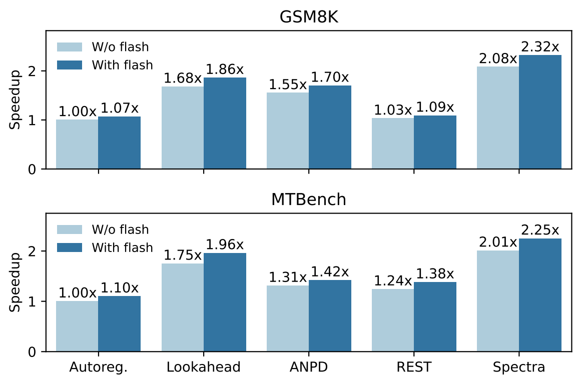

Figure 4: Effect of FlashAttention on speculative decoding speed: Measured speedups on GSM8K and MTBench (LLama2-7B-Chat, greedy decoding). "No Flash" uses standard attention; "With Flash" uses FlashAttention for faster parallel verification.

SPECTRA on different GPU types and quantization settings are described in Appendix C, while its performance in distributed settings with multiple GPUs is discussed in Appendix E.

6 Related Works

Large language models (LLMs) are increasingly deployed in a range of applications, motivating ongoing research into more efficient inference (Liu et al., 2025). Common strategies include quantizing model weights into lower-precision formats (Liu et al., 2024b; Lin et al., 2024; Zhao et al., 2024; Park et al., 2024), pruning redundant parameters (Ma et al., 2023; Xia et al., 2023; Sun et al., 2023a; Le et al., 2025), and employing knowledge distillation (Gu et al., 2024; Friha et al., 2024; Zhang et al., 2024b). These techniques help reduce the computational load per forward pass, thereby lowering generation latency. However, they often introduce some degradation in model performance, forcing practitioners to balance quality with efficiency.

A growing line of work explores *speculative decoding* as a strategy for accelerating generation while maintaining the output distribution (Chen et al., 2023; Leviathan et al., 2023). Some speculative decoding approaches train a smaller LLM (draft model) (Chen et al., 2023; Leviathan et al., 2023; Miao et al., 2024; Sun et al., 2023b; Zhou et al., 2024; Cai et al., 2024), or train the original LLM itself in a special manner (self-speculative) (Elhoushi et al., 2024; Liu et al., 2024a; Yang et al., 2024; Zhang et al., 2024a; Li et al., 2024b) to guess several subsequent tokens and then verify them parallelly using the original LLM. As these approaches require training, they pose limitations, such as requiring heavy computational resources and losing the original model capabilities.

To avoid additional training, alternative speculative decoding methods leverage external resources or structural properties of language generation. Retrieval-based methods sidestep draft model training by using a datastore indexed with observed prefixes to retrieve guess sequences (Yang et al., 2023; He et al., 2024; Li et al., 2024a). Other approaches, such as Jacobi-like parallel decoding (Santilli et al., 2023) and lookahead decoding (Fu et al., 2024), mitigate left-to-right dependencies by generating and validating multiple candidate tokens in parallel. These training-free techniques achieve comparable speedups to learned methods without requiring model optimization, making them ideal for scenarios with computational constraints.

7 Conclusions

In this work, we have introduced SPECTRA, a new, training-free framework for accelerating large language model inference by harnessing both internal and external speculation. By integrating our plug-and-play SPECTRA-CORE module—which leverages multi-level N-gram storage and bidirectional search—with the refined SPECTRA-RETRIEVAL module that selects high-quality external cues via perplexity-based filtering, our approach achieves substantial speedups (up to 4.08×) across diverse tasks and model architectures while preserving the original model's output quality. By offering a lossless speedup, SPECTRA provides a practical, high-impact solution for accelerating inference in LLMs.

8 Limitations

(1) Cost of Building External Datastores. While SPECTRA-CORE—our internal-knowledge module—relies solely on sequences observed during generation and thus requires no extra external data, SPECTRA-RETRIEVAL depends on constructing and indexing a sizeable external datastore from potentially large corpora. This process can be time-consuming and memory-intensive, particularly in domains where data updates frequently or storage is constrained. Although this additional investment can yield substantial speedups by boosting token acceptance rates, it may not be universally feasible or cost-effective.

(2) Limited Evaluation Scope. Our experiments center primarily on English-language benchmarks in conversational and coding tasks using LLaMA-based models. Although SPECTRA can, in principle, be applied to other models or languages, additional factors such as domain-specific tokenization or specialized textual structures may affect the acceptance rate and overall speedup. Future work is needed to assess the generality of SPECTRA across diverse linguistic settings (e.g., low-resource languages or specialized technical documents) and for a wider range of model families (beyond LLaMA-based architectures) to confirm and refine its applicability.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen Technical Report.

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. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org. Place: Vienna, Austria.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating Large Language Model Decoding with Speculative Sampling.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing Chat Language Models by Scaling High-quality Instructional Conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3029–3051, Singapore. Association for Computational Linguistics.

