SpecBound: Adaptive Bounded Self-Speculation with Layer-wise Confidence Calibration

Zhuofan Wen[1,2,3], **Yang Feng**[1,2,3†]
[1]Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences (ICT/CAS) [2] State Key Laboratory of AI Safety, Institute of Computing Technology, Chinese Academy of Sciences
[3] University of Chinese Academy of Sciences, Beijing, China
wenzhuofan24z@ict.ac.cn, fengyang@ict.ac.cn

Abstract

Speculative decoding has emerged as a promising approach to accelerate autoregressive inference in large language models (LLMs). Self-draft methods, which leverage the base LLM itself for speculation, avoid the overhead of auxiliary draft models but face limitations: shallow layers often produce overconfident yet incorrect token predictions, and the presence of difficult tokens in a draft sequence forces redundant computation through deeper layers, undermining both draft acceptance and overall speedup. To address these issues, we propose a novel self-draft framework that suppresses spurious confidence via layer-wise temperature annealing in early-exit decision and adaptively bounds speculation length based on token-wise decoding difficulty. By reprocessing the hidden states of draft tokens in a unified parallel pass through deep layers, our method maintains exact output equivalence with the original model while maximizing computational efficiency. It requires no modifications to the base LLM parameters and achieves up to **2.33×** wall-time speedup over standard autoregressive decoding across diverse long-form generation tasks and multiple model architectures.[1]

1 Introduction

Large Language Models (LLMs) excel at diverse text generation tasks (Achiam et al., 2023; Guo et al., 2025), yet practical applications increasingly require long outputs, such as chain-of-thought reasoning (Wei et al., 2022), multilingual capability (Bu and Feng, 2026) and tool coordination in AI agents (Shen et al., 2023). For standard autoregressive decoding, the combination of these long sequences and ever-larger model sizes leads to high inference latency, motivating the need for more efficient parallel decoding strategies (Geng et al., 2021; Nie et al., 2025; Zhang et al., 2025). Among these, speculative decoding (SD) has been proposed as a practical framework for accelerating inference (Leviathan et al., 2023; Chen et al., 2023).

Early SD methods rely on an independent draft model—a smaller, separately trained network that proposes candidate tokens—requiring additional model selection and training overhead (Cai et al., 2023; Li et al., 2024; Xia et al., 2023; Wen et al., 2024). To eliminate this dependency, recent work has shifted toward self-draft strategies that leverage the base LLM itself for speculation. These approaches broadly fall into three categories: (1) fine-tuning, where the LLM is adapted with non-autoregressive objectives to enable parallel token prediction, but at the risk of degrading generalization due to parameter updates (Lin et al., 2025; Yi et al., 2024; Gloeckle et al., 2024); (2) layer skip, which accelerates draft generation by computing only a subset of layers, yet often compromises output quality and disrupts KV cache consistency (Zhang et al., 2024); and (3) early exit, which generates draft tokens from shallow layers of the base LLM and feeds them into subsequent decoding steps. While early exit ensures lossless output, current implementations achieve limited speedup (Elhoushi et al., 2024; Liu et al., 2024).

Through visualization of intermediate layer dynamics in Figure 1, we observe significant heterogeneity in token-level decoding difficulty (Xia et al., 2025). While most tokens can correctly exit at shallow layers, a small fraction of difficult tokens require deeper computation to resolve semantic ambiguity. Due to the batched verification requirement in speculative decoding, these difficult tokens force the entire draft sequence to be processed through deeper layers, leading to redundant computation. Moreover, pretraining loss function supervises only the final-layer output, leaving shallow layers without direct optimization signals. As a result, shallow layers often assign spurious high confidence to incorrect tokens—a tendency that

†Corresponding author: Yang Feng.
[1]https://github.com/ictnlp/SpecBound

Findings of the Association for Computational Linguistics: ACL 2026, pages 16309–16321

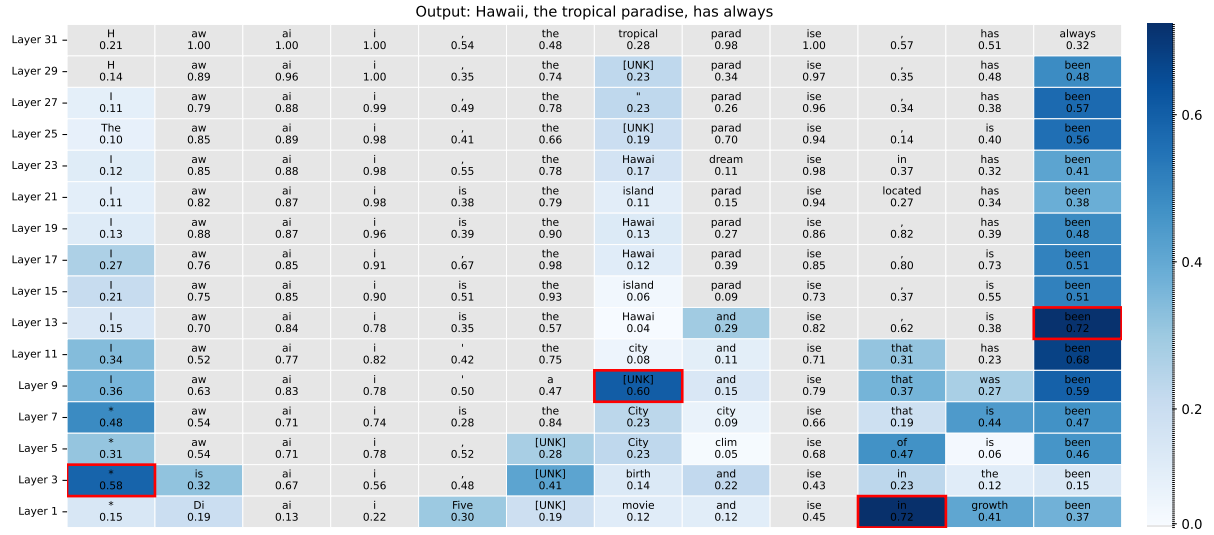

Figure 1: Layer-wise token prediction and confidence of a segment with the MT-Bench prompt. Each cell shows the greedy-decoded token and its confidence (color intensity: darker = higher probability) at a given layer. Gray cells indicate that the token has already correctly exited at an earlier layer and is no longer considered for early-exit decisions. This figure reveals the issues of spurious high-confidence predictions and heterogeneous exit depths.

easily misleads naive early exit methods relying solely on confidence thresholds, severely reducing draft acceptance and limiting acceleration.

