ICASSP 2026 - 2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 979-8-3315-6701-9/26/$31.00 ©2026 IEEE | DOI: 10.1109/ICASSP55912.2026.11463423

ACCEPTANCE-GUIDED ADAPTIVE SPECULATIVE DECODING FOR EFFICIENT LARGE LANGUAGE MODEL INFERENCE

Weiyang Sun[1,2], *Zifeng Zhu*[1,2], *Yixian Dai*[1,2], *Binghui Guo*[2], *Yifan Sun*[3⋆], *Feng Zhou*[3⋆]

[1]Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing
[2]School of Artificial Intelligence, Beihang University, China
[3]Center for Applied Statistics and School of Statistics, Renmin University of China, China

ABSTRACT

As Large Language Models grow, autoregressive decoding leads to severe inference latency. Speculative decoding mitigates this issue by allowing a lightweight draft model to generate multiple candidate tokens, which are then fed into the target model for parallel verification. A key hyperparameter is the draft length K: a larger K may waste computation due to early rejection, while a smaller K underutilizes the draft model's capacity. Prior work fixes K, which is suboptimal across contexts. We propose Acceptance-Guided Adaptive Speculative Decoding, which retains an upper bound but dynamically selects an effective length via an acceptance head attached to the draft model. The head predicts per-token acceptance probabilities and triggers early exit when rejection risk exceeds a principled threshold derived from a runtime cost model. This reduces wasted verification and exploits confident drafts. Experiments on four tasks under two temperature settings, evaluated on two series of LLMs, show reduced latency. The code is available at https://github.com/sunnywyang/ACCEPTANCE-GUIDED-ADAPTIVE-SPECULATIVE-DECODING.

***Index Terms*—** Efficient LLM Inference, Adaptive Speculative Decoding, Dynamic Draft Length

1. INTRODUCTION

As the scale of Large Language Models (LLMs) [1] and sequence length increase, the frequent reading and writing of massive parameters and the sequential nature of autoregressive decoding lead to low GPU resource utilization and low inference efficiency of LLMs [2]. Speculative decoding (SD) [3] addresses this challenge by first generating multiple candidate tokens with a smaller draft model and then verifying them in parallel with the target LLM. However, SD performance is constrained by an important hyperparameter: the draft length K, i.e., the number of candidate tokens the target model must verify per round.

Currently, most methods [4, 5] use a fixed K value. Leviathan et al. [3] made an independent and identically distributed assumption about the acceptance probability of candidate tokens and theoretically derived the optimal choice of K. Subsequent methods, such as EAGLE [6], utilized a smaller model of the LLM's own hidden states to significantly accelerate inference while maintaining output quality. EAGLE-2 [7] introduced a context-aware dynamic draft tree based on EAGLE. However, they all use a fixed draft length K, which leads to performance variability of SD across different tasks [8]. A lower value of K may fail to fully exploit the capacity of the draft model to accelerate decoding, whereas a higher value of K may result in many draft tokens being discarded, consequently leading to wasted resources in the draft stage.

To overcome this limitation, our work proposes an adaptive SD framework that adaptively adjusts the draft length K during the draft stage. Specifically, we incorporate two modules in the draft model: the decoder and the acceptance head. The decoder is responsible for generating candidate tokens, while the acceptance head predicts the rejection probability of these generated candidates. When the rejection probability exceeds a threshold (derived from runtime statistics), the draft phase is terminated early, and the process immediately enters the verification phase. This mechanism reduces verification overhead for low-confidence candidates while exploiting the draft model's full generation capacity under high confidence, thereby balancing speculative depth and verification efficiency.

Unlike prior work that treats K as a fixed hyperparameter, our approach sets a safe upper bound $K_{\max}$ but allows the acceptance head—operating on the draft model's hidden states—to predict per-token acceptance probabilities $\hat{P}_{\text{acc}}$. These predictions enable an online early-stopping mechanism, dynamically determining the effective draft length K_{eff} at runtime. Concretely, during draft generation, the acceptance head assesses the rejection risk of each candidate token and terminates the speculation for the current round once the estimated rejection probability exceeds a principled threshold. This threshold is derived from a lightweight runtime cost model and explicitly balances the draft-stage computational overhead against the verification cost. It is worth noting that the acceptance head is lightweight, requires no modifications to the target LLM, and integrates seamlessly into existing SD pipelines. We validate our method across multiple generation tasks and model pairings, demonstrating improved inference speed over other state-of-the-art methods while maintaining comparable output quality.

