EAGLE: Speculative Sampling Requires Rethinking Feature Uncertainty

Yuhui Li[1 2] Fangyun Wei[3] Chao Zhang[1] Hongyang Zhang[2 4]

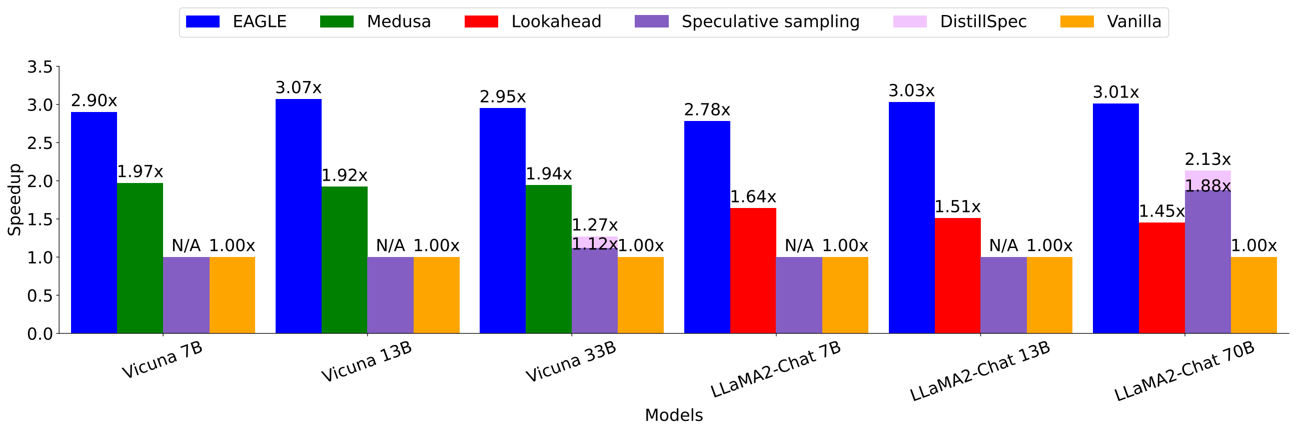

Figure 1: Speedup ratio of Vicuna and LLaMA2-Chat inference latency on the MT-bench for greedy (temperature=0) settings. Speedup ratio of Medusa and Lookahead are copied from their original technical reports. With speculative sampling, there is a lack of suitable draft models to accelerate the 7B model. Employing a 7B model as the draft model for a 13B model results in slow speeds due to the high overhead of the 7B model, rendering it less efficient than vanilla autoregressive decoding. These scenarios are marked as N/A. *In this paper, we only compare with speculative sampling based methods that do not need to finetune the original LLMs, ensuring the output text distribution remains constant.*

Abstract

Autoregressive decoding makes the inference of Large Language Models (LLMs) time-consuming. In this paper, we reconsider speculative sampling and derive two key observations. Firstly, autoregression at the feature (second-to-top-layer) level is more straightforward than at the token level. Secondly, the inherent uncertainty in feature (second-to-top-layer) level autoregression constrains its performance. Based on these insights, we introduce EAGLE (Extrapolation Algorithm for Greater Language-model Efficiency), a simple yet highly efficient speculative sampling framework. By incorporating a token sequence advanced by one time step, EAGLE effectively resolves the uncertainty, enabling precise second-to-top-layer feature prediction with minimal overhead. We conducted comprehensive evaluations of EAGLE, including all models from the Vicuna and LLaMA2-Chat series, the MoE model Mixtral 8x7B Instruct, and tasks in dialogue, code generation, mathematical reasoning, and instruction following. For LLaMA2-Chat 70B, EAGLE achieved a latency speedup ratio of **2.7x-3.5x**, doubled throughput, while maintaining the distribution of the generated text. The code is available at `https://github.com/SafeAILab/EAGLE`.

1. Introduction

Autoregressive decoding, the de facto standard for large language models (LLMs), generates tokens sequentially, leading to slow and costly generation. Speculative sampling (Leviathan et al., 2023; Chen et al., 2023a) based methods address this by dividing the process into a low-cost drafting stage and a *parallelized* verification stage over the drafted tokens, allowing for multiple tokens to be validated in a single LLM pass. These approaches accelerate generation by producing multiple tokens per pass. More importantly,

[1]Peking University [2]University of Waterloo [3]Microsoft Research [4]Vector Institute. Correspondence to: Hongyang Zhang <hongyang.zhang@uwaterloo.ca>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

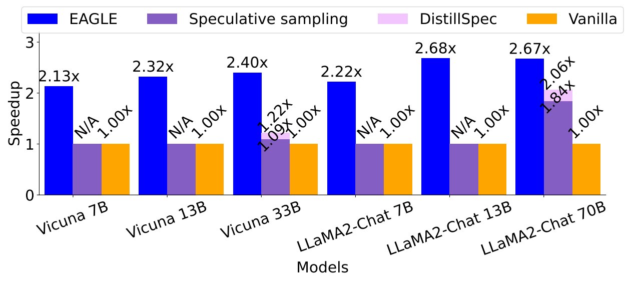

Figure 2: Speedup ratio on the MT-bench for non-greedy (temperature=1) settings. Lookahead is confined to greedy decoding, and the non-greedy generation of Medusa does not guarantee lossless performance. Therefore, EAGLE is not compared with these methods.

the verification stage ensures that the text distribution aligns precisely with the decoding results of the original LLM, maintaining the integrity of the generated content.