To address the above challenges, we propose SpecBound, a self-draft speculative decoding framework featuring two key components. First, to suppress spurious high-confidence predictions in shallow layers, we introduce the Annealed Confidence Threshold early exit mechanism, which applies a layer-dependent temperature schedule during early-exit decisions to smooth softmax outputs and attenuate false peaks. Second, to mitigate redundant computation caused by difficult tokens, we design the Bounded Speculation with Cached States algorithm: during autoregressive draft generation, if any token fails to exit before reaching the maximum depth or the maximum draft length, the current speculation phase is terminated; cached hidden states of all draft tokens are then jointly processed through the remaining layers in a single parallel pass, ensuring lossless output while maximizing parallel efficiency.

Our contributions are as follows: (1) We propose Annealed Confidence Threshold, a more reliable early-exit criterion that suppresses spurious confidence in shallow layers via layer-wise temperature annealing; (2) We design Bounded Speculation with Cached States algorithm that coordinates speculation and verification through adaptive depth and width control; (3) SpecBound achieves up to **2.33**× wall-time speedup over standard autoregressive decoding across diverse long-form generation tasks and multiple base models, requiring no modification to the base LLM parameters and producing outputs identical to the original model.

2 Methods

In this section, we present our proposed acceleration framework, SpecBound. We begin by visualizing the intermediate layer computation of LLM in Section 2.1, which reveals two key patterns observed in our preliminary experiments. Guided by these insights, we introduce the temperature-annealed early-exit mechanism in Section 2.2 and the Bounded Speculation with Cached States Algorithm in Section 2.3. Finally, we provide a theoretical wall-time speedup analysis to demonstrate the effectiveness of our method in Section 2.4.

2.1 Layer-wise Decoding Analysis

Layer-wise Visualization While several recent works have explored early-exit-based approaches for lossless acceleration, their speedup remains limited and currently falls short of the performance achieved by mainstream independent-draft methods. To investigate the factors that constrain the acceleration potential of early exit, we analyze a representative example from MT-Bench which requires generating a blog on traveling in Hawaii.

For each token during generation, we probe the output of every Transformer layer in the base LLM by projecting its hidden state through the language

model (LM) head and applying greedy decoding to obtain the most likely token at that layer (Chuang et al., 2023; Zhu et al., 2025). Notably, to account for the poor predictive capability of shallow layers, we follow AdaDecode (Wei et al., 2025) and train lightweight intermediate-layer LM heads for early-exit decisions. Figure 1 visualizes the predicted token and its confidence at each layer, from which we identify two distinct early-exit patterns that appear to influence the achievable speedup.

Spurious High Confidence in Shallow Layers First, for those challenging tokens that ultimately require deep layers to be correctly predicted (e.g., token "H" and "tropical"), shallow layers often exhibit spuriously high confidence in incorrect tokens which is marked by red boxes in figure 1. We hypothesize that this stems from the pretraining objective of base models: since the loss function supervises only the final-layer output, shallow layers lack direct optimization signals and thus may develop overconfident but incorrect predictions. Since many existing early-exit methods rely on simple threshold-based criteria, these overconfident yet erroneous predictions can easily pass the exit condition, leading to low-quality drafts.

Heterogeneous Token Difficulty Second, as shown in Figure 1, decoding sequences exhibit significant heterogeneity in token-level difficulty. While a consecutive span of tokens, from "aw" to "the", stabilizes correctly at shallow layers, a few semantically challenging tokens, such as "H", "tropical", and "always", remain uncertain through shallow-to-mid layers and require deep computation. Due to the batched verification requirement in speculative decoding, the presence of even one such difficult token forces the entire draft sequence to be processed through deeper layers. This leads to two sources of inefficiency: first, when the draft is accepted, simple tokens incur redundant deep computation that erodes their early-exit speedup potential; second, when rejected, the substantial time spent computing deep representations for the difficult token becomes wasted overhead, directly slowing down end-to-end decoding.

2.2 Annealed Confidence Threshold

To address the issue of spurious high-confidence predictions in shallow layers, we propose the Annealed Confidence Threshold(ACT) early exit mechanism. Unlike prior early-exit methods that determine exit solely by comparing the probability of the greedy-decoded token against a fixed threshold, ACT dynamically modulates the sampling temperature based on layer depth and couples it with a fixed confidence threshold for early-exit decisions, suppressing premature exits caused by false certainty.

Specifically, considering a base LLM that contains L transformer layer, we scale the early exit logits $\mathbf{z}^{(\ell)}$ at layer ℓ ($1 \leq \ell \leq L$) with a layer-dependent temperature T_ℓ:

$$T_\ell = 1 + \alpha \left(1 - \frac{\ell}{L}\right) \tag{1}$$

where L is the total number of layers in the base LLM and $\alpha > 0$ controls the strength of annealing. This linear schedule ensures higher temperatures in shallow layers (e.g., $T_1 = 1 + \alpha$) to flatten the output softmax distribution and reduce overconfidence, while gradually cooling to the standard temperature $T_L = 1$ at the final layer which ensuring that the output distribution remains identical to that of the original base LLM. With adaptive sampling temperature, the confidence of the top-1 token at layer ℓ is then computed as:

$$p^{(\ell)} = \max\left(\text{softmax}\left(\frac{\mathbf{z}^{(\ell)}}{T_\ell}\right)\right) \tag{2}$$

Early exit is triggered at the first layer ℓ where $p^{(\ell)} \geq \tau$, with $\tau \in (0, 1)$ being a predefined threshold. By raising the temperature in shallow layers, ACT effectively lowers the raw confidence $p^{(\ell)}$ for incorrect tokens that would otherwise appear overconfident under $T = 1$, making it harder for them to satisfy the exit condition. This results in more reliable draft tokens without altering the base model's parameters and incurs only negligible computational overhead, as temperature scaling is a lightweight post-activation operation.

2.3 Bounded Speculation with Cached States

To address the issue of heterogeneous exit depths observed in prior work, we introduce the Bounded Speculation with Cached States Algorithm (BSCS) Algorithm, which explicitly limits both the depth and width of speculative generation. We define two hyperparameters: $d_{\max}$: the maximum layer depth allowed for early-exit draft generation; $w_{\max}$: the maximum number of consecutive tokens that can be drafted before triggering verification.This dual-bound design ensures predictable speculation latency and prevents excessive computation