2. METHOD

In this section, we first introduce the fundamental concepts and background of SD. We then present our proposed method, describing how the draft length is dynamically adjusted to improve overall decoding efficiency. Finally, we provide a detailed description of the core component of our approach: the acceptance head.

2.1. Speculative Decoding

SD is a *draft-and-verify* paradigm designed to accelerate autoregressive generation in Transformer-based large language models [9, 10]. In standard decoding, given a prefix $x_1, \ldots, x_t$, the target model M_q predicts the next token as

$$x_{t+1} \sim q_{t+1} = M_q(x \mid x_{<t+1}), \tag{1}$$

⋆ Corresponding authors.

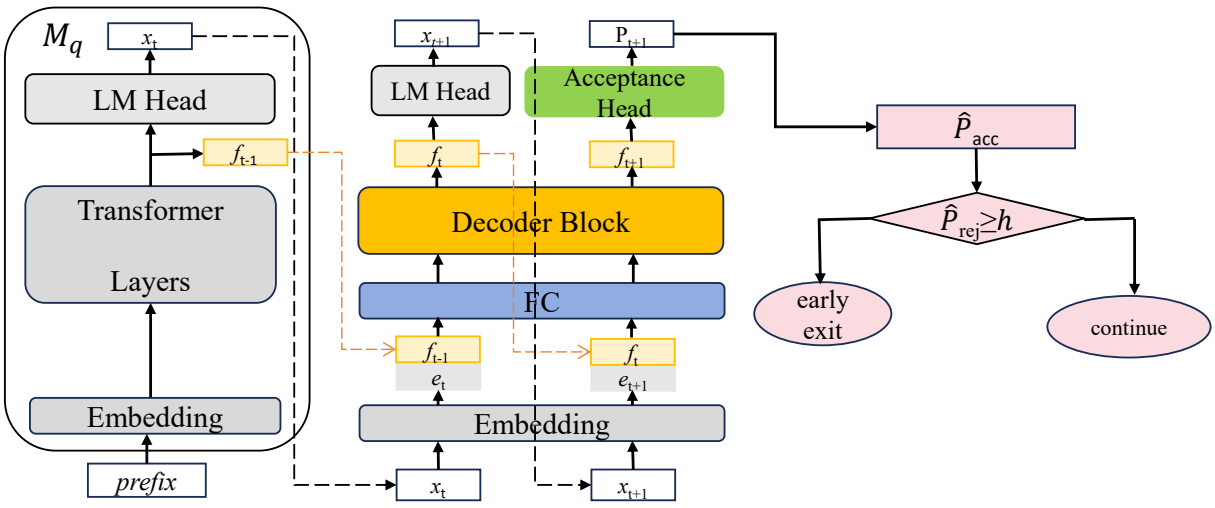

Fig. 1. Overview of Acceptance-Guided Adaptive Speculative Decoding. Given a prefix, the draft model generates K candidate tokens per speculative round. The acceptance head, attached to the decoder block, computes per-token acceptance probabilities $\hat{P}_{\text{acc}}$ from draft hidden states f_t. The system monitors the rejection probability $\hat{P}_{\text{rej}}$ and triggers an early exit to the target LLM M_q when it exceeds threshold h.

where q_{t+1} denotes the conditional probability distribution computed by the target model. This strictly sequential process significantly limits inference efficiency.

To address this, speculative decoding introduces an auxiliary draft model M_p to efficiently propose multiple future candidates. Formally, at decoding step t the draft model M_p sequentially computes K conditional token distributions and samples:

$$p_i = M_p(x \mid x_{\leq t}, \tilde{x}_{<i}), \qquad \tilde{x}_i \sim p_i, \qquad i = 1, \ldots, K, \tag{2}$$

where p_i denotes the conditional distribution over the next token given the prefix $x_{\leq t}$ and the previously drafted tokens $\tilde{x}_{<i}$, and $\tilde{x}_i$ is the i-th drafted token.