Applying speculative sampling hinges on finding a draft model that mirrors the original LLM's functionality but with reduced latency, often involving a lower-parameter version from the same LLM series. For instance, in the LLaMA2 (Touvron et al., 2023) series which includes models with 7B, 13B, and 70B parameters, using the 7B model as a draft model of the 70B model is valid, while finding a suitable draft model for the smallest 7B variant is tricky. An alternative could be to use TinyLLaMA (Zhang et al., 2024), but it is not feasible for instruct-tuned models due to the inconsistency in instruction templates between LLaMA2-Chat and TinyLLaMA-Chat. Despite the 7B model's potential as a draft model, its high overhead diminishes acceleration gains. Training a new, appropriately sized draft model specifically for speculative sampling is not an ideal solution either due to the high cost: TinyLLaMA is trained on 3,000B tokens, whereas EAGLE is trained on 2-4B tokens.

The key to enhancing acceleration in speculative sampling lies in reducing the time overhead and improving the acceptance rate of the draft by the original LLM (Chen et al., 2023b; Xia et al., 2023; Santilli et al., 2023). Numerous approaches focus on reducing the overhead of the drafting phase. Lookahead (Fu et al., 2023) employs n-gram and Jacobi iteration, while Medusa (Cai et al., 2023) utilizes a set of MLPs that predict tokens based on the second-to-top-layer feature of the original LLM. These strategies significantly decrease the latency in generating drafts, leading to improved acceleration. However, their effectiveness is limited by the lower accuracy of the resulting drafts, with Medusa achieving an accuracy of about 0.6, and Lookahead even lower. In contrast, our method attains an accuracy of approximately 0.8.

To overcome these limitations, we introduce EAGLE (Extrapolation Algorithm for Greater Language-model Efficiency), an efficient speculative sampling method, grounded in the following two observations.

Firstly, autoregression at the feature level is simpler than at the token level. In this paper, "features" refer to the second-to-top-layer features of the original LLM, located before the LM head. Compared to token sequences, which are simple transformations of natural language, feature sequences exhibit more regularity. Autoregressively processing at the feature level and then deriving tokens using the LM head of the original LLM yields better results than directly autoregressively predicting tokens. As illustrated in Figure 3, autoregressively predicting features yields better performance, demonstrated by a higher speedup ratio of 1.9x compared to 1.5x.

Secondly, the uncertainty inherent in the sampling process significantly constrains the performance of predicting the next feature. In text generation, the target LLM predicts the distribution of tokens and samples accordingly, introducing randomness. Features, being high-dimensional and continuous, cannot be treated similarly. As depicted in Figure 4, sampling different tokens like "am" or "always" leads to distinct feature sequences, introducing ambiguity into the feature-level autoregression. Medusa faces a similar issue in predicting spaced tokens, where it is uncertain whether the true target for the input f_I should be p_{am} or p_{always}. To address this issue, EAGLE inputs the token sequence from one time step ahead, which includes the sampling outcomes, into the draft model. In the example illustrated in Figure 4, this involves predicting f_{always} based on f_I and t_{always}, and predicting f_{am} based on f_I and t_{am}. As illustrated in Figure 3, by addressing the uncertainty, the speedup ratio further increases from 1.9x to 2.8x.

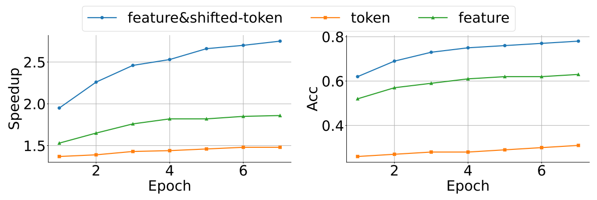

Figure 3: Accuracy and speedup ratio of draft models based on tokens, features and feature&shifted-token at temperature=0, tested on MT-bench with Vicuna 7B as the original LLM. Feature&shifted-token refers to using a feature sequence and a token sequence advanced by one time step as inputs.

We conducted experiments across dialogue, code generation, mathematical reasoning, and instruction following tasks using the MT-bench, HumanEval, GSM8K, and Alpaca datasets, respectively. Tested LLMs included all models from the Vicuna and LLaMA2-Chat series, along with

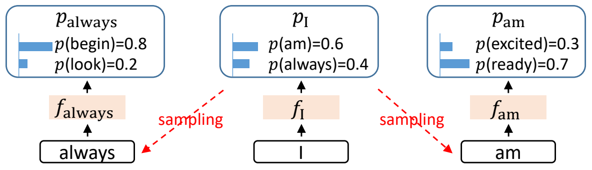

Figure 4: Uncertainty in feature sequences. The next feature following f_I is contingent on the sampling outcome and cannot be determined solely based on f_I, where both "always" and "am" are possible to follow the token "I" and lead to two branches.

Mixtral 8x7B Instruct. For LLaMA2-Chat 70B, EAGLE achieved a speedup ratio of **2.7x-3.5x**, doubled throughput, and theoretically guaranteed the preservation of the generated text's distribution. Figure 1 and 2 illustrates the performance of EAGLE on the MT-bench (Zheng et al., 2023), a highly realistic benchmark simulating actual applications and real-world scenarios, including multi-turn instructions akin to dialogues with ChatGPT. We have chosen to utilize this benchmark as it has been employed by the current state-of-the-art, including Lookahead and Medusa, to demonstrate their speedup ratios. This choice facilitates a fair and direct comparison between our approach and these methods. Compared to the recently proposed speculative sampling-based frameworks, Lookahead and Medusa, EAGLE achieves **1.7x-2.1x** and **1.5x-1.6x** speedups, respectively. EAGLE operates in parallel with other acceleration or throughput-improving methods, such as quantization, compilation, etc. Combining EAGLE with these techniques could further reduce the operational costs of LLM systems. For example, with gpt-fast (PyTorch Labs, 2023), EAGLE accelerates LLaMA2-Chat 7B decoding to 160.4 tokens/s on a single RTX 3090 GPU.