Xueying Du, Mingwei Liu, Kaixin Wang, Hanlin Wang, Junwei Liu, Yixuan Chen, Jiayi Feng, Chaofeng Sha, Xin Peng, and Yiling Lou. 2023. Classeval: A manually-crafted benchmark for evaluating llms on class-level code generation. *arXiv preprint arXiv:2308.01861*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed Aly, Beidi Chen, and Carole-Jean Wu. 2024. LayerSkip: Enabling Early Exit Inference and Self-Speculative Decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12622–12642, Bangkok, Thailand. Association for Computational Linguistics.

Othmane Friha, Mohamed Amine Ferrag, Burak Kantarci, Burak Cakmak, Arda Ozgun, and Nassira Ghoualmi-Zine. 2024. Llm-based edge intelligence: A comprehensive survey on architectures, applications, security and trustworthiness. *IEEE Open Journal of the Communications Society*.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. 2024. Break the sequential dependency of LLM inference using LOOKAHEAD DECODING. In *Proceedings of the 41st International Conference on*

Machine Learning, ICML'24. JMLR.org. Place: Vienna, Austria.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2024. Minillm: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason Lee, and Di He. 2024. 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)*, pages 1582–1595, Mexico City, Mexico. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The Curious Case of Neural Text Degeneration. In *International Conference on Learning Representations*.

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.

Denis Kocetkov, Raymond Li, Loubna Ben allal, Jia LI, Chenghao Mou, Yacine Jernite, Margaret Mitchell, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro Von Werra, and Harm de Vries. 2023. The Stack: 3 TB of permissively licensed source code. *Transactions on Machine Learning Research*.

Wouter Kool, Herke van Hoof, and Max Welling. 2020. Ancestral Gumbel-Top-k Sampling for Sampling Without Replacement. *Journal of Machine Learning Research*, 21(47):1–36.

Khang Nguyen Le, Ryo Sato, Dai Nakashima, Takeshi Suzuki, and Minh Le Nguyen. 2025. Optiprune: Effective pruning approach for every target sparsity. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 3600–3612.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org. Place: Honolulu, Hawaii, USA.

Minghan Li, Xilun Chen, Ari Holtzman, Beidi Chen, Jimmy Lin, Wen-tau Yih, and Xi Victoria Lin. 2024a. Nearest Neighbor Speculative Decoding for LLM Generation and Attribution. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024b. EAGLE-2: Faster Inference of Language Models with Dynamic Draft Trees. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7421–7432, Miami, Florida, USA. Association for Computational Linguistics.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100.

Fangcheng Liu, Yehui Tang, Zhenhua Liu, Yunsheng Ni, Duyu Tang, Kai Han, and Yunhe Wang. 2024a. Kangaroo: Lossless Self-Speculative Decoding for Accelerating LLMs via Double Early Exiting. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Yiheng Liu, Hao He, Tianle Han, Xu Zhang, Mengyuan Liu, Jiaming Tian, Yutong Zhang, Jiaqi Wang, Xiaohui Gao, Tianyang Zhong, Yi Pan, Shaochen Xu, Zihao Wu, Zhengliang Liu, Xin Zhang, Shu Zhang, Xintao Hu, Tuo Zhang, Ning Qiang, Tianming Liu, and Bao Ge. 2025. Understanding llms: A comprehensive overview from training to inference. *Neurocomputing*, 620:129190.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2024b. LLM-QAT: Data-free quantization aware training for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 467–484, Bangkok, Thailand. Association for Computational Linguistics.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720.

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. 2024. SpecInfer: Accelerating Large Language Model Serving with Tree-based Speculative Inference and Verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS '24, pages 932–949, New York, NY, USA. Association for Computing Machinery. Event-place: La Jolla, CA, USA.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, 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*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, C. J. Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. GPT-4 Technical Report.

Jie Ou, Yueming Chen, and Prof. Tian. 2024. Lossless Acceleration of Large Language Model via Adaptive N-gram Parallel Decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, pages 10–22, Mexico City, Mexico. Association for Computational Linguistics.

Yeonhong Park, Jake Hyun, SangLyul Cho, Bonggeun Sim, and Jae W. Lee. 2024. Any-precision llm: Low-cost deployment of multiple, different-sized llms. In *Proceedings of the 41st International Conference on Machine Learning*.

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. 2024. Code Llama: Open Foundation Models for Code. _eprint: 2308.12950.

Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodola. 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)*, pages 12336–12355, Toronto, Canada. Association for Computational Linguistics.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter.

2023a. A simple and effective pruning approach for large language models. *ArXiv*, abs/2306.11695.

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. 2023b. SpecTr: fast speculative decoding via optimal transport. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc. Event-place: New Orleans, LA, USA.

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, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. *ArXiv*, abs/2310.06694.

Nan Yang, Tao Ge, Liang Wang, Binxing Jiao, Daxin Jiang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. Inference with Reference: Lossless Acceleration of Large Language Models. _eprint: 2304.04487.