These candidates are then verified by the target model M_q, which computes the conditional distributions in parallel [3][11]:

$$q_i = M_q(x \mid x_{\leq t}, \tilde{x}_{<i}), \quad i = 1, \ldots, K+1. \tag{3}$$

Each draft token $\tilde{x}_i$ is accepted with probability: $\min\left\{1, \frac{q_i(\tilde{x}_i)}{p_i(\tilde{x}_i)}\right\}$. The first rejected token is replaced by a token sampled from a modified distribution and all subsequent draft tokens are discarded. If K draft tokens are all accepted, an extra token is sampled from q_{K+1}. These accepted tokens are appended to the prefix, and this draft-and-verify process is repeated until the termination condition is met.

The hyperparameter K, i.e., the draft length per round, is central to SD [12]. A lower value of K may fail to fully exploit the capacity of the draft model to accelerate decoding, whereas a higher value of K may result in many draft tokens being discarded, consequently leading to wasted resources in the draft stage. Thus, K directly controls the overall acceleration. In this work, we focus on making K *adaptive*, to achieve better acceleration performance.

2.2. Adaptive Speculative Decoding

We propose an adaptive SD scheme that: (1) employs a lightweight **acceptance head** to dynamically determine the draft length K during runtime. The acceptance head estimates per-token acceptance probabilities and triggers early-exit based on a principled cost model, thereby adaptively trading off draft quality against verification cost and determining the effective draft length K_{eff} online; (2) organizes draft outputs into a *context-aware dynamic tree* of candidate continuations (as proposed in EAGLE-2 [7]), which preserves structured multi-branch information and supports parallel verification.

The context-aware dynamic draft tree's root corresponds to the prefix; each branch represents a candidate continuation. The tree operates in three stages: (i) expand high-value nodes in parallel; (ii) rerank and prune to retain top-n branches; and (iii) flatten retained nodes and perform parallel verification with adjusted attention masks. The acceptance head supplies per-token scores aggregated at branch level; if a branch's rejection estimate exceeds a runtime threshold, the draft round stops.

Our draft model consists of the decoder and the acceptance head that predicts token-level acceptance from draft hidden states. The design of the acceptance head is inspired by adaptive-depth ideas [13], enabling expressive draft generation while supporting real-time confidence-based early-exit decisions with negligible overhead—thus achieving dynamic K control adapted to instantaneous draft confidence, as illustrated in Fig. 1.

At each decoding step t, the draft model first concatenates the current hidden state f_t with the corresponding token embedding e_{t+1}, and feeds it through a fully connected layer and the decoder block to obtain the next hidden state:

$$f_{t+1} = \text{Decoder}\Big(\text{FC}(\text{Cat}(f_t, e_{t+1}))\Big). \tag{4}$$

The acceptance head then predicts the per-token acceptance probability from the draft hidden state:

$$\hat{P}_{\text{acc},t+1} = \text{Acceptance Head}(f_{t+1}). \tag{5}$$

The rejection probability for branch b is given by

$$P_{\text{rej}}^{(b)}(t) = 1 - \prod_{i=1}^{t} \hat{P}_{\text{acc},\,i}^{(b)}, \quad \forall b \in \mathcal{B}, \tag{6}$$

where $\mathcal{B}$ representing the set of active branches in the context-aware dynamic draft tree. The effective draft length K_{eff} for each branch is

dynamically determined based on rejection probability:

$$K_{\text{eff}}^{(b)} = \min\{i \mid P_{\text{rej}}^{(b)}(i) > h\}, \quad K_{\text{eff}}^{(b)} \leq K_{\max}, \tag{7}$$

where h is a threshold.

During drafting, all active branches expand in parallel while the acceptance head updates per-branch rejection probabilities and prunes low-confidence candidates. If any branch's rejection probability exceeds the threshold h, drafting halts and the current candidates are sent to the target LLM for parallel verification. Accepted tokens are appended to the output; rejected tokens trigger resampling or sequential backtracking to restore correctness. This draft–and-verify loop repeats until generation terminates. By combining context-aware branching with real-time acceptance scoring, the system adaptively determines K_{eff} per branch and balances speculative depth against verification overhead.