EAGLE boasts low training costs. For the LLaMA2-Chat 70B model, EAGLE trains a decoder layer with fewer than 1B parameters using no more than 70k dialogues from the ShareGPT dataset. The training is completed in 1-2 days on 4x A100 (40G) GPUs. The training of EAGLE on 7B, 13B and 33B models can even be conducted on a RTX 3090 node in 1-2 days. In practical applications, EAGLE requires only a single training session to provide acceleration for each query. As the number of queries increases, the amortized training cost of EAGLE becomes negligible.

Beyond performance, EAGLE offers additional advantages:

- **Generality:** EAGLE is applicable to any autoregressive LLMs (at least in principle). We have applied EAGLE to LLaMA2-Chat (7B, 13B, 70B), Vicuna (7B, 13B, 33B) and Mixtral 8x7B Instruct in a zero-shot way on the MT-bench, GSM8K, HumanEval and alpaca datasets. EAGLE adheres to the commonly used zero-shot/few-shot settings within the LLM community. All experiments employ the same weights, trained exclusively on the ShareGPT dataset, without any additional training on the evaluation datasets. The method adds only a lightweight plug-in (a single transformer decoder layer) to the LLM, which can be easily deployed in a production environment.
- **Reliability:** EAGLE does not involve any fine-tuning of the original LLM, and the preservation of the output distribution by EAGLE is theoretically guaranteed for both the greedy and non-greedy settings. This is in sharp contrast to Lookahead and Medusa which either focus solely on greedy settings or do not guarantee the preservation of distribution in these settings.

2. Preliminaries

Notations. In this paper, "target LLM" denotes the LLM intended for acceleration, while "draft model" refers to the model used for draft generation. "Feature" generally signifies the second-to-top-layer feature of a LLM, the hidden state before the Lm head. Tokens are denoted by lowercase t, their embeddings by e, features by f, and distributions by p. Sequences are represented in uppercase, for example, $T_{i:j}$ for $(t_i, t_{i+1}, \ldots, t_j)$. In a LLM, input $T_{1:j}$ is transformed into embeddings $E_{1:j}$ through the embedding layer, then to features $F_{1:j}$, and the LM Head maps f_j to a distribution $p_{j+1} = \text{LM\_Head}(f_j)$, sampling the next token t_{j+1}. Vanilla autoregression at the token level is described by $T_{1:j} \to E_{1:j} \to f_j \to p_{j+1} \to t_{j+1}$ for any integer $j \geq 1$.

Speculative sampling. Speculative sampling operates through draft and verification phases, with the drafting phase using a smaller model to generate γ tokens $\hat{T}_{j+1:j+\gamma}$ and their distributions $\hat{P}_{j+1:j+\gamma}$. In the verification phase, a single forward pass of the target LLM yields the probabilities $P_{j+1:j+\gamma}$. Tokens are then sequentially evaluated, with a token $\hat{t}_{j+i}$ having an acceptance probability $\min(1, p_{j+i}(\hat{t}_{j+i})/\hat{p}_{j+i}(\hat{t}_{j+i}))$. Upon the rejection of a token $\hat{t}_{j+i}$, all subsequent tokens are discarded, and this token is resampled based on a distribution $\text{norm}(\max(0, p_{j+i} - \hat{p}_{j+i}))$. As proven in Appendix A.1 of speculative sampling (Leviathan et al., 2023), this method equates to sampling directly from the target LLM. EAGLE adopts this method, ensuring that **the distribution of the generated text remains unchanged for both the greedy and non-greedy settings**. We provide a more detailed introduction to speculative sampling in Appendix A.

3. EAGLE

EAGLE, aligning with other speculative sampling-based methods, incorporates both a drafting phase and a verification phase.

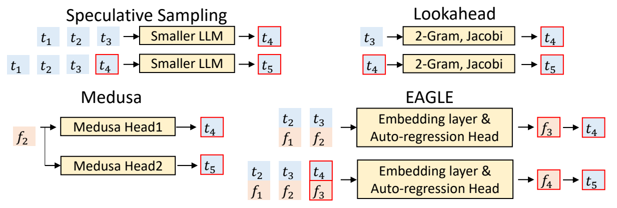

Figure 5: A comparison of the methods for drafting the fourth and fifth tokens, t_4 and t_5. t (represented by blue blocks) denotes tokens, and f (orange blocks) signifies the features, with subscripts indicating their positions in the sequence. The red border indicates the predictions of the draft model. For simplicity, the n in the n-gram for Lookahead, as shown in the figure, has been set to 2.

3.1. Drafting phase

The primary distinction between EAGLE and other methods lies predominantly in the drafting phase. Figure 5 illustrates a schematic of the drafting phase for different methods. Speculative sampling (Leviathan et al., 2023; Chen et al., 2023a) and Lookahead (Fu et al., 2023) predict tokens based on tokens. Medusa (Cai et al., 2023) independently predicts t_4 and t_5 using the feature f_2 from the target LLM. EAGLE predicts f_3 using the feature sequence (f_1, f_2) and the token sequence (t_2, t_3), advanced by one time step. From $p_4 =$ LM Head(f_3), t_4 is sampled. Subsequently, f_3 and t_4 are concatenated into the input sequence to predict the next feature f_4 and sample the subsequent token t_5.