Seongjun Yang, Gibbeum Lee, Jaewoong Cho, Dimitris Papailiopoulos, and Kangwook Lee. 2024. Predictive Pipelined Decoding: A Compute-Latency Trade-off for Exact LLM Decoding. *Transactions on Machine Learning Research*.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. 2024a. 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)*, pages 11263–11282, Bangkok, Thailand. Association for Computational Linguistics.

Songming Zhang, Xue Zhang, Zengkui Sun, Yufeng Chen, and Jinan Xu. 2024b. Dual-space knowledge distillation for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18164–18181, Miami, Florida, USA. Association for Computational Linguistics.

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. 2024. Atom: Low-bit quantization for efficient and accurate llm serving. *Proceedings of Machine Learning and Systems*, 6:196–209.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. 2024. DistillSpec: Improving Speculative Decoding via Knowledge Distillation. In *The Twelfth International Conference on Learning Representations*.

A More on Speculative Decoding

Autoregressive decoding (Touvron et al., 2023; Bai et al., 2023; Jiang et al., 2023; OpenAI et al., 2024), suffers from inefficiency because it generates text one token at a time (Figure 5, Left). Speculative decoding (Chen et al., 2023; Leviathan et al., 2023) follows a *guess-and-verify* paradigm (Figure 5, Right). In speculative decoding, a smaller LLM (draft model) (Chen et al., 2023; Leviathan et al., 2023; Miao et al., 2024; Sun et al., 2023b; Zhou et al., 2024; Cai et al., 2024) or the original LLM trained in a specialized manner (self-speculative decoding) (Elhoushi et al., 2024; Liu et al., 2024a; Yang et al., 2024; Zhang et al., 2024a; Li et al., 2024b) predicts multiple tokens in advance. The original LLM then verifies these predictions in parallel, improving efficiency.

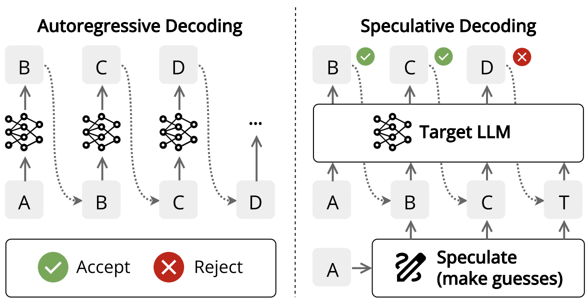

Figure 5: Examples of Autoregressive decoding (Left) and Speculative Decoding (Right). While autoregressive decoding generates one token per forward step, speculative decoding generates three tokens with one forward step.

LLMs process discrete integer sequences as inputs, where each integer represents a token. We define the input sequence as $\mathbf{x} = (x_1, x_2, \ldots, x_s) \in \mathbb{N}^s$ of length s, and denote a slice of length m at step t as $\mathbf{x}_{1:m} = (x_1, x_2, \ldots, x_m)$. The output of an LLM represents the probability distribution over the next token. The probability of generating the s-th token, conditioned on all preceding tokens, is given by $P_M(x_s \mid x_{1:s-1})$. The next token x_s is then sampled from this distribution using various methods (e.g., greedy, top-k, and top-p sampling; see (Kool et al., 2020; Holtzman et al., 2020)). In the case of greedy sampling, the next token is selected as $x_s = \arg\max P_M(x_s \mid x_{1:s-1})$

Let $\mathbf{x}$ be the prompt tokens provided by the user. The LLM generates an output sequence of length m, with each generated token y_i computed autoregressively. Assuming greedy sampling, the decoding process follows:

$$\begin{cases} y_1 = \arg\max P_M(y_1 \mid \mathbf{x}) \\ y_2 = \arg\max P_M(y_2 \mid y_1, \mathbf{x}) \\ \vdots \\ y_m = \arg\max P_M(y_m \mid y_{1:m-1}, \mathbf{x}). \end{cases} \tag{1}$$

A.1 Speculative Decoding

Speculative decoding follows a *Guess-And-Verify* approach, where multiple candidate future tokens are speculated and subsequently verified in a single decoding step. With tree attention (Miao et al., 2024), multiple drafts can be verified simultaneously. Let G denote the number of guesses, and define the set of guesses as $\tilde{Y} = \{\tilde{y}^{(1)}, \tilde{y}^{(2)}, \ldots, \tilde{y}^{(G)}\}$, where each guess sequence has length K. The j-th token of the i-th guess is denoted as $\tilde{y}_j^{(i)}$.

In the case of speculative decoding with greedy sampling, given the prompt $\mathbf{x}$, a drafting method is used to generate the draft sequences $\tilde{Y}$. Using these drafts, the LLM then computes the true tokens $(y'_1, y'_2, \ldots, y'_K)$ in parallel. For instance, for the guess sequence $\tilde{y}^{(1)}$, the true tokens are determined as:

$$\begin{cases} y'_1 = \arg\max P_M(y_1 \mid \mathbf{x}) \\ y'_2 = \arg\max P_M(y_2 \mid \tilde{y}_1^{(1)}, \mathbf{x}) \\ \vdots \\ y'_K = \arg\max P_M(y_K \mid \tilde{y}_{1:K-1}^{(1)}, \mathbf{x}). \end{cases} \tag{2}$$

These generated tokens are then verified. Let h be the highest number of correct guessed tokens across all guesses. Consequently, $h+1$ tokens are generated in one forward step. Algorithm 2 outlines speculative decoding with greedy sampling.