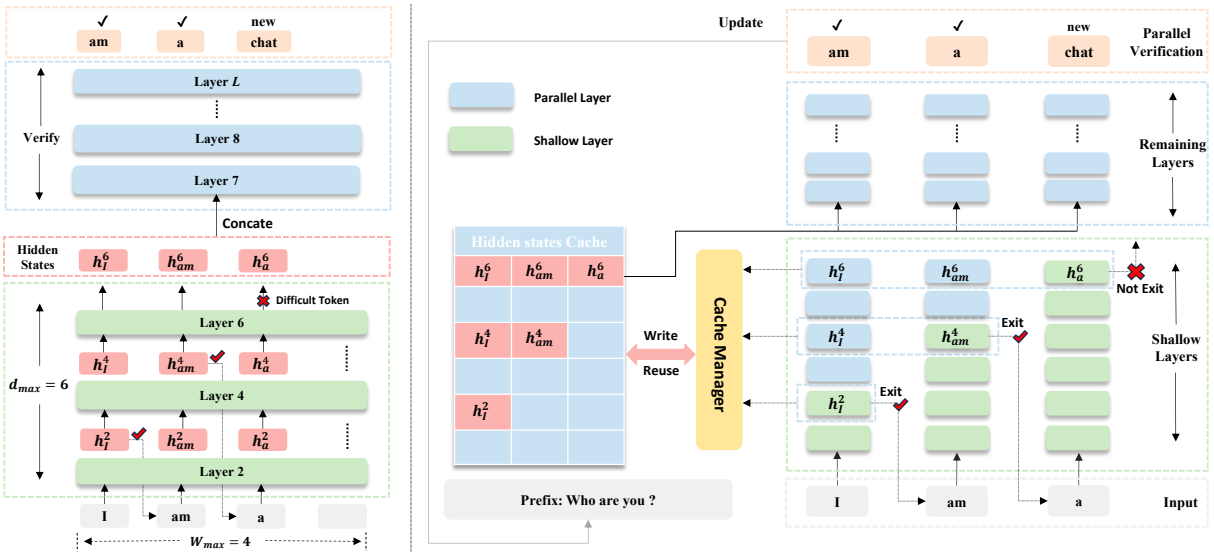

Figure 2: **Left:** Overview of the BSCS algorithm under the setting $d_{\max} = 6$ and $w_{\max} = 4$. When token "a" fails to exit by layer 6, speculation is terminated. All hidden states of previously generated tokens ("I", "am", "a") are concatenated and passed to the remaining layers for parallel verification. **Right:** Detailed illustration of per-token layer-wise computation and hidden-state cache management.

on difficult tokens or error-prone long draft sequences.Figure 2 illustrates a complete cycle of BSCS's draft-verify process using a concrete example with $d_{\max} = 6$ and $w_{\max} = 4$ for clarity, while the actual parameter settings used in our evaluation are reported in Section 3.1.

Handling Difficult Tokens Given a decoding history prefix "Who are you?", the base model early exits token "am" at layer 2 and token "a" at layer 4 at the first two steps. In the third step, the model attempts to draft the next token but fails to meet the early-exit condition mentioned in Section 2.2 even after computing up to the depth bound $d_{\max} = 6$, signaling a difficult token that requires deeper processing. At this point, BSCS immediately terminates the current speculation phase and initiates parallel verification of all previously generated draft tokens. Crucially, since these tokens have their hidden states cached up to Layer $d_{\max}$, they can be jointly processed through the remaining layers ($d_{\max} + 1$ to L) in a single parallel pass. If all prior tokens are accepted, the system resumes decoding from the correct position — effectively using the verification step to also decode the difficult token without wasting prior computation.

Handling Long Shallow Chains Another case BSCS handles is when $w_{\max}$ consecutive tokens exit successfully at shallow layers, the risk of cumulative errors increases as each subsequent draft relies on the previous potentially unreliable predictions. To mitigate this, BSCS also triggers an early verification round. The operation is the same as that used for difficult tokens, except that it first aligns all $w_{\max}$ tokens to the deepest exit layer among them by running their missing layers in parallel which ensuring consistent context before batch verification. This mechanism controls error propagation, ensuring high accuracy for the next draft sequence. Due to space limitation, a detailed illustration of this case is provided in Appendix.

Hidden-State Cache Management To enable the draft interruption and verification triggering described above, BSCS maintains a hidden-state cache that stores intermediate representations. As depicted in Figure 2, the Cache Manager support two core operations: Write and Reuse. In the Write phase, when token t_i exits at layer ℓ, its hidden state $\mathbf{h}_i^{(\ell)}$ is stored in the cache. In the Reuse phase, cached hidden states are retrieved if they are needed, either when subsequent tokens require the missing KV cache of earlier tokens (Vaswani et al., 2017) or when batch verification begins, at which point the hidden states cached at layer $d_{\max}$ are concatenated and processed together through the deeper layers. This design guarantees that every output token undergoes computation across all layers, enabling lossless acceleration.

In summary, BSCS transforms speculative decoding from an unbounded, token-level process into a bounded, block-wise pipeline — balancing

speed, reliability, and resource efficiency.

2.4 Wall-time Speedup Analysis

To demonstrate the effectiveness of SpecBound, we theoretically model its speedup over standard autoregressive decoding and analyze how key parameters influence the acceleration performance.

We define speedup as the ratio of decoding throughput between SpecBound and autoregressive (AR) decoding, where throughput is measured as the number of tokens generated per unit wall-clock time. Let T_{AR} denote the time required by AR decoding to generate one token (equivalent to a full forward pass through all L layers). Our goal is to compare the expected token generation throughput of SpecBound against the AR baseline $1/T_{\text{AR}}$.

Variables Definition We model the decoding throughput of SpecBound using two parameters: w, the number of draft tokens generated per speculation round ($1 \leq w \leq w_{\max}$); and α, the per-token acceptance rate—the probability that a draft token is accepted during verification. Note that $d_{\max}$, $w_{\max}$, and L have been defined in earlier sections.

Calculate Speedup Ratio In our method, a speculation round comprises two phases: (1) Draft phase: sequential generation of w tokens, each with $d_{\max}$ layers; (2) Verification phase: parallel processing of all w tokens through the remaining $L - d_{\max}$ layers. Thus, total time per round is:

$$T_{\text{round}} = \frac{w d_{\max} + L - d_{\max}}{L} \cdot T_{\text{AR}} \tag{3}$$

Not all draft tokens will be accepted every round. Under the standard geometric-acceptance assumption (Leviathan et al., 2023), the expected number of accepted tokens per round $\mathbb{E}[N_{\text{acc}}]$ is:

$$\sum_{k=0}^{w-1} k\alpha^k(1-\alpha) + w\alpha^w = \frac{\alpha(1-\alpha^w)}{1-\alpha} \tag{4}$$

Then our method's throughput is $\mathbb{E}[N_{\text{acc}}]/T_{\text{round}}$. Dividing by the AR decoding throughput $1/T_{\text{AR}}$ and simplifying, we obtain the final speedup ratio (denoted as SD):

$$\text{SD} = \frac{L\alpha(1-\alpha^w)}{(1-\alpha)\,(w d_{\max} + L - d_{\max})}. \tag{5}$$

Parameter Analysis From Equation (5), we observe that the overall speedup increases monotonically with the token acceptance rate α. The Annealed Confidence Threshold early exit mechanism improves α by suppressing spurious tokens at shallow layers. Moreover, speedup can also be enhanced by increasing w (or $w_{\max}$) or reducing $d_{\max}$, both of which lower the per-round time cost; however, such choices may compromise generation quality and, in turn, potentially reduce α, reflecting an empirical trade-off that must be carefully balanced. Detailed experimental analysis of these hyperparameters is provided in Section 3.3.