Figure 6 illustrates the structure of EAGLE's draft model. The Embedding layer and LM Head employ the parameters of the target LLM and do not necessitate additional training. Figure 7 presents EAGLE's pipeline. The draft model takes as input a feature sequence of shape (bs, seq_len, hidden_dim) and an advanced token sequence of shape (bs, seq_len). It then converts the token sequence into a token embedding sequence of shape (bs, seq_len, hidden_dim), and concatenates it to form a fused sequence of shape (bs, seq_len, 2×hidden_dim). The Autoregression Head consisting of an FC layer and a decoder layer. The FC layer reduces the dimensionality of the fused sequence to (bs, seq_len, hidden_dim) and then we utilize the decoder layer to predict the next feature. The LM Head calculates the distribution based on the feature, from which the next token is sampled. Finally, the predicted feature and the sampled token are concatenated into the input, facilitating the continuation of the autoregressive process. EAGLE creates a tree-structured draft using tree attention, generating a draft tree with depth m and more than m tokens through m forward passes. For instance, as shown in Figure 7, EAGLE drafts a 10-token tree with just 3 forward passes. The actual tree structure employed by EAGLE is detailed in Appendix B.1.

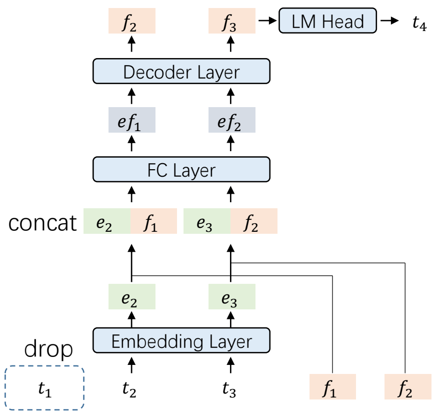

Figure 6: The network architecture of EAGLE's draft model. t represents token, f represents feature, and e represents token embedding. f and e are k-dimensional vectors. The input to the draft model is the token sequence from one step ahead, so the first token t_1 is dropped. t_2 and t_3 are input into the embedding layer to obtain embeddings e_1 and e_2, which are then concatenated with f_1 and f_2, resulting in two $2k$-dimensional vectors. The FC layer reduces their dimensionality back to k, and then they are input into the decoder layer, finally yielding f_2 and f_3. Feature f_3 is fed into the LM head to obtain the draft token t_4.

3.2. Training of the draft models

Predicting the next feature constitutes a regression task, for which we employ Smooth L1 loss (see Figure 6):

$$\hat{f}_{i+1} = \text{Draft\_Model}(T_{2:i+1}, F_{1:i}),$$

$$L_{reg} = \text{Smooth L1}(f_{i+1}, \hat{f}_{i+1}).$$

Predicting features is an intermediary objective of the draft model, with the ultimate goal being the prediction of tokens to generate a sequence of tokens. Consequently, we also employ classification loss to directly optimize towards this final objective:

$$p_{i+2} = \text{Softmax}(\text{LM\_Head}(f_{i+1})),$$
$$\hat{p}_{i+2} = \text{Softmax}(\text{LM\_Head}(\hat{f}_{i+1})),$$
$$L_{cls} = \text{Cross\_Entropy}(p_{i+2}, \hat{p}_{i+2}).$$

By integrating regression loss and classification loss, we train the Autoregression Head using the combined loss function $L = L_{reg} + w_{cls}L_{cls}$. Typically, the classification loss is an order of magnitude larger than the regression loss in numerical terms. Consequently, we set w_{cls} to 0.1.

EAGLE's Autoregression Head is ideally trained with autoregressively generated text from the target LLM, yet this approach is costly. Fortunately, EAGLE exhibits low sensitivity to training data (ablation study in Section 4.3.3).

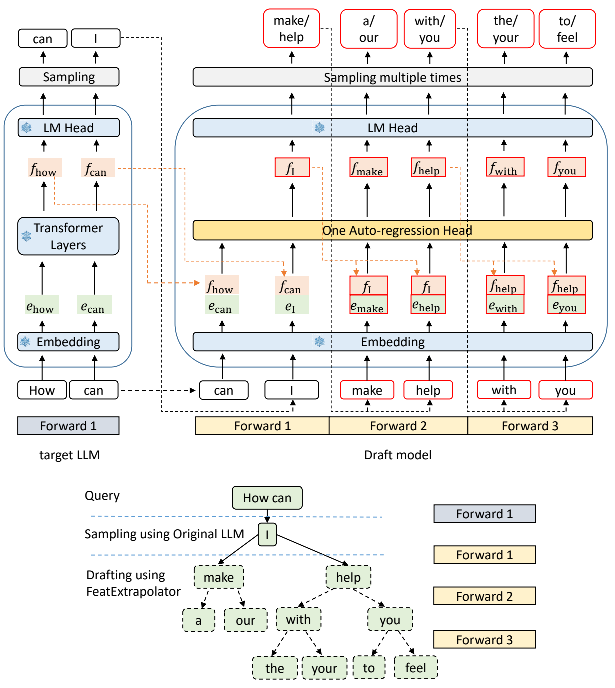

Figure 7: Pipeline of EAGLE. The upper section illustrates the computational process, while the lower section displays the corresponding generation results for each step. In the upper section, green blocks represent token embeddings, orange blocks represent features, red boxes indicate the predictions of the draft model, and blue modules with snowflake icons represent the use of target LLM parameters, which are not subject to training.