2.3. Acceptance Head: Threshold h and Training

The acceptance head is designed as a lightweight module to provide real-time feedback on the reliability of tokens generated by the draft model. Specifically, it employs a compact ResNet architecture that processes the hidden state f_{t+1} of the draft model and outputs acceptance logits, which are then squashed through a sigmoid function to obtain a probability estimate:

$$\hat{P}_{\text{acc},t+1} = \sigma(\text{ResNet}(f_{t+1})), \tag{8}$$

where $\sigma(\cdot)$ is the sigmoid function. To ensure minimal computational overhead, the ResNet consists of 1–2 layers with SiLU activation, balancing expressive capacity with efficiency. The resulting probability $\hat{P}_{\text{acc}}$ serves as a dynamic quality signal during inference, guiding whether additional tokens should be drafted or whether verification by the target model should be triggered immediately.

The training set is produced in two stages. First, we sample reference continuations from the target model and record its next-token probabilities $q(\cdot)$. Second, we run the draft model on the same prefixes to sample candidate tokens, collect draft probabilities $p(\cdot)$ and the final hidden embeddings f_i. For each draft token $\tilde{x}_i$ we set the supervision target as the acceptance probability:

$$P_i^{\text{true}} \;=\; \min\left(1,\, \frac{q(\tilde{x}_i)}{p(\tilde{x}_i)}\right). \tag{9}$$

The distribution shift between the draft model and the target model may cause certain biases [14] in the acceptance head, such as over-confidence in acceptance. To mitigate this issue and construct more informative training data for the acceptance head, we adopt a token mixing strategy [15]: for a given target model response sequence, we generate a corresponding draft sequence from the draft model, and then create a mixed sequence by randomly selecting half of the tokens from the target response and replacing the remaining half with tokens from the draft sequence.

Due to the inherent imbalance between accepted and rejected tokens, we employ a weighted binary cross-entropy objective:

$$\mathcal{L}_{\text{acc}} \;=\; -w_{\text{acc}}\, P_i^{\text{true}} \log \hat{P}_i \;-\; w_{\text{rej}} \left(1 - P_i^{\text{true}}\right) \log\!\left(1 - \hat{P}_i\right). \tag{10}$$

Inspired by [13], the threshold h is derived from a latency-cost model [16]. Let $T_{\text{d}}(k)$ be the time to generate k draft tokens, T_{v} be the time to verify the current candidate set with the target model, and T_{r} be the expected extra cost (resampling + additional verification) when a rejection occurs. The estimated probability that at least one of the next k tokens will be rejected is P_{rej}.

Continue drafting only if the expected time of continuing drafting is less than immediate verification:

$$T_{\text{d}}(k) + P_{\text{rej}}(k) T_{\text{r}} < T_{\text{v}}. \tag{11}$$

Rearranging gives a cost threshold for $P_{\text{rej}}(k)$:

$$P_{\text{rej}}(k) < h = \frac{T_{\text{v}} - T_{\text{d}}(k)}{T_{\text{r}}}. \tag{12}$$

This ensures that speculation proceeds only when it is cost-effective.

3. EXPERIMENTS

In this section, we evaluate the effectiveness of our method by comparing it to several state-of-the-art SD baselines. We first describe experimental settings and implementation details, then report main results on multiple generation tasks under two temperature regimes.

3.1. Experimental Setup

We evaluate our Acceptance-Guided Adaptive Speculative Decoding on four benchmarks: multi-turn dialogue (**MT-bench** [17]), question answering (**Natural Questions** [18]), document summarization (**CNN/Daily Mail** [19]), and instruction following (**Alpaca** [20]).

For comparison we evaluate several recent and representative SD baselines—**Medusa** [21], **EAGLE** [6], **EAGLE-2** [7], and **Hydra** [22]. Experiments are performed under two decoding temperature regimes: temperature = 0 and temperature = 1. Temperature = 0 is greedy decoding; temperature = 1 is a stochastic decoding mode that increases output diversity via logits processing.