B Implementation Details

B.1 Frameworks and Libraries

We implement SPECTRA in Python using PyTorch 2.1.0 and the Hugging Face `transformers` library (version 4.36.2).

B.2 Models and Checkpoints

We run our experiments primarily with:

- **LLaMA-2-Chat** (Touvron et al., 2023) in sizes 7B, 13B, 70B.
- **CodeLlama** (Rozière et al., 2024) in sizes 7B and 13B.
- **LLaMA-3-Instruct** (Dubey et al., 2024) in sizes 8B and 70B.

All checkpoints are sourced from official repositories or Hugging Face without fine-tuning or modification. For the 7B and 13B models, we use 16-bit (FP16) precision with a pre-allocated key-value cache. For large-scale models such as LLaMA-2-70B and LLaMA-3-70B, we quantize them to 8-bit for the primary results presented in Table 1. Additionally, we evaluate the 70B models in FP16 precision, as reported in Appendix E. We also verify numerical consistency by comparing the 32-bit and 16-bit outputs of LLaMA-2-7B, detailed in Appendix F.

B.3 Hardware

Most experiments are conducted on a single NVIDIA A100 GPU with 80GB of memory. To analyze hardware-specific scaling (Appendix C), we also test on other NVIDIA GPUs, including the RTX 3090, RTX 8000, A40, and A6000. For the largest models (70B) that exceed single-GPU memory constraints under FP16 settings, we distribute computation across multiple GPUs (2x, 4x, or 8x H100) using Hugging Face's pipeline parallelism (Appendix E).

B.4 Hyperparameters

Lookahead, REST, and ANPD. We replicate each baseline using their publicly available GitHub code, keeping to the default settings and hyperparameters outlined in the original papers.

Spectra. By default, we use a 5-gram setup for forward/backward dictionaries. A candidate pool of size $\mathbf{W} = 15$ is maintained per key to generate new n-gram records. After each forward pass, candidate sequences are shifted by one token and then re-populated. We introduce a threshold $\tau \in [0, 1]$, set to 0.1 by default, to determine when to force the selection of a token not yet present in the forward dictionary. At each speculative decoding step, up to $\mathbf{G} = 15$ guesses are allowed. Internal guesses receive priority, and if the guess limit is not reached, external guesses are added.

For external lookups, we implement a Trie structure for rapid prefix queries, following a design similar to REST (He et al., 2024). For **conversation** tasks (e.g., MT-Bench), we gather approximately 100k examples from the UltraChat dataset (Ding et al., 2023), focusing on those with minimal perplexity under the *same* LLM we aim to accelerate. For **code** tasks (e.g., HumanEval, MBPP), we draw from TheStack (Kocetkov et al., 2023) and again refine it to the 100k snippets with the lowest perplexity for memory efficiency. We measure perplexity by running a single forward pass (in streaming mode) over candidate samples and ranking them.

All speedup and throughput metrics are computed at a batch size of 1. In code generation tasks, the maximum generation length is typically 512 tokens, whereas for conversation tasks (MT-Bench, GSM8K), we allow up to 1024 tokens or stop early if the model outputs an end-of-sequence token. All random seeds are set to 0.

C Evaluating SPECTRA in Different GPU Types

Table 3 reports speedups on GSM8K and MT-Bench across four GPUs with varying memory throughput and compute capabilities. While absolute wall-clock times differ across GPUs, the *relative* accelerations remain consistent. SPECTRA consistently outperforms other baselines, including Lookahead, achieving higher speedups in all cases. On older GPUs (e.g., RTX 3090 or RTX 8000), the gap between Lookahead and SPECTRA narrows slightly due to less efficient parallelism, but SPECTRA maintains its lead. These results demonstrate that SPECTRA is robust to hardware variations and effective across both data-center and consumer-grade GPUs.

GPU	Method	GSM8K		MTBench	
		Speedup	τ	Speedup	τ
A40	Lookahead	1.49	1.93	1.53	2.07
	SPECTRA	**1.92**	**2.46**	**1.84**	**2.36**
A6000	Lookahead	1.48	1.92	1.52	2.06
	SPECTRA	**1.92**	**2.46**	**1.84**	**2.36**
RTX8000	Lookahead	1.33	1.93	1.34	2.08
	SPECTRA	**1.70**	**2.46**	**1.58**	**2.35**
RTX3090	Lookahead	1.32	1.92	1.30	2.06
	SPECTRA	**1.84**	**2.46**	**1.74**	**2.36**

Table 3: Hardware scalability of SPECTRA decoding on GSM8K and MTBench for various GPU architectures.