Instead of employing text generated by the target LLM, we utilize a fixed dataset, substantially reducing the overhead. During the drafting phase, EAGLE autoregressively processes features. Inaccuracies in features can lead to error accumulation. To mitigate this issue, we employ data augmentation by adding random noise sampled from a uniform distribution $\mathcal{U}(-0.1, 0.1)$ to features of the target LLM during training (Jain et al., 2023).

3.3. Verification phase

Employing tree attention, the target LLM computes the probability of each token in the tree-structured draft through a single forward pass. At every node of the draft tree, we recursively apply speculative sampling algorithms to sample or adjust the distribution (details in Appendix B.2), consistent with SpecInfer (Miao et al., 2023), ensuring that the distribution of the output text aligns with that of the target LLM. Concurrently, we document accepted tokens and their features for use in the next drafting phase.

4. Experiments

Models and tasks. We conducted experiments on Vicuna models (7B, 13B, 33B), LLaMA2-chat models (7B, 13B, 70B), and Mixtral 8x7B Instruct, encompassing the common sizes of current mainstream LLMs. We evaluated EAGLE across multiple tasks including multi-turn dialogue, code generation, mathematical reasoning, and instruction following, employing the MT-bench (Zheng et al., 2023), HumanEval (Chen et al., 2021), GSM8K (Cobbe et al., 2021), and Alpaca (Taori et al., 2023) datasets, respectively. Speculative sampling (Leviathan et al., 2023) conducted experiments with a batch size of 1, a setting subsequently adopted by other works such as DistillSpec (Zhou et al., 2023) and BiLD (Kim et al., 2023). Similarly, the majority of our experiments also adopted this setting. Experiments with a batch size greater than 1 are presented in Section 4.4.

Metrics. Like other speculative sampling-based methods, EAGLE primarily focuses on latency rather than throughput. We assess acceleration effects using the following metrics:

- Walltime speedup ratio: The actual test speedup ratio relative to vanilla autoregressive decoding.
- Average acceptance length τ: The average number of tokens accepted per forward pass of the target LLM.
- Acceptance rate α: The ratio of accepted to generated tokens during drafting, gauges draft accuracy. It's less applicable for tree drafts due to multiple tokens sampled per location with only one accepted. Hence, when measuring this metric, **we utilize chain drafts without tree attention**, aligning with speculative sampling and DistillSpec. EAGLE's draft model inputs feature and token sequences. Autoregressive feature processing can propagate errors, so we measure the acceptance rate as n-α, considering n features predicted by the draft model, potentially with inaccuracies.

Acceleration of EAGLE theoretically guarantees the preservation of the target LLMs' output distribution. Consequently, evaluating the quality of EAGLE's generated results is both unnecessary and meaningless.

Training. We fixed the target LLMs. EAGLE was trained on the ShareGPT dataset, utilizing 68,000 dialogue iterations with a learning rate set at 3e-5. We employed the AdamW optimizer with beta values (β_1, β_2) set to (0.9, 0.95) and implemented gradient clipping of 0.5. The trainable parameters of EAGLE corresponding to the 7B, 13B, 33B, and 70B models are 0.24B, 0.37B, 0.56B, and 0.99B, respectively. The trainable parameters of EAGLE for MoE model Mixtral 8x7B is 0.28B. EAGLE is characterized by its low training cost; the Autoregression Head is trainable within 1-2 days on an A100 40G server for the 70B models.

4.1. Effectiveness

Figures 1 and 2, along with Table 1, display the speedup ratios of EAGLE. EAGLE demonstrates better acceleration at temperature=0 compared to temperature=1. For instance, for LLaMA2-Chat 13B at temperature=0, the speedup ratios range from 3.01x to 3.76x, while at temperature=1, they range from 2.66x to 2.89x. In code generation tasks (HumanEval), EAGLE achieves its best acceleration performance. This is attributed to the prevalence of fixed templates in code, making it easier to generate drafts for these templates. Compared to recently introduced speculative sampling-based methods, Lookahead and Medusa, EAGLE is faster by 1.70x-2.08x and 1.47x-1.60x, respectively. Employing speculative sampling in the Vicuna and LLaMA2-Chat series is challenging. For the 7B model, there is no suitable draft model. For other sizes, using the 7B model as the draft model, we iterated through draft lengths from 2 to 10 and reported the highest speedup ratio. For the 13B model, we observed no improvement in speed. For the 33B and 70B models, the speedup ratios were 1.12x and 1.88x, respectively. For DistillSpec, to ensure fairness, we used the same training data as EAGLE. Additionally, the divergence function employed follows the FKL as detailed in Appendix A.1 of the DistillSpec paper. While distillation slightly improved the speedup ratio, the limited enhancement is because distillation aims to increase the draft model's acceptance rate, while the bottleneck for speculative sampling performance lies in the high overhead of the draft model.

Tables 1 and 2 indicate that in EAGLE, the target LLM generates 3.2-4.5 tokens per forward pass, surpassing vanilla decoding which produces only one token per forward pass, thereby significantly increasing generation speed. As shown in Figure 2 and Appendix C, the acceptance rate for completely accurate feature sequences, 0-α, significantly exceeds that for sequences with a single erroneous feature, 1-α, indicating the impact of feature errors on draft model performance. Yet, the slight variation between 1-α to 4-α underscores EAGLE's robustness to feature errors and its adept handling of error accumulation.