We conduct experiments using four target LLMs: **Vicuna-7B/13B** [23] and **Llama2-chat7B/13B** [24]. Target LLMs are fixed during training, with only the draft model being trained. The acceptance head is implemented as a lightweight one-layer ResNet with SiLU activation, which consumes draft hidden states and outputs a sigmoid acceptance score. The Acceptance Head is trained once on the ShareGPT dataset and evaluated across all benchmarks to assess generalization. At inference we bound the draft length by $K_{\max} \in \{5, 6, 7\}$ and compute an early-exit threshold h from a simple runtime cost model; in practice we sweep h over $\{0.94, 0.95, 0.96\}$ to characterize the speed/quality trade-off. Our experiments are conducted on a single L20-48GB GPU.

3.2. Results

Experimental results are shown in Tab. 1. Consistent with prior work on SD methods [7], to quantify acceleration and the behavior of the speculative stage we report the following metrics: **(i) Wall-time speedup ratio** — the end-to-end wall-clock speedup relative to standard autoregressive decoding measured on real runs under identical hardware and batching conditions; and **(ii) Average acceptance length** τ — the mean number of tokens accepted from a single speculative draft per forward pass of the target LLM.

Because measured acceleration ratios are subject to run-to-run variability, we execute each experimental configuration four times under identical hardware and batching conditions and report the arithmetic mean as the primary summary statistic. The table entries are presented as "mean ± var," where ± denotes the sample variance across four measurements; this variance quantifies the stability of the speedup estimate and helps assess whether observed differences between methods are substantive or within measurement noise. Our Acceptance-Guided Adaptive Speculative Decoding achieves the

Table 1. Speed-up ratios of different methods versus average tokens accepted (τ) on L20. V denotes Vicuna, L2 denotes LLaMA2-Chat. Ours denotes our method: Acceptance-Guided Adaptive Speculative Decoding. We present results for different methods across four datasets. The mean represents the average performance across these four datasets. Methods like Medusa relax acceptance conditions under non-greedy settings, which do not guarantee lossless acceleration. Therefore, we do not compare our method with these methods.