D Details Results with Throughputs

We provide a detailed throughput analysis to complement the speedup ratios reported in the main text. Our goal is to demonstrate how SPECTRA scales across various model sizes, datasets, and GPU architectures. We measure throughput using two key metrics:

- **Macro Throughput (Mac-TP).** Calculated as the average of per-generation token-processing rates—i.e., for each generation step i, we compute $token_i/time_i$ and then average over all steps.
- **Micro Throughput (Mic-TP).** Calculated as the total number of generated tokens divided by the total elapsed time

Table 5 focuses on GSM8K and MTBench performance across four different GPU models, while Table 4 provides more granular results on additional datasets and model configurations. In all cases, SPECTRA consistently achieves higher throughput than both non-speculative baselines and other training-free accelerators, as evidenced by improvements in both Mic-TP and Mac-TP. Notably, this performance advantage remains stable even on older GPUs (e.g., the RTX 3090 and RTX 8000), demonstrating SPECTRA's robustness to varying hardware capabilities.

E Evaluating SPECTRA in Multi-GPU Environments

A critical consideration for practical deployment is how SPECTRA scales when models are distributed across multiple GPUs—a common requirement for large LLMs exceeding single-device memory capacity. To evaluate this, we measure SPECTRA's

Model	Method	Classeval		GSM8K		Humaneval		MBPP		MTBench	
		Mac-TP	Mic-TP	Mac-TP	Mic-TP	Mac-TP	Mic-TP	Mac-TP	Mic-TP	Mac-TP	Mic-TP
Greedy (temperature=0)											
CL-13B	Autoregressive	30.85	30.85	32.03	32.03	32.35	32.35	32.07	32.07	30.69	30.63
	ANPD	59.77	58.03	89.99	89.18	67.43	64.65	86.76	86.41	80.10	76.68
	Lookahead	69.28	68.62	89.73	89.00	74.33	73.23	93.38	92.80	79.38	78.67
	REST	39.53	37.73	29.93	29.47	51.15	47.49	27.41	27.39	28.92	27.18
	SPECTRA (Ours)	**73.47**	**72.98**	**93.36**	**93.23**	**84.91**	**84.41**	**105.44**	**105.39**	**81.32**	**80.68**
CL-7B	Autoregressive	41.17	41.17	41.17	41.17	41.41	41.41	41.60	41.60	38.91	38.93
	ANPD	94.76	93.02	132.26	131.30	89.26	87.13	131.35	130.99	130.41	126.64
	Lookahead	106.51	105.95	123.04	121.90	103.45	103.51	120.75	120.23	125.58	124.77
	REST	59.49	56.61	37.61	37.21	70.38	65.22	40.11	40.09	39.64	36.70
	SPECTRA (Ours)	**111.09**	**110.68**	**137.24**	**136.86**	**122.54**	**122.41**	**148.32**	**148.07**	**143.98**	**144.32**
L2-13B	Autoregressive	31.85	31.56	32.40	32.43	32.27	32.27	32.19	32.19	31.93	31.78
	ANPD	43.30	44.44	47.54	45.22	43.24	42.28	36.20	35.84	37.44	34.84
	Lookahead	57.49	58.94	47.44	47.62	55.76	55.58	44.41	44.15	48.11	46.62
	REST	38.81	37.74	30.36	30.22	40.47	39.70	30.70	30.67	36.39	37.02
	SPECTRA (Ours)	**63.64**	**64.31**	**59.21**	**58.63**	**63.39**	**63.18**	**52.43**	**52.19**	**56.04**	**53.75**
L2-70B	Autoregressive	2.60	2.60	2.61	2.61	2.61	2.61	2.63	2.63	2.60	2.60
	ANPD	4.72	4.80	4.25	4.10	4.85	4.76	3.07	3.07	3.47	3.30
	Lookahead	6.90	7.16	4.87	5.12	6.71	6.73	3.92	3.93	5.05	5.02
	SPECTRA (Ours)	**8.07**	**8.35**	**6.58**	**6.75**	**8.41**	**8.41**	**4.88**	**4.88**	**6.32**	**6.22**
L2-7B	Autoregressive	40.33	40.32	41.01	41.03	41.14	41.13	41.00	41.04	40.48	40.50
	ANPD	65.54	68.10	62.40	59.38	63.27	59.98	48.94	47.67	52.47	50.06
	Lookahead	88.41	91.05	68.00	68.20	84.69	83.87	59.79	60.76	70.04	69.07
	REST	54.74	53.93	41.43	41.38	57.99	56.41	41.28	40.74	50.58	51.79
	SPECTRA (Ours)	**96.88**	**98.75**	**86.51**	**85.50**	**98.77**	**98.38**	**72.39**	**73.22**	**81.93**	**79.20**
L3-70B	Autoregressive	2.58	2.57	2.58	2.58	2.59	2.59	2.59	2.59	2.55	2.55
	ANPD	3.97	4.19	3.86	3.72	4.72	4.75	3.77	3.59	3.14	3.03
	Lookahead	6.17	6.47	3.99	3.96	6.63	6.75	3.70	3.66	4.49	4.53
	SPECTRA (Ours)	**6.87**	**7.18**	**5.43**	**5.34**	**7.33**	**7.50**	**5.01**	**4.88**	**5.25**	**5.16**
L3-8B	Autoregressive	36.59	36.58	36.74	36.74	36.20	36.21	35.24	35.20	36.55	36.69
	ANPD	77.21	78.76	141.89	141.36	66.31	65.57	118.47	112.95	41.77	40.20
	Lookahead	94.92	97.09	136.32	135.92	89.99	90.47	133.67	133.12	56.09	55.49
	SPECTRA (Ours)	**103.61**	**105.88**	**142.89**	**142.72**	**92.86**	**93.16**	**143.80**	**142.72**	**61.69**	**60.22**
Sampling (temperature=1.0)											
CL-13B	Autoregressive	30.90	30.64	31.38	31.37	31.24	31.39	31.46	31.45	30.71	30.67
	ANPD	35.48	34.86	33.54	32.34	32.64	34.36	31.57	30.95	70.92	65.68
	Lookahead	42.54	40.74	33.79	32.49	40.25	42.17	32.02	31.19	71.50	68.46
	REST	35.15	33.22	25.67	25.24	39.58	38.49	26.43	25.89	28.41	26.69
	SPECTRA (Ours)	**51.86**	**50.04**	**37.57**	**35.67**	**51.60**	**52.64**	**36.29**	**35.27**	**72.90**	**69.98**
CL-7B	Autoregressive	39.60	39.58	40.85	40.87	40.05	40.10	40.81	40.81	40.49	40.50
	ANPD	50.89	51.76	47.44	46.68	44.14	46.34	45.86	45.81	112.29	103.57
	Lookahead	60.87	60.29	48.54	47.64	57.12	61.14	48.64	48.27	110.07	105.00
	REST	48.64	46.41	35.98	35.46	53.35	52.26	37.04	36.57	39.36	36.51
	SPECTRA (Ours)	**71.70**	**71.78**	**55.24**	**52.81**	**67.27**	**69.20**	**54.48**	**52.91**	**112.43**	**108.49**
L2-13B	Autoregressive	31.23	31.17	31.44	31.47	31.41	31.42	32.02	32.06	31.67	31.59
	ANPD	37.53	37.94	39.11	37.99	36.79	36.75	32.97	32.71	36.91	34.34
	Lookahead	47.59	47.35	41.60	41.76	46.33	46.51	37.82	37.82	47.35	45.48
	REST	36.78	36.17	29.33	29.25	37.46	36.71	29.38	29.28	35.50	36.21
	SPECTRA (Ours)	**53.13**	**52.28**	**48.60**	**48.11**	**52.93**	**53.11**	**42.95**	**43.03**	**54.98**	**52.42**
L2-7B	Autoregressive	39.89	39.88	40.58	40.59	40.09	40.10	40.59	40.66	40.65	40.70
	ANPD	52.14	52.78	54.23	52.90	51.40	50.97	44.73	43.77	50.92	48.24
	Lookahead	70.82	71.17	61.15	61.34	68.78	69.01	50.84	51.83	68.27	66.77
	REST	50.35	49.99	40.19	40.09	50.86	50.06	38.94	38.18	49.12	50.54
	SPECTRA (Ours)	**78.46**	**78.74**	**72.13**	**71.68**	**81.71**	**81.76**	**59.77**	**60.09**	**80.21**	**77.00**
L3-8B	Autoregressive	35.75	35.76	35.16	35.17	36.01	36.02	36.05	36.07	35.39	35.48
	ANPD	44.71	43.72	69.12	66.73	51.48	51.57	68.03	64.54	40.84	39.23
	Lookahead	53.05	50.57	72.68	69.11	64.59	63.79	71.88	68.90	55.46	53.74
	SPECTRA (Ours)	**69.50**	**68.92**	**79.88**	**76.53**	**69.09**	**68.62**	**78.99**	**76.69**	**60.33**	**57.69**