3 Experiments

3.1 Experimental Setup

Backbone Models and Training We choose Vicuna (Chiang et al., 2023; Touvron et al., 2023) and CodeLlama-Instruct (Roziere et al., 2023) models with 7B and 13B parameters as backbone LLMs. During training, all base model parameters are frozen; only a lightweight LM head is trained per intermediate layer to enable early exit(Wei et al., 2025). The LM heads are trained for 20 epochs on 68K multi-turn conversations from the ShareGPT dataset, using the AdamW optimizer with learning rate 3×10^{-5} which taking approximately 2 hours on four NVIDIA H800 GPUs.

Evaluation Benchmarks and Baselines We evaluate our method on Spec-Bench, a diverse benchmark covering machine translation, question answering, and other text generation tasks(Xia et al., 2024). We compare against a range of established self-drafting approaches, including Lookahead (Fu et al., 2024), Medusa (Cai et al., 2023), REST (He et al., 2024), Kangaroo (Liu et al., 2024), SPACE (Yi et al., 2024), and AdaDecode (Wei et al., 2025).Performance is measured using two key metrics: (1) compression rate (CR), defined as the average number of accepted tokens per speculation round, and (2) wall-time speedup (SD) relative to standard autoregressive decoding. For inference with our method, we set $w_{\max} = 8$, $d_{\max} = 10$, $\alpha = 0.2$, and $\gamma = 0.55$—a configuration that balances draft consumption and generation accuracy; a detailed exploration of inference parameters is provided in the ablation study. For fair comparison, all methods are evaluated on a single NVIDIA H800 GPU. All results are presented in Table 1.

3.2 Main Results

SpecBound achieves better overall speedup performance across tasks and base models. As

Method	Math		Multi.		QA		RAG		Sum.		Trans.		Overall
	CR	SD	CR	SD	CR	SD	CR	SD	CR	SD	CR	SD	
Vicuna-7B													
Lookahead (Fu et al., 2024)	1.92	1.54×	1.74	1.49×	1.52	1.23×	1.53	1.24×	1.57	1.38×	1.28	1.16×	1.35×
Medusa (Cai et al., 2023)	1.77	**1.96**×	1.72	**1.95**×	1.54	1.67×	1.47	1.47×	1.53	1.59×	1.55	1.63×	1.71×
REST (He et al., 2024)	1.49	1.19×	2.04	1.65×	1.87	1.64×	1.96	1.75×	1.60	1.34×	1.58	1.33×	1.47×
Kangaroo (Liu et al., 2024)	2.14	1.61×	2.22	1.68×	1.87	1.43×	2.05	1.52×	1.87	1.50×	1.41	1.24×	1.50×
SPACE (Yi et al., 2024)	2.15	1.65×	2.31	1.71×	1.72	1.33×	1.88	1.24×	2.19	1.47×	1.73	1.37×	1.46×
Ours	**3.78**	1.89×	**3.05**	1.65×	**4.23**	**2.16**×	**5.48**	**2.31**×	**3.97**	**1.95**×	**6.41**	**2.94**×	**2.15**×
Vicuna-13B													
Lookahead (Fu et al., 2024)	1.98	1.61×	1.64	1.41×	1.43	1.20×	1.48	1.19×	1.54	1.30×	1.22	1.07×	1.30×
Medusa (Cai et al., 2023)	1.76	**2.06**×	1.86	**2.07**×	1.53	1.71×	1.49	1.59×	1.57	1.64×	1.65	1.73×	1.81×
REST (He et al., 2024)	1.59	1.21×	1.94	1.50×	1.96	1.55×	1.83	1.53×	1.62	1.35×	1.57	1.19×	1.37×
Kangaroo (Liu et al., 2024)	2.42	1.63×	2.44	1.66×	1.79	1.34×	2.16	1.40×	2.00	1.41×	1.45	1.18×	1.44×
Ours	**4.09**	1.91×	**3.22**	1.69×	**4.54**	**2.18**×	**5.25**	**2.26**×	**4.73**	**2.18**×	**5.93**	**2.77**×	**2.16**×
Codellama-7B-Instruct													
Lookahead (Fu et al., 2024)	1.88	1.52×	1.78	1.51×	1.49	1.21×	1.57	1.26×	1.45	1.32×	1.32	1.18×	1.33×
Medusa (Cai et al., 2023)	1.78	1.96×	1.68	**1.92**×	1.58	1.69×	1.44	1.45×	1.57	1.61×	1.51	1.61×	1.70×
REST (He et al., 2024)	1.45	1.17×	2.07	1.67×	1.84	1.62×	1.99	1.77×	1.57	1.32×	1.62	1.35×	1.47×
AdaDecode (Wei et al., 2025)	2.20	1.62×	2.00	1.40×	2.30	1.50×	2.50	1.48×	2.15	1.38×	2.40	1.52×	1.45×
Ours	**3.63**	**2.02**×	**2.74**	1.66×	**3.13**	**1.84**×	**3.98**	**1.88**×	**2.98**	**1.78**×	**4.74**	**2.42**×	**1.93**×‘
Codellama-13B-Instruct													
Lookahead (Fu et al., 2024)	2.02	1.63×	1.61	1.39×	1.46	1.22×	1.45	1.17×	1.57	1.32×	1.18	1.05×	1.31×
Medusa (Cai et al., 2023)	1.74	2.04×	1.89	**2.09**×	1.42	1.65×	1.60	1.65×	1.47	1.58×	1.68	1.75×	1.81×
REST (He et al., 2024)	1.63	1.23×	1.91	1.48×	1.99	1.57×	1.80	1.51×	1.73	1.41×	1.46	1.13×	1.39×
AdaDecode (Wei et al., 2025)	2.40	1.75×	2.20	1.52×	2.50	1.60×	2.70	1.58×	2.35	1.48×	2.60	1.65×	1.52×
Ours	**3.49**	**2.20**×	**2.69**	1.85×	**2.97**	**1.97**×	**3.82**	**3.25**×	**3.00**	**2.13**×	**4.02**	**2.56**×	**2.33**×

Table 1: Performance comparison of SpecBound against other self-drafting methods across multiple base models and diverse text generation tasks on Spec-Bench. We report compression rate (CR, average accepted tokens per draft round) and wall-time speedup (SD) relative to autoregressive decoding.

shown in Table 1, our method achieves strong performance across diverse tasks, yielding the best overall speedup compared to existing approaches. Notably, it excels on tasks such as translation, where it attains speedup up to $2.94\times$, with consistently competitive results across some other tasks such as multi-turn dialogue. This performance variation likely stems from task-dependent early-exit behaviors: the inference hyperparameters selected to maximize overall speedup may not be optimal for every individual task. In practice, task-specific tuning of these parameters could further improve acceleration efficiency.