Model	Method	Natural Ques.		CNN/DM		Alpaca		MT-bench		Mean	
		Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ
Temperature = 0											
V 13B	Medusa	$1.688 \pm 0.0002\times$	2.19	$1.593 \pm 0.0005\times$	2.09	$1.938 \pm 0.0005\times$	2.44	$2.015 \pm 0.0004\times$	2.58	$1.808 \pm 0.0301\times$	2.33
	Hydra	$2.120 \pm 0.0003\times$	2.88	$1.953 \pm 0.0003\times$	2.82	$2.518 \pm 0.0008\times$	3.49	$2.578 \pm 0.0013\times$	3.65	$2.292 \pm 0.0693\times$	3.21
	EAGLE	$2.125 \pm 0.0025\times$	2.94	$2.388 \pm 0.0008\times$	3.44	$2.493 \pm 0.0008\times$	3.48	$2.778 \pm 0.0021\times$	3.88	$2.446 \pm 0.0546\times$	3.44
	EAGLE-2	$2.360 \pm 0.0088\times$	3.64	$2.457 \pm 0.1100\times$	4.27	$2.873 \pm 0.01115\times$	4.50	$3.016 \pm 0.0043\times$	4.84	$2.676 \pm 0.0754\times$	4.31
	ours	$\mathbf{2.410 \pm 0.0440}\times$	3.60	$\mathbf{2.490 \pm 0.0040}\times$	4.19	$\mathbf{2.890 \pm 0.0036}\times$	4.40	$\mathbf{3.025 \pm 0.0019}\times$	4.83	$\mathbf{2.704 \pm 0.0675}\times$	4.26
V 7B	Medusa	$1.630 \pm 0.0003\times$	2.05	$1.500 \pm 0.0004\times$	2.01	$1.825 \pm 0.0009\times$	2.48	$1.928 \pm 0.0005\times$	2.51	$1.721 \pm 0.0276\times$	2.26
	Hydra	$1.960 \pm 0.0002\times$	2.91	$1.788 \pm 0.0004\times$	2.71	$2.375 \pm 0.0009\times$	3.57	$2.390 \pm 0.0008\times$	3.59	$2.128 \pm 0.0685\times$	3.20
	EAGLE	$2.063 \pm 0.0001\times$	3.12	$2.185 \pm 0.0007\times$	3.32	$2.300 \pm 0.0016\times$	3.66	$2.538 \pm 0.0008\times$	3.83	$2.271 \pm 0.0307\times$	3.48
	EAGLE-2	$2.185 \pm 0.0068\times$	3.71	$2.210 \pm 0.0064\times$	4.07	$2.593 \pm 0.0308\times$	4.61	$2.763 \pm 0.0106\times$	4.85	$2.437 \pm 0.0613\times$	4.31
	ours	$\mathbf{2.195 \pm 0.0017}\times$	3.75	$\mathbf{2.238 \pm 0.0002}\times$	4.02	$\mathbf{2.663 \pm 0.0646}\times$	4.62	$\mathbf{2.775 \pm 0.0005}\times$	4.80	$\mathbf{2.468 \pm 0.0649}\times$	4.30
L2 13B	EAGLE	$2.433 \pm 0.0027\times$	3.42	$2.425 \pm 0.0019\times$	3.56	$2.623 \pm 0.0058\times$	3.70	$2.730 \pm 0.0084\times$	3.80	$2.553 \pm 0.0168\times$	3.62
	EAGLE-2	$2.633 \pm 0.0161\times$	4.14	$2.570 \pm 0.0078\times$	4.29	$\mathbf{2.902 \pm 0.0155}\times$	4.54	$\mathbf{2.917 \pm 0.0035}\times$	4.68	$\mathbf{2.756 \pm 0.0243}\times$	4.41
	ours	$\mathbf{2.660 \pm 0.0054}\times$	3.90	$\mathbf{2.580 \pm 0.0015}\times$	4.00	$2.878 \pm 0.0470\times$	4.16	$2.903 \pm 0.0046\times$	4.39	$2.755 \pm 0.0191\times$	4.11
L2 7B	EAGLE	$2.253 \pm 0.0065\times$	3.43	$2.170 \pm 0.0024\times$	3.41	$2.460 \pm 0.0067\times$	3.65	$2.503 \pm 0.0064\times$	3.72	$2.346 \pm 0.0193\times$	3.55
	EAGLE-2	$2.443 \pm 0.0062\times$	4.17	$2.265 \pm 0.0061\times$	4.08	$\mathbf{2.655 \pm 0.0115}\times$	4.56	$2.680 \pm 0.0073\times$	4.52	$2.512 \pm 0.0286\times$	4.33
	ours	$\mathbf{2.448 \pm 0.0003}\times$	4.01	$\mathbf{2.280 \pm 0.0002}\times$	3.94	$2.615 \pm 0.0001\times$	4.32	$\mathbf{2.683 \pm 0.0001}\times$	4.38	$\mathbf{2.755 \pm 0.0191}\times$	4.16
Temperature = 1											
V 13B	EAGLE	$2.055 \pm 0.0042\times$	2.95	$2.168 \pm 0.0023\times$	3.16	$2.345 \pm 0.0013\times$	3.47	$2.455 \pm 0.0037\times$	3.49	$2.256 \pm 0.0239\times$	3.27
	EAGLE-2	$2.218 \pm 0.0002\times$	3.48	$2.353 \pm 0.0061\times$	3.90	$2.588 \pm 0.0002\times$	4.16	$2.760 \pm 0.0099\times$	4.21	$2.472 \pm 0.0459\times$	3.94
	ours	$\mathbf{2.273 \pm 0.0033}\times$	3.35	$\mathbf{2.375 \pm 0.0001}\times$	3.87	$\mathbf{2.593 \pm 0.0003}\times$	4.12	$\mathbf{2.768 \pm 0.0063}\times$	4.20	$\mathbf{2.502 \pm 0.0369}\times$	3.89
V 7B	EAGLE	$1.825 \pm 0.0014\times$	2.72	$1.940 \pm 0.0019\times$	3.03	$2.065 \pm 0.0009\times$	3.27	$2.175 \pm 0.0010\times$	3.42	$2.001 \pm 0.0173\times$	3.11
	EAGLE-2	$1.995 \pm 0.0009\times$	3.37	$2.033 \pm 0.0016\times$	3.69	$2.340 \pm 0.0008\times$	3.78	$2.400 \pm 0.0038\times$	4.12	$2.192 \pm 0.0324\times$	3.74
	ours	$\mathbf{2.013 \pm 0.0002}\times$	3.25	$\mathbf{2.038 \pm 0.0002}\times$	3.66	$\mathbf{2.343 \pm 0.0007}\times$	3.75	$\mathbf{2.470 \pm 0.0062}\times$	4.01	$\mathbf{2.216 \pm 0.0384}\times$	3.67
L2 13B	EAGLE	$2.285 \pm 0.0006\times$	3.32	$2.243 \pm 0.0008\times$	3.42	$2.485 \pm 0.0005\times$	3.58	$2.530 \pm 0.0005\times$	3.58	$2.001 \pm 0.0173\times$	3.48
	EAGLE-2	$2.498 \pm 0.0010\times$	4.05	$2.413 \pm 0.0009\times$	4.20	$2.690 \pm 0.0009\times$	4.39	$\mathbf{2.753 \pm 0.0106}\times$	4.54	$\mathbf{2.588 \pm 0.0191}\times$	4.30
	ours	$\mathbf{2.513 \pm 0.0008}\times$	3.82	$\mathbf{2.423 \pm 0.0012}\times$	3.94	$\mathbf{2.693 \pm 0.0021}\times$	4.08	$2.713 \pm 0.0002\times$	4.18	$2.585 \pm 0.0149\times$	4.00
L2 7B	EAGLE	$1.990 \pm 0.0004\times$	3.20	$1.888 \pm 0.0009\times$	3.14	$2.168 \pm 0.0009\times$	3.50	$2.175 \pm 0.0008\times$	3.45	$2.055 \pm 0.0148\times$	3.32
	EAGLE-2	$2.298 \pm 0.0095\times$	4.00	$2.228 \pm 0.00137\times$	3.98	$2.534 \pm 0.0235\times$	4.33	$\mathbf{2.567 \pm 0.0213}\times$	4.23	$2.407 \pm 0.0215\times$	4.13
	ours	$\mathbf{2.381 \pm 0.0001}\times$	3.95	$\mathbf{2.230 \pm 0.0002}\times$	3.80	$\mathbf{2.538 \pm 0.0003}\times$	4.18	$2.520 \pm 0.0006\times$	4.13	$\mathbf{2.417 \pm 0.0154}\times$	4.02