Table 4: Micro throughput (Mic-TP) and Macro throughput (Mac-TP) across multiple tasks and models.

GPU	Method	GSM8K		MTBench	
		Mac-TP	Mic-TP	Mac-TP	Mic-TP
A40	Autoregressive	32.66	32.66	32.14	31.66
	Lookahead	48.59	48.73	49.13	47.96
	SPECTRA	**62.56**	**61.52**	**59.00**	**56.80**
A6000	Autoregressive	39.15	39.17	38.78	38.24
	Lookahead	58.13	58.30	58.84	57.40
	SPECTRA	**75.20**	**74.16**	**71.3**	**69.28**
RTX8000	Autoregressive	34.03	34.27	34.21	34.02
	Lookahead	45.25	45.42	45.73	44.16
	SPECTRA	**57.95**	**57.09**	**54.16**	**52.32**
RTX3090	Autoregressive	40.67	40.76	41.17	41.22
	Lookahead	53.69	53.75	53.51	52.09
	SPECTRA	**74.87**	**73.88**	**71.58**	**69.79**

Table 5: Throughput results for different GPU types on GSM8K and MTBench.

performance under three distributed configurations of LLaMA-2-70B: (1) 2xH100 with full precision, (2) 4xH100 with full precision, and (3) 8xH100 with full precision. We also include a baseline of 1xH100 with 8-bit quantization for memory-constrained single-GPU inference. Table 6 reports throughput and speedup metrics.