Table 3 reveals that EAGLE achieved a 1.5x speedup with the Mixtral 8x7B Instruct model. This modest acceleration, compared to models like LLaMA, is due to a shorter average acceptance length and the complexity of accelerating MoE models via speculative sampling. MoE models typically require reading the weights of only two experts per token during vanilla autoregressive decoding. However, during the verification phase of speculative sampling, processing multiple tokens may necessitate accessing the weights of more than two experts, contrasting with dense decoder-only models where all weights are read regardless of the number of tokens forwarded.

Table 1: Speedup ratio and average acceptance length τ on HumanEval, GSM8K, and Alpaca. T denotes temperature, V represents Vicuna, and LC stands for LLaMA2-Chat.

		HumanEval		GSM8K		Alpaca	
	Model	Speedup	τ	Speedup	τ	Speedup	τ
T=0	V 7B	3.33x	4.29	3.01x	4.00	2.79x	3.86
	V13B	3.58x	4.39	3.08x	3.97	3.03x	3.95
	V 33B	3.67x	4.28	3.25x	3.94	2.97x	3.61
	LC 7B	3.17x	4.24	2.91x	3.82	2.78x	3.71
	LC 13B	3.76x	4.52	3.20x	4.03	3.01x	3.83
	LC 70B	3.52x	4.42	3.03x	3.93	2.97x	3.77
T=1	V 7B	2.39x	3.43	2.34x	3.29	2.21x	3.30
	V13B	2.65x	3.63	2.57x	3.60	2.45x	3.57
	V 33B	2.76x	3.62	2.77x	3.60	2.52x	3.32
	LC 7B	2.61x	3.79	2.40x	3.52	2.29x	3.33
	LC 13B	2.89x	3.78	2.82x	3.67	2.66x	3.55
	LC 70B	2.92x	3.76	2.74x	3.58	2.65x	3.47

Table 2: Average acceptance length τ and acceptance rate α on MT-bench. T denotes temperature.

	Model	τ	0-α	1-α	2-α	3-α	4-α
T=0	Vicuna 7B	3.94	0.79	0.74	0.72	0.73	0.67
	Vicuna 13B	3.98	0.79	0.74	0.72	0.74	0.70
	Vicuna 33B	3.68	0.74	0.69	0.67	0.67	0.66
	LLaMA2-Chat 7B	3.62	0.76	0.69	0.67	0.68	0.68
	LLaMA2-Chat 13B	3.90	0.77	0.69	0.69	0.70	0.71
	LLaMA2-Chat 70B	3.81	0.75	0.69	0.65	0.64	0.64
T=1	Vicuna 7B	3.17	0.71	0.68	0.66	0.66	0.65
	Vicuna 13B	3.20	0.73	0.68	0.68	0.67	0.69
	Vicuna 33B	3.22	0.71	0.67	0.64	0.64	0.64
	LLaMA2-Chat 7B	3.30	0.71	0.66	0.66	0.66	0.64
	LLaMA2-Chat 13B	3.45	0.73	0.69	0.66	0.67	0.67
	LLaMA2-Chat 70B	3.46	0.73	0.67	0.64	0.66	0.65

4.2. Case study: EAGLE + gpt-fast

EAGLE is compatible with other acceleration technologies. We conducted experiments combining EAGLE with gpt-fast, which employs quantization and compilation for acceleration. As shown in Table 4, by integrating EAGLE with gpt-fast, we increased the generation speed of LLaMA2-Chat 7B on a single RTX 3090 to 160.4 tokens/s.

4.3. Ablation study

4.3.1. Tree attention

EAGLE, similar to SpecInfer and Medusa, employs tree attention, where both the generation and validation of drafts are tree-structured. In contrast, methods like speculative sampling do not use tree attention, resulting in chain-structured draft generation and validation. Figure 8 and Table 5 present comparative results indicating the impact of using tree attention. The implementation of tree draft and verification in EAGLE results in an approximate increase of 0.6-0.8 in the average acceptance length and about 0.3-0.5

Table 3: Speedup ratio, average acceptance length τ, and acceptance rate α on MT-bench at temperature=0. The target LLM is Mixtral 8x7B Instruct-v0.1.

Speedup	τ	0-α	1-α	2-α	3-α	4-α
1.50x	3.25	0.67	0.62	0.61	0.64	0.63

Table 4: Generation speed of EAGLE combined with gpt-fast, evaluated on MT-bench with LLaMA2-Chat 7B at temperature=0.

Precision	FP16	int4
Vanilla (Huggingface)	24.5 tokens/s	N/A
gpt-fast	55.1 tokens/s	106.9 tokens/s
EAGLE + gpt-fast	100.2 tokens/s	160.4 tokens/s

in the speedup ratio. Compared to chain draft and verification, tree draft and verification do not increase the number of forward passes in the model (both the target LLM and the draft model), but they do increase the number of tokens processed per forward pass. Consequently, the improvement in the speedup ratio is less pronounced than the increase in average acceptance length. Notably, even without employing tree draft and verification, EAGLE demonstrates a significant acceleration effect, approximately in the range of 2.3x-2.7x.

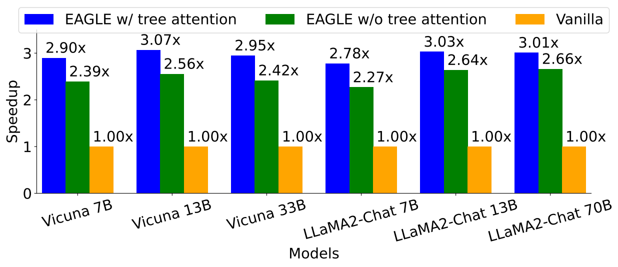

Figure 8: Speedup ratios of EAGLE with and without the use of tree attention. The evaluation dataset is MT-bench, with the temperature parameter set to 0.