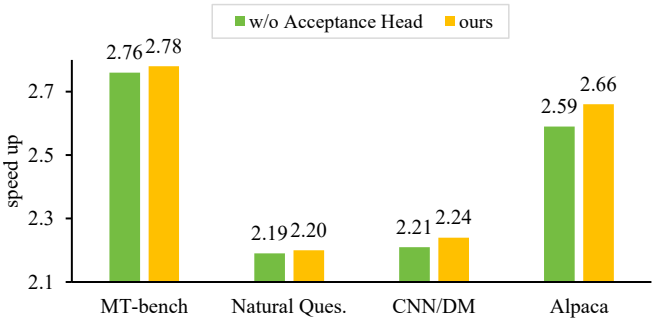

Fig. 2. Results of ablation studies on four tasks with temperature set to 0 on Vicuna 7B. "w/o Acceptance Head" indicates not using the acceptance head.

best or near-best speedup ratios across model sizes, tasks, and temperature settings. Compared with state-of-the-art SD baselines, it consistently yields higher mean speedups across configurations.

An important empirical nuance is that the measured average acceptance length τ for our method is generally smaller than that of the baselines in most tasks. This does not contradict the observed latency gains: the acceptance head primarily improves efficiency by early-stopping low-confidence expansions, thereby reducing wasted verification and resampling. The ablation in Fig. 2 illustrates this effect. In other words, adding the acceptance head yields modest but consistent wall-time improvements even when τ does not increase. Consequently, the speedup ratio is driven not merely by increasing τ but by a combination of (i) fewer verification attempts and rollbacks, (ii) better matching of the effective draft length K_{eff} to per-context difficulty, and (iii) consistent reductions in verification overhead. Importantly, the acceptance head introduces negligible runtime cost and requires no modification to the target LLM, facilitating low-cost integration into practical inference pipelines.