SpecBound enables better speedup through high-quality drafting. Our method generates draft tokens using multiple transformer layers, incurring higher computational cost per draft step compared to lightweight alternatives (e.g., linear heads or single-layer transformers). However, this investment yields significantly more accurate drafts — as evidenced by our consistently highest acceptance rate (CR ≥ 3.0 across all tasks), which enables deeper and more efficient multi-token speculation. Crucially, because each draft token is computed only up to a shallow or intermediate layer, the per-token draft cost remains substantially lower than that of a complete forward pass through the base LLM. As a result, the time saved from fewer speculative rejections more than compensates for the additional draft generation cost, resulting in superior overall speedup. This presents an alternative design principle: while most prior work focuses on minimizing draft latency, we prioritize draft quality to maximize end-to-end throughput.

Larger models benefit more from early exit despite similar draft quality. Interestingly, SpecBound consistently achieves higher speedup on 13B models than their 7B counterparts—even when compression rates (CR) are comparable. This suggests that deeper architectures offer greater acceleration potential through early exit, as a larger fraction of their deep layers are redundant for most easy tokens, enabling early-exited drafts to be verified in parallel over more layers. As a result, wall-

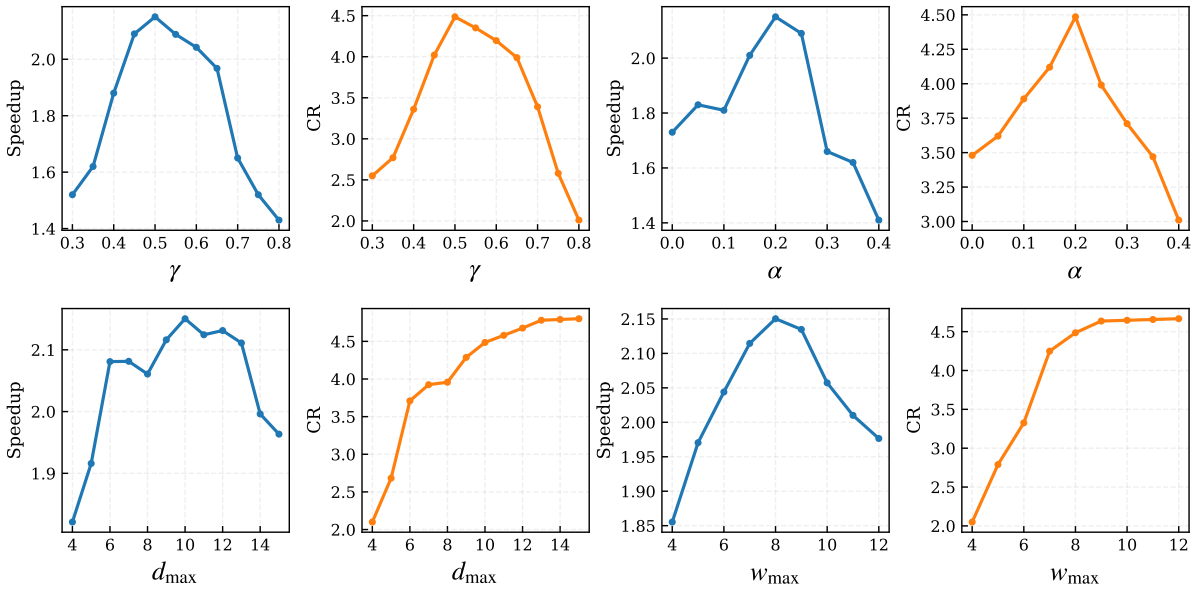

Figure 3: Hyperparameter Sensitivity Analysis on Vicuna-7B: wall-time speedup (blue) and compression rate (CR, orange) under varying exit threshold γ, annealing coefficient α, depth bound $d_{\max}$, and width bound $w_{\max}$.

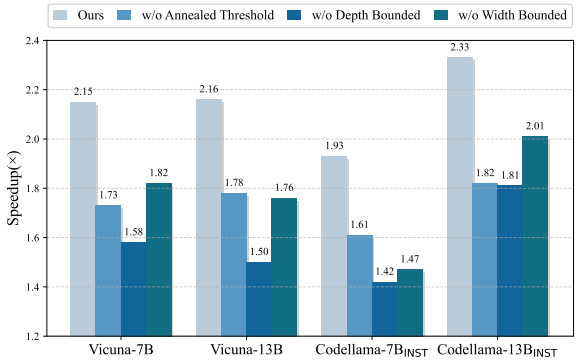

Figure 4: Ablation study of SpecBound components on wall-time speedup across different base models, comparing the full method against variants that removing the Annealed Confidence Threshold early exit mechanism, the depth bound $d_{\max}$, or the width bound $w_{\max}$.

time savings per accepted token are amplified, leading to higher net speedup even when CR plateaus.

Performance Robustness Across Model Domains Furthermore, as a chat-optimized model, Vicuna achieves consistently higher compression rates (CR) across all Spec-Bench tasks, which are predominantly natural-language oriented. In contrast, CodeLlama-Instruct, fine-tuned primarily on code datasets, exhibits relatively lower CR on these general-language tasks, underscoring the impact of domain alignment between the base model and the evaluation benchmark. Nevertheless, SpecBound consistently delivers the highest speedup across both model families, demonstrating its robustness to architectural and domain differences without any modification to the base LLM parameters.

3.3 Ablation Study

To demonstrate the robustness of our method and identify its effective hyperparameter range, we conduct a series of ablation studies on Vicuna-7B using the Spec-Bench benchmark. In each experiment, we vary one parameter—exit threshold γ, annealing coefficient α, depth bound $d_{\max}$, or width bound $w_{\max}$—while fixing all others at their optimal values. The resulting wall-time speedup (blue) and compression rate (CR, orange) are shown in Figure 3. We further conduct an ablation study to assess the contribution of each component in SpecBound. The experiment evaluates the full method against three variants—each removing one key component: the Annealed Confidence Threshold, the depth bound , or the width bound—across multiple base models on the Spec-Bench benchmark, as shown in Figure 4.