4.3.2. Inputs of draft models

Compared to other speculative sampling-based methods, the key innovation of EAGLE lies in its utilization of features computed by the target LLM and the incorporation of sampling outcomes into the input of the draft model to address randomness. We conducted an ablation study on Vicuna 7B, assessing draft models with varying inputs. We tested four types of inputs: feature&shifted-token (EAGLE), feature&unshifted-token, token, and feature. Both feature&shifted-token (EAGLE) and feature&unshifted-token integrate semantic information at different levels. The distinction lies in the fact that feature&shifted-token (EAGLE) inputs tokens advanced by one time step, equipping it to address randomness effectively. Apart from the use of a FC layer to reduce dimensionality for the feature&token input, the structure of the draft model remains entirely consistent. Figure 9 presents the experimental outcomes on the MT-bench with Vicuna 7B as the target LLM. Three observations can be drawn.

Table 5: Average acceptance length τ of EAGLE with and without the use of tree attention. The evaluation dataset is MT-bench, with the temperature parameter set to 0.

Vicuna			LLaMA2-Chat		
Size	Chain	Tree	Size	Chain	Tree
7B	3.20	3.94 (+0.74)	7B	3.00	3.62 (+0.62)
13B	3.23	3.98 (+0.75)	13B	3.18	3.90 (+0.68)
33B	2.97	3.68 (+0.71)	70B	3.12	3.81 (+0.69)

- First, when the number of parameters of the draft model is limited, utilizing features yields slightly better results than tokens.
- Second, merging features and tokens modestly boosts performance, mainly as discrete, error-free tokens mitigate feature error accumulation, evident from the similar 0-α of feature&unshifted-token and feature-only draft models, with a significantly improved 1-α.
- Third, addressing the randomness inherent in the sampling process results in the most significant improvement. The feature&shifted-token scheme, compared to feature&unshifted-token, adds no complexity yet markedly enhances the draft model's capability by simply advancing the token by one time step, allowing the draft model to account for the randomness in sampling.

4.3.3. Training data

EAGLE uses a fixed dataset for training, avoiding increased overhead from using the target LLM for generating training data. Ablation study (see Table 6) shows that data from the target LLM marginally improves performance, indicating EAGLE's low sensitivity to training data and justifying the fixed dataset approach for cost reduction.

4.4. Batch size and throughput

Inference in LLMs is memory-bound (Patterson, 2004; Shazeer, 2019), leaving GPU computational resources underutilized. The principle behind the speculative sampling-based approach in enhancing generation speed lies in more effectively utilizing GPU computational resources. As the

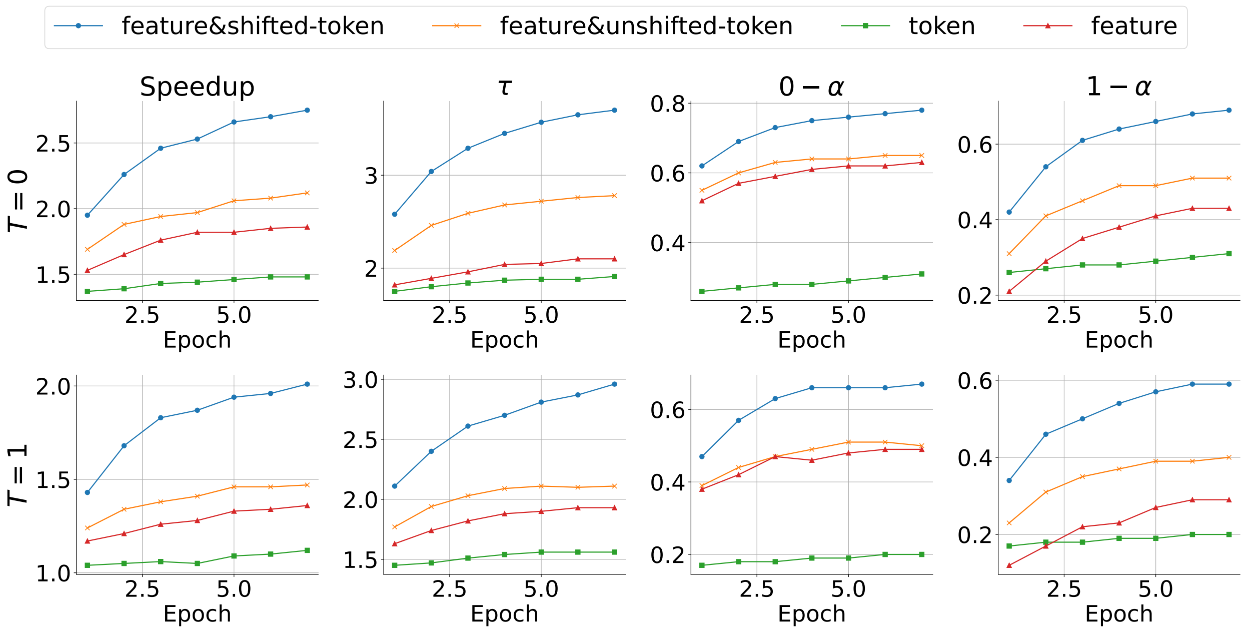

Figure 9: Performance of draft models with varying inputs. The target LLM is Vicuna 7B, and the test dataset is MT-bench. Speed refers to the walltime speedup ratio, τ denotes the average acceptance length, 0-α represents the acceptance rate with entirely precise inputs, 1-α indicates the acceptance rate when the input includes one imprecise feature, and T refers to the temperature.