4. CONCLUSIONS

We propose Acceptance-Guided Adaptive Speculative Decoding, a lightweight framework that improves SD by dynamically adjusting draft length using an acceptance head. Unlike prior methods that fix K, our approach estimates token-level acceptance probabilities and adapts draft length, reducing unnecessary verifications and rollbacks. Experiments across tasks, model scales, and temperatures show consistent speedup gains. These gains come from aligning draft depth with context difficulty rather than longer acceptance sequences. The acceptance head is efficient, incurs negligible overhead, and requires no target LLM modifications, enabling practical deployment. Future work will extend to larger models and refine acceptance prediction and thresholding robustness.

5. ACKNOWLEDGMENTS

This work was supported by the NSFC Projects (No.12171479, No.62576346, No.12371516), the MOE Project of Key Research Institute of Humanities and Social Sciences (22JJD110001), the fundamental research funds for the central universities, and the research funds of Renmin University of China (24XNKJ13), and Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing.

6. REFERENCES

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

[2] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean, "Efficiently scaling transformer inference," *Proceedings of machine learning and systems*, vol. 5, pp. 606–624, 2023.

[3] Yaniv Leviathan, Matan Kalman, and Yossi Matias, "Fast inference from transformers via speculative decoding," in *International Conference on Machine Learning*. PMLR, 2023, pp. 19274–19286.

[4] Jinze Li, Yixing Xu, Haiduo Huang, Xuanwu Yin, Dong Li, Edith CH Ngai, and Emad Barsoum, "Gumiho: A hybrid architecture to prioritize early tokens in speculative decoding," *arXiv preprint arXiv:2503.10135*, 2025.

[5] Lawrence Stewart, Matthew Trager, Sujan Kumar Gonugondla, and Stefano Soatto, "The n-grammys: Accelerating autoregressive inference with learning-free batched speculation," *arXiv preprint arXiv:2411.03786*, 2024.

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

[7] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang, "Eagle-2: Faster inference of language models with dynamic draft trees," *arXiv preprint arXiv:2406.16858*, 2024.

[8] Fenglu Hong, Ravi Raju, Jonathan Lingjie Li, Bo Li, Urmish Thakker, Avinash Ravichandran, Swayambhoo Jain, and Changran Hu, "Training domain draft models for speculative decoding: Best practices and insights," *arXiv preprint arXiv:2503.07807*, 2025.

[9] Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui, "Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, Houda Bouamor, Juan Pino, and Kalika Bali, Eds., Singapore, Dec. 2023, pp. 3909–3925, Association for Computational Linguistics.

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

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

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

[13] Kaixuan Huang, Xudong Guo, and Mengdi Wang, "Specdec++: Boosting speculative decoding via adaptive candidate lengths," *arXiv preprint arXiv:2405.19715*, 2024.

[14] Chaojun Wang and Rico Sennrich, "On exposure bias, hallucination and domain shift in neural machine translation," *arXiv preprint arXiv:2005.03642*, 2020.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 2019, pp. 4171–4186.

[16] Benjamin Howson, Ciara Pike-Burke, and Sarah Filippi, "Delayed feedback in generalised linear bandits revisited," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2023, pp. 6095–6119.

[17] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al., "Judging llm-as-a-judge with mt-bench and chatbot arena," *Advances in neural information processing systems*, vol. 36, pp. 46595–46623, 2023.

[18] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al., "Natural questions: a benchmark for question answering research," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 453–466, 2019.

[19] Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al., "Abstractive text summarization using sequence-to-sequence rnns and beyond," *arXiv preprint arXiv:1602.06023*, 2016.

[20] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto, "Stanford alpaca: An instruction-following llama model," 2023.

[21] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao, "Medusa: Simple llm inference acceleration framework with multiple decoding heads," *arXiv preprint arXiv:2401.10774*, 2024.

[22] Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon, "Hydra: Sequentially-dependent draft heads for medusa decoding," *arXiv preprint arXiv:2402.05109*, 2024.

[23] Wei-Lin Chiang, Zhuohan Li, Ziqing Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al., "Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality," *See https://vicuna. lmsys. org (accessed 14 April 2023)*, vol. 2, no. 3, pp. 6, 2023.

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