Impact of Annealed Confidence Threshold Parameters Figure 3 (top) shows that both the exit threshold γ and annealing coefficient α critically affect performance. A low γ may admit spurious shallow token exits, while a high γ suppresses valid token exits. Similar trade-offs arise with the annealing strength: a small α fails to deal with spurious overconfidence, whereas a large α over-smooths

Method	Math	Multi.	QA	RAG	Sum.	Trans.	Overall
greedy decoding	**1.89**×	**1.65**×	**2.16**×	**2.31**×	**1.95**×	**2.94**×	**2.15**×
temperature sampling (T=0.3)	1.82×	1.56×	2.01×	2.20×	1.86×	2.86×	2.08×
temperature sampling (T=1.0)	1.71×	1.44×	1.92×	2.08×	1.73×	2.63×	1.88×
top-p sampling (p=0.9)	1.78×	1.48×	1.99×	2.15×	1.78×	2.71×	1.94×

Table 2: Speedup performance of SpecBound (Vicuna-7B) under different decoding strategies on different tasks of Spec-Bench. We choose greedy docoding, temperature sampling and top-p sampling with different settings.

the distribution, hindering early exit. Generally, the best speedup is achieved with a moderate γ and a small α, which mildly adjusts the softmax to balance reliability and efficiency.

Impact of Depth and Width Bounds Figure 3 (bottom) shows that speedup initially rises with $d_{\max}$ and $w_{\max}$, as more draft tokens can be generated and verified per speculation step. However, excessive $d_{\max}$ reduce the proportion of computation that can benefit from parallel shallow-depth processing. Similarly, an overly large $w_{\max}$ cause the errors to accumulate. These effects diminish end-to-end throughput despite increased the length of accepted tokens. The optimal balance is achieved at $d_{\max} = 10$ and $w_{\max} = 8$, maximizing speculative efficiency without frequent interruption.

Component Contributions in Ablation Study Figure 4 shows that removing any component of SpecBound leads to performance degradation, confirming their collective contribution to speedup. Removing the depth bound ($d_{\max}$) causes the largest drop, highlighting its critical role in mitigating difficult tokens by capping speculative depth and preserving parallelism across deeper layers. Removing the width bound ($w_{\max}$) results in a smaller decline: when removed, the system may still benefit from $d_{\max}$, as longer draft sequences tend to push exit layers deeper due to increasing model uncertainty, naturally triggering early verification. Nevertheless, $w_{\max}$ remains essential as the primary gate against excessively long shallow draft sequences, ensuring reliability without sacrificing efficiency.

Effect of Sampling-Based Decoding Strategies To further evaluate the versatility of SpecBound, we conduct experiments on Spec-Bench using Vicuna-7B to assess its performance under various sampling-based decoding strategies, including temperature sampling ($T = 0.3$ and 1.0) and top-p sampling ($p = 0.9$). As summarized in Table 2, SpecBound consistently delivers significant wall-time speedup across all tested regimes, ranging from $1.88\times$ to $2.08\times$, even as sampling randomness increases. While the peak acceleration of $2.15\times$ is observed under greedy decoding, the performance remains robust across diverse stochastic settings. The slight moderation in speedup at higher temperatures is expected, as increased token entropy inherently poses challenges for speculative acceptance. These results demonstrate that SpecBound is not restricted to greedy search and can effectively accelerate various practical decoding configurations in real-world applications.

Metric	AR (Base)	Ours	Improv.
TTFT (ms)	20.8	21.1	-
Per-token latency (ms)	17.5	8.8	**1.98**×
Throughput (tokens/s)	57.3	123.1	**2.15**×
End-to-End Latency (s)	3.31	1.89	**1.75**×

Table 3: Serving latency metrics comparison on Vicuna-7B using a single NVIDIA H800 GPU. "AR" denotes standard autoregressive decoding.

Detailed Serving Latency Analysis To provide a more actionable view of SpecBound's practical utility, we evaluate absolute serving latency metrics using Vicuna-7B on SpecBench with NVIDIA H800 GPUs, maintaining a fixed maximum length of 1024 and batch size of 1. As summarized in Table 3, SpecBound achieves significant improvements in per-token streaming latency and throughput over the autoregressive baseline. Notably, the prefill phase remains identical to the base model, resulting in a Time to First Token (TTFT) nearly equal to the baseline. The marginal difference is attributed to minimal one-time operations such as variable initialization and the allocation of the small, fixed-size cache table ($L \times w_{\max}$). Furthermore, because the cache management involves minimal hardware operations and the table is reused for each draft window rather than growing with the input, SpecBound is not negatively impacted by long-context tasks in terms of memory overhead.

4 Related Work

Recent advancements in accelerating Large Language Models (LLMs) stem from the draft-then-verify paradigm, first introduced by Blockwise Decoding (Stern et al., 2018), which later inspired Speculative Decoding (Leviathan et al., 2023) and Speculative Sampling (Chen et al., 2023). Building upon speculative decoding with an independent draft model, recent works explore self-draft strategies that leverage the base LLM itself for speculation. These Self-draft methods can be broadly categorized into three paradigms: (1) fine-tuning (2) layer skip and (3) early exit.

4.1 Fine-tuning Methods

Some works fine-tune the base LLM on non-autoregressive objectives to endow it with parallel token prediction capabilities (Lin et al., 2025; Yi et al., 2024). However, such parameter updates not only require substantial computational resources for training but also risk degrading the model's original generalization performance on diverse downstream tasks. In contrast, our approach requires no fine-tuning and keeps all parameters frozen, ensuring that the model's inherent capabilities remain intact while significantly reducing the barrier to deployment. This plug-and-play nature allows SpecBound to be seamlessly integrated with existing pre-trained LLMs without any additional training overhead or performance trade-offs.

4.2 Layer Skip Methods

Another line of work accelerates drafting by using only a subset of the LLM's layers—similar to layer pruning (Kim et al., 2024; Zeng et al., 2023; Yang et al., 2024; Men et al., 2025). For example, (Zhang et al., 2024) adaptively skips intermediate layers during draft generation. However, since skipped layers are never executed for draft tokens, their hidden states remain incomplete, complicating rigorous verification and risking output correctness. Furthermore, such skipping often leads to a mismatch in the KV cache across layers. In contrast, our method ensures every output token passes through all base LLM layers during verification, preserving distributional fidelity. By maintaining a continuous and complete computation flow, SpecBound eliminates the risks associated with missing intermediate representations, thereby guaranteeing that the accelerated output is mathematically identical to that of the original model.

4.3 Early Exit Methods

To preserve per-layer computation, early exit methods adopt the strategy of producing draft tokens by executing only the initial few layers of the base LLM, and then feeding these drafts into the subsequent decoding step for full-depth generation (Bolukbasi et al., 2017; Li et al., 2021; Kong et al., 2022; Varshney et al., 2023). Notably, AdaDecode (Wei et al., 2025) also adopts an early-exit plus verification strategy similar in spirit to ours. However, its draft generation relies on a simplistic early-exit criterion and imposes no bound on speculation depth, which limits its acceleration potential. In contrast, SpecBound introduces a more refined early-exit mechanism to improve draft accuracy and Bounded Speculation with Cached States Algorithm to maximize parallel efficiency—enabling higher speedup while preserving exact output equivalence.