Table 6: The speedup ratios and average acceptance length τ using different training datasets evaluated on the MT-bench, with the target LLM being LLaMA2-Chat 7B and the temperature set to 0. "Fixed dataset" refers to both questions and answers originating from the ShareGPT dataset. "Data generated by target LLM" denotes that while questions are sourced from the ShareGPT dataset, the answers are generated by the target LLM.

Training data	Speedup	τ
Fixed dataset	2.78x	3.62
Data generated by target LLM	2.88x	3.75

batch size increases, the available computational capacity of the GPU decreases, leading to a reduction in the acceleration effect. In this section, we present experimental results for scenarios where the batch size exceeds 1. As demonstrated in Table 7, the speedup ratio diminishes with increasing batch size. When using Vicuna 7B as the target LLM, the speedup ratio at bs=4 is higher than at bs=3. This is attributed to the fact that, during the verification phase of EAGLE, the target LLM processes multiple tokens in a single forward pass, and the processing at bs=4 is faster than at bs=3. In contrast, with vanilla autoregressive decoding where the target LLM processes one token per forward pass, the speeds at bs=3 and bs=4 are nearly identical.

Although speculative sampling-based methods predominantly focus on latency, we also investigated EAGLE's throughput for batch size > 1, another key metric for LLM systems. Compared to vanilla autoregressive decoding, EAGLE requires slightly more CUDA memory. For Vicuna 7B as the target LLM, operating under a memory constraint of a single RTX 3090 with 24G of CUDA memory, the maximum batch size (bs) for vanilla autoregressive decoding and EAGLE are 8 and 7, respectively. In the case of LLaMA2-Chat 70B, constrained by 4 A100 (40G) GPUs totaling 160G of CUDA memory, the maximum bs for vanilla autoregressive decoding and EAGLE are 5 and 4, respectively. All evaluations were conducted at FP16 precision. We calculated the throughput for different bs and selected the maximum value. Both vanilla autoregressive decoding and EAGLE achieve maximum throughput at their respective maximum bs. Tree attention consumes more computational resources. At bs=7, the computational resources are less abundant, making the non-use of tree attention more advantageous. As illustrated in Table 7, EAGLE achieves a 2x increase in throughput.

Table 7: Speedup ratios at different batch sizes and throughput of EAGLE. The evaluation dataset is MT-bench, with the temperature parameter set to 0.

Batch size	1	2	3	4	Throughput
Vicuna 7B	2.90x	2.87x	2.65x	2.76x	1.97x
LLaMA2-Chat 70B	3.01x	2.81x	2.50x	2.40x	1.99x

5. Related Work

There has been considerable research into accelerating language models, involving techniques such as distillation (Hinton et al., 2015), quantization (Hubara et al., 2018; Shen et al., 2020; Kim et al., 2021; Zadeh et al., 2020; Zafrir et al., 2019), pruning (Gale et al., 2019; Sanh et al., 2020), and efficient operator design (Dao et al., 2022). These methods aim to reduce the latency per forward pass.

Similar to our approach are frameworks based on speculative sampling. Early works (Stern et al., 2018; Sun et al., 2021) accelerated greedy decoding, while speculative sampling (Leviathan et al., 2023; Chen et al., 2023a) extended it to non-greedy sampling, provably maintaining the original output distribution. Ensuring unchanged output distribution makes acceleration more challenging; many studies have explored lossy acceleration as a trade-off. For instance, DistillSpec (Zhou et al., 2023) modifies acceptance probabilities using a lenience function, BiLD (Kim et al., 2023) accepts drafts if the distance metric from the target LLM distribution is below a certain threshold, and Medusa (Cai et al., 2023) uses a minimum of a hard threshold and an entropy-dependent threshold for truncation. In contrast, EAGLE does not employ any relaxations and maintains the output distribution of the LLM unchanged.

The primary differences among speculative sampling-based methods manifest predominantly in the drafting phase. Speculative sampling (Leviathan et al., 2023; Chen et al., 2023a) utilizes a lower-parameter version of the target LLM as the draft model. Self-Speculative Decoding (Zhang et al., 2023) skips some layers of the target LLM during draft generation. SpecInfer (Miao et al., 2023) employs a set of small models to generate drafts in parallel. Cascade Speculative Drafting (Chen et al., 2023b) and Staged Speculative Decoding (Spector & Re, 2023) cascade different overhead draft models. Online Speculative Decoding (Liu et al., 2023) trains the draft model on a distribution of queries. Methods (Hooper et al., 2023; Fu et al., 2023; Yang et al., 2023b; Liu et al., 2024; Ankner et al., 2024) such as Medusa (Cai et al., 2023) do not employ a separate target LLM; instead, they generate drafts by utilizing features or weights from the target LLM. REST (He et al., 2023) generates drafts based on retrieval methods. LLMA (Yang et al., 2023a), used for tasks like grammatical correction where input and output overlap, retrieves drafts directly from the input.

6. Conclusion

In this paper, we introduce EAGLE, an efficient framework for speculative sampling. EAGLE conducts the drafting process autoregressively at the more structured (second-to-top-layer) feature level and mitigates sampling uncertainty in predicting the next feature by incorporating tokens from one time step ahead. EAGLE is guaranteed to preserve the output distribution of the LLM while significantly enhancing generation speed. On MT-bench, EAGLE is 2.1x-3.8x faster than vanilla autoregressive decoding, 1.7x-2.1x faster than Lookahead, and 1.5x-1.6x faster than Medusa