SPECTRA achieves consistent speedups of 2.00—2.03× across all multi-GPU configurations while maintaining a stable compression ratio (τ) of 2.52. This demonstrates robust scalability—partitioning model weights introduces minimal overhead, and the speculative verification process remains efficient despite inter-GPU communication. Notably, even in the quantized single-GPU setting, SPECTRA provides a 2.43× speedup, outperforming standard autoregressive decoding. These results validate SPECTRA's practicality for large-scale deployments where memory constraints necessitate distributed inference.

F Verifying Generation Quality with SPECTRA Decoding

Greedy Decoding Performance. To assess the quality of greedy decoding, we compare the inference results of the LLaMA-2-7B Chat model using SPECTRA Decoding against Hugging Face's standard greedy search. Our baseline consists of single-precision (FP32) inference on 160 conversational turns from the MT-Bench dataset. Under FP32, SPECTRA Decoding produces identical outputs to the baseline.

However, when transitioning to half-precision (FP16), even Hugging Face's native greedy search generates 25 discrepancies (out of 160) compared to the FP32 baseline. SPECTRA Decoding exhibits a similar discrepancy rate (26), confirming that it maintains the output distribution within the numerical error margins typically observed in standard half-precision inference libraries.

Sampling Decoding Performance. We also assess generation quality under a stochastic sampling setting (temperature = 1.0). As detailed in Table 7, SPECTRA Decoding produces ROUGE-1, ROUGE-2, and ROUGE-L scores on both the CNN/DailyMail (Nallapati et al., 2016) and XSum (Narayan et al., 2018) summarization datasets that are nearly identical to those of standard autoregressive sampling. At the same time, SPECTRA achieves notable speedups (1.60× on CNN/DailyMail and 1.69× on XSum) with compression ratios of 2.05 and 2.08, respectively. These results confirm that SPECTRA Decoding accelerates inference while preserving generation quality across diverse tasks.

These findings reaffirm that SPECTRA Decoding, does not degrade generation quality compared to conventional greedy or sampling-based methods.

G Token Acceptance Rate Analysis

Figure 6 plots the cumulative number of accepted tokens versus decoding steps for each dataset (MT-Bench, HumanEval, MBPP, and GSM8K) using LLama2-7B-chat with greedy decoding. The steeper ascent of the SPECTRA curve indicates that our method requires substantially fewer decoding

GPU & Model Setting	Method	MTBench			
		Mac-TP	Mic-TP	Speedup	τ
1xH100 - Quantized Int8	Autoregressive	2.60	2.60	1.00	1.00
	SPECTRA	**6.32**	**6.22**	**2.43**	**2.51**
2xH100 - FP16	Autoregressive	14.81	14.70	1.00	1.00
	SPECTRA	**29.62**	**28.91**	**2.00**	**2.52**
4xH100 - FP16	Autoregressive	14.60	14.48	1.00	1.00
	SPECTRA	**29.67**	**28.89**	**2.03**	**2.52**
8xH100 - FP16	Autoregressive	14.39	14.28	1.00	1.00
	SPECTRA	**29.27**	**28.55**	**2.03**	**2.52**

Table 6: Results in multi-GPU Enviroments on GSM8K and MTBench using LLama-2-chat-70B.

Dataset	Method	ROUGE-1	ROUGE-2	ROUGE-L	Speedup	τ
CNN	Autoregressive	9.77	0.39	7.20	1.00	1.00
	SPECTRA	9.74	0.41	7.18	1.60	2.05
XSUM	Autoregressive	18.12	4.36	12.43	1.00	1.00
	SPECTRA	18.13	4.40	12.49	1.69	2.08

Table 7: Evaluation of SPECTRA Decoding on CNN/DailyMail and XSum using a temperature of 1.0. ROUGE scores, speedups over autoregressive decoding, and compression ratio (τ) are reported for LLaMA-2-7B-Chat.

steps compared to alternatives, for example, almost two times shorter than ANPD. This improvement is attributed to a higher token acceptance rate, which in turn reduces the overall number of decoding iterations and enhances the efficiency of the generation process.

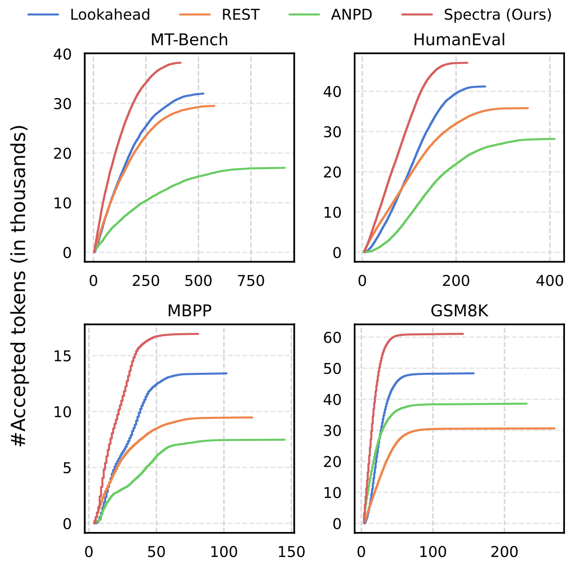

Figure 6: Total number of accepted tokens across all samples at each decoding step.