5 Conclusion

We present SpecBound, a self-draft speculative decoding method that explicitly addresses the heterogeneity in early-exit difficulty across tokens during decoding. To mitigate spurious high-confidence predictions in shallow layers, we introduce a carefully designed annealed temperature-sampling early exit judge mechanism. To prevent redundant computation on difficult tokens and error propagation in long shallow draft chains, we propose a cache-based draft-verification algorithm with adaptive depth and width bounds. Notably, SpecBound achieves strong and consistent speedup across diverse base models and text generation tasks, while ensuring lossless acceleration without any modification to the base LLM parameters.

Limitations

Our approach, while avoiding the costly training of a separate draft model, still requires lightweight trainable heads at intermediate layers to enable accurate early exiting, and is therefore not fully training-free. Future work could explore mechanisms for reliable shallow exits without any parameter updates. Moreover, the method involves several hyperparameters whose optimal settings currently rely on empirical tuning. A promising direction is to develop adaptive or tuning-free strategies, e.g., dynamically adjusting speculation bounds based on online token-level uncertainty, to enhance robustness and ease of deployment.

Acknowledgements

We gratefully acknowledge all the reviewers for their valuable comments and suggestions. This work was supported by the Natural Science Foundation of Beijing, China (Grant No. L257006).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. 2017. Adaptive neural networks for efficient inference. In *International conference on machine learning*, pages 527–536. PMLR.

Mengyu Bu and Yang Feng. 2026. Language on demand, knowledge at core: Composing llms with encoder-decoder translation models for extensible multilinguality. *arXiv preprint arXiv:2603.17512*.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, and Tri Dao. 2023. Medusa: Simple framework for accelerating llm generation with multiple decoding heads.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, and 1 others. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. 2023. Dola: Decoding by contrasting layers improves factuality in large language models. *arXiv preprint arXiv:2309.03883*.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, and 1 others. 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.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. 2024. Break the sequential dependency of llm inference using lookahead decoding. *arXiv preprint arXiv:2402.02057*.

Xinwei Geng, Xiaocheng Feng, and Bing Qin. 2021. Learning to rewrite for non-autoregressive neural machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3297–3308.

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

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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.

Bo-Kyeong Kim, Geonmin Kim, Tae-Ho Kim, Thibault Castells, Shinkook Choi, Junho Shin, and Hyoung-Kyu Song. 2024. Shortened llama: A simple depth pruning for large language models. *arXiv preprint arXiv:2402.02834*, 11:1.

Jun Kong, Jin Wang, Liang-Chih Yu, and Xuejie Zhang. 2022. Accelerating inference for pretrained language models by unified multi-perspective early exiting. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4677–4686.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Xiaonan Li, Yunfan Shao, Tianxiang Sun, Hang Yan, Xipeng Qiu, and Xuan-Jing Huang. 2021. Accelerating bert inference for sequence labeling via early-exit. In *Proceedings of the 59th annual meeting of the Association for Computational Linguistics and the 11th international joint conference on natural language processing (volume 1: long papers)*, pages 189–199.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*.

Feng Lin, Hanling Yi, Yifan Yang, Hongbin Li, Xiaotian Yu, Guangming Lu, and Rong Xiao. 2025. Bita: Bi-directional tuning for lossless acceleration in large language models. *Expert Systems with Applications*, 279:127305.

Fangcheng Liu, Yehui Tang, Zhenhua Liu, Yunsheng Ni, Kai Han, and Yunhe Wang. 2024. Kangaroo: Lossless self-speculative decoding via double early exiting. *arXiv preprint arXiv:2404.18911*.

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2025. Shortgpt: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20192–20204.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025. Large language diffusion models. *arXiv preprint arXiv:2502.09992*.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, and 1 others. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Neeraj Varshney, Agneet Chatterjee, Mihir Parmar, and Chitta Baral. 2023. Accelerating llm inference by enabling intermediate layer decoding. *CoRR*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Zhepei Wei, Wei-Lin Chen, Xinyu Zhu, and Yu Meng. 2025. Adadecode: Accelerating llm decoding with adaptive layer parallelism. *arXiv preprint arXiv:2506.03700*.

Zhuofan Wen, Shangtong Gui, and Yang Feng. 2024. Speculative decoding with ctc-based draft model for llm inference acceleration. *Advances in Neural Information Processing Systems*, 37:92082–92100.

Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. 2023. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3909–3925.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. 2025. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. 2024. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. *arXiv preprint arXiv:2401.07851*.

Yifei Yang, Zouying Cao, and Hai Zhao. 2024. Laco: Large language model pruning via layer collapse. *arXiv preprint arXiv:2402.11187*.

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

Dewen Zeng, Nan Du, Tao Wang, Yuanzhong Xu, Tao Lei, Zhifeng Chen, and Claire Cui. 2023. Learning to skip for language modeling. *arXiv preprint arXiv:2311.15436*.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. 2024. 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.

Lingzhe Zhang, Liancheng Fang, Chiming Duan, Minghua He, Leyi Pan, Pei Xiao, Shiyu Huang, Yunpeng Zhai, Xuming Hu, Philip S Yu, and 1 others. 2025. A survey on parallel text generation: From parallel decoding to diffusion language models. *arXiv preprint arXiv:2508.08712*.

Jingze Zhu, Yongliang Wu, Wenbo Zhu, Jiawang Cao, Yanqiang Zheng, Jiawei Chen, Xu Yang, Bernt Schiele, Jonas Fischer, and Xinting Hu. 2025. Layercake: Token-aware contrastive decoding within large language model layers. *arXiv preprint arXiv:2507.04404*.

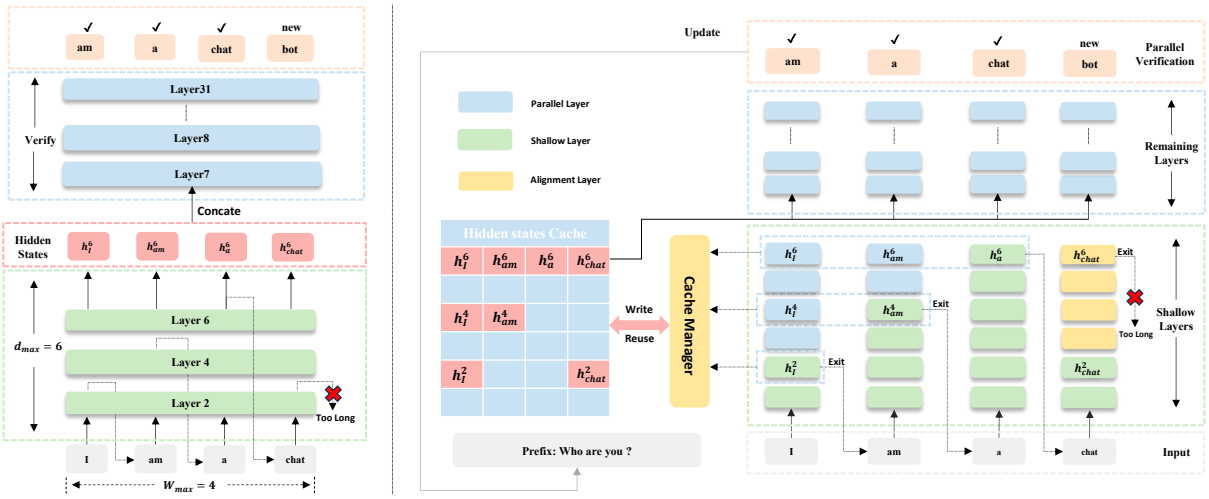

Figure 5: **Left:** Overview of the BSCS algorithm under the setting $d_{\max} = 6$ and $w_{\max} = 4$. When token "chat" successfully exits at layer 2 and the total draft length reaches the width bound $w_{\max}$, speculation is terminated. All hidden states of previously generated tokens ("I", "am", "a", "chat") are concatenated and passed to the remaining layers for parallel verification. **Right:** Detailed illustration of per-token layer-wise computation and hidden-state cache management.

A Illustration of Long Shallow Chains

Given a decoding history prefix "Who are you?", the base model early exits token "I" at layer 2, token "am" at layer 4, token "a" at layer 6, and token "chat" at layer 2 in four consecutive steps. After the fourth token, the draft length reaches the width bound $w_{\max} = 4$, triggering an early verification despite all tokens having successfully exited at shallow layers. At this point, BSCS terminates the speculation phase and initiates parallel verification of all draft tokens. Crucially, because these tokens exited at different shallow layers, their hidden states are first aligned to the deepest exit layer among them (layer 6 in this case) by running the missing intermediate layers in parallel. The aligned representations are then jointly processed through the remaining layers (layer 7 to L) in a single forward pass, ensuring consistent context for batched verification. This alignment step prevents error propagation from shallow, potentially unreliable predictions while preserving the efficiency of multi-token speculation.

B Baseline Implementation Details

For a fair comparison across all base models and tasks in Spec-Bench, we retrain the draft components of AdaDecode (Wei et al., 2025) and Kangaroo (Liu et al., 2024) on the same 68K-shareGPT dataset used for SpecBound. Although public weights for these methods are available for certain model configurations (e.g., Vicuna-7B), they do not cover the full range of our evaluation settings(base models or benchmarks). All other baselines (Lookahead (Fu et al., 2024), Medusa (Cai et al., 2023), REST (He et al., 2024), SPACE (Yi et al., 2024)) are evaluated using their officially released weights and the standard Spec-Bench(Xia et al., 2024) pipeline without modification. All results are measured on a single NVIDIA H800 GPU, with speedup reported relative to autoregressive decoding using the same base LLM.

C Training Overhead Details

As detailed in Section 3.1, the training process is highly efficient, requiring only 20 epochs on 68K multi-turn conversations from the ShareGPT dataset. This setup is completed in approximately two hours using four NVIDIA H800 GPUs.

A key factor in this efficiency is the "feature-caching" strategy employed during data generation. By storing the intermediate hidden states of the base LLM as it decodes the training inputs, the base model does not need to be re-run during the iterative training of the LM heads. Instead, we perform parallel supervision only on a set of lightweight linear layers, significantly reducing the computational burden. Once trained, these task-agnostic weights provide a permanent reduction in inference-time latency and resource consumption across various downstream applications.

Device	Math	Multi-turn	QA	RAG	Summary	Translation	Overall
H800	1.89×	1.65×	2.16×	2.31×	1.95×	2.94×	**2.15**×
RTX 3090	1.81×	1.58×	2.12×	2.13×	1.92×	2.69×	**2.01**×

Table 4: Speedup performance of SpecBound (Vicuna-7B) on different GPU devices across various Spec-Bench tasks. The evaluation are conducted on H800 and RTX 3090 with all other settings are the same.

D Empirical Robustness Across Different Hardware Architectures

To validate the robustness of SpecBound across diverse hardware, we additionally evaluated our method on consumer-grade NVIDIA GeForce RTX 3090 GPUs (24GB), using Vicuna-7B as the base model. The RTX 3090 was selected for its more constrained memory subsystem—specifically its lower memory bandwidth and higher latency compared to the H800—to verify SpecBound's effectiveness under stricter hardware limitations. As shown in table 4, while the overall speedup slightly decreases from $2.15\times$ to $2.01\times$ when switching to the RTX 3090, SpecBound remains consistently effective across all tasks. These results demonstrate the method's practical robustness and its suitability for diverse deployment scenarios, even on hardware with significant resource constraints.

E Robustness and Break-even Analysis

To address the potential concerns regarding the robustness in low-acceptance regimes (e.g., domain shift or highly ambiguous sequences), we provide a comprehensive analysis of its performance lower bound and empirical speedup distribution.

Theoretical Worst-Case Analysis Theoretically, the worst-case scenario for SpecBound occurs when the model speculates the maximum width $w_{\max}$ at depth $d_{\max}$, but all draft tokens are rejected during verification. In this case, only one token is successfully decoded in the cycle. Based on the speedup model in Eq. 5, as the per-token acceptance rate $a \to 0$ and speculation width $w \to w_{\max}$, the speedup ratio SD can theoretically fall below 1.0. However, SpecBound mitigates this "wasted" computation by reusing cached hidden states $h^{(d_{\max})}$ for the subsequent verification pass. Since the cost of a few early-layer passes is fractional compared to a full-model forward pass, the efficiency gap between consumption and gain is significantly narrowed, ensuring that the "investment" in drafting usually yields a net positive return.

Speedup Range	Number	Percentage
$3.54\times - 4.16\times$ (Max)	16	3.33%
$2.93\times - 3.54\times$	48	10.00%
$2.31\times - 2.93\times$	103	21.46%
$1.69\times - 2.31\times$	197	41.04%
$1.07\times$ (Min) $- 1.69\times$	116	24.17%

Table 5: Empirical speedup distribution of SpecBound on Vicuna-7B across 480 prompts from SpecBench.

Empirical Distribution across Prompts To empirically validate the robustness, we analyze the per-prompt speedup distribution for Vicuna-7B on SpecBench (480 prompts). As summarized in Table 5, SpecBound consistently maintains a speedup above $1.0\times$ even in its tail cases. Specifically, 24.17% of prompts fall into the minimum speedup range ($1.07\times - 1.69\times$), while over 34% of prompts achieve a speedup exceeding $2.93\times$. The fact that the entire distribution remains above the break-even point ($SD > 1.0$) underscores the practical reliability of our adaptive bounded speculation, even when encountering difficult tokens or varying context lengths.