H Algorithms

Algorithm 2 Speculative Decoding (Multiple guesses and Greedy Sampling)

Given guess size K, number of guesses G, and target length T.
Given initial prompt sequence $\mathbf{x}$.
while $n < T$ **do**
    Obtain multiple drafts $\tilde{Y} = \{\tilde{y}^{(1)}, \tilde{y}^{(2)}, \dots, \tilde{y}^{(G)}\}$.
    In parallel, compute $K + 1$ verification tokens y':
    for $i = 1 : K$ **do**
        $y_i'^{(g)} = \arg\max P_M(y_i \mid \tilde{y}_{i-1}^{(g)}, \mathbf{x}), \quad \forall g \in \{1, \dots, G\}$
    end for
    Identify the sequence $\tilde{y}^{(g^*)}$ with the highest token matches and the corresponding $y'^{(g)}$.
    for $t = 1 : K$ **do**
        if $y_t'^{(g)} = \tilde{y}_t^{(g^*)}$ **then**
            Set $y_{n+t} \leftarrow \tilde{y}_t^{(g^*)}$ and $n \leftarrow n + 1$.
        else
            $y_{n+t} \leftarrow y_t'^{(g)}$ and exit for loop.
        end if
    end for
end while

Algorithm 3 Greedy Verification with SPECTRA DECODING

Require: sequence $\mathbf{x}$, model P_M, guesses $\mathcal{G} = \{g^i\}$ with $i \in [0, G-1]$
Ensure: o {accepted tokens of length 1 to N}

function GREEDYVERIFICATION($\mathbf{x}, P_M, \mathcal{G}$)
  $D \leftarrow \emptyset$ ▷ Store the distributions
  $V \leftarrow \mathcal{G}$ ▷ Store the current guesses
  for $i = 0$ to $G-1$ **do**
    D.append$(P_M(g'^{(i)}, x_{\text{next}} | g^{(i)}, \mathbf{x}))$ ▷ Last token of $\mathbf{x}$ and $g^{(i)}$ outputs – total N distributions
  end for
  for $i = 1$ to $N-1$ **do**
    $j \leftarrow 1$
    is_accept $\leftarrow 0$
    $\mathcal{P} \leftarrow D[1]_i$
    while $j \leq$ size(V) **do**
      $s_j \leftarrow V[j]_i$
      if $s_j = \arg\max \mathcal{P}$ **then** ▷ accepted, update all potential speculations and probabilities
        o.append(s_j)
        is_accept $\leftarrow 1$
        $V_{\text{new}}, D_{\text{new}} \leftarrow \emptyset, \emptyset$
        for $k = j$ to size(V) **do**
          if $s_j = V[k]_i$ **then**
            V_{new}.append$(V[k])$
            D_{new}.append$(D[k])$
          end if
        end for
        $V, D \leftarrow V_{\text{new}}, D_{\text{new}}$
        break
      else ▷ rejected, go to next speculation
        $j \leftarrow j + 1$
      end if
    end while
    if is_accept **then**
      continue
    else ▷ guarantee one step movement
      o.append$(\arg\max \mathcal{P})$
      break
    end if
  end for
  if is_accept **then**
    o.append$(\arg\max D[1]_N)$
  end if
  return o
end function

Algorithm 4 Sample Verification with SPECTRA DECODING

Require: sequence x, model P_M, guesses g^i with $i \in [0, G-1]$

Ensure: o {accepted tokens of length 1 to N}

```
function SAMPLEVERIFICATION(x, P_M, g)
    D ← ∅                                                ▷ Store the distributions
    V ← 𝒢                                                ▷ Store the current guesses
    for i = 0 to G − 1 do
        D.append(P_M(g'^(i), x_next | g^(i), x))   ▷ Last token of x and g^(i) outputs – total N distributions
    end for
    for i = 1 to N − 1 do
        j ← 1
        is_accept ← 0
        𝒫_j ← D[j]_i
        while j ≤ size(V) do
            s_j ← V[j]_i
            sample r ∼ U(0, 1)
            if r ≤ 𝒫_j(s_j) then        ▷ accepted, update all potential speculations and probabilities
                o.append(s_j)
                is_accept ← 1
                V_new, D_new ← ∅, ∅
                for k = j to size(V) do
                    if s_j = V[k]_i then
                        V_new.append(V[k])
                        D_new.append(D[k])
                    end if
                end for
                V, D ← V_new, D_new
                break
            else                                         ▷ rejected, go to next speculation
                𝒫_j(s_j) ← 0
                𝒫_{j+1} = norm(𝒫_j)
                j ← j + 1
            end if
        end while
        if is_accept then
            continue
        else                                             ▷ guarantee one step movement
            sample x_next ∼ 𝒫_j
            o.append(x_next)
            break
        end if
    end for
    if is_accept then
        o.append(sample x_next ∼ D[1]_N)
    end if
    return o
end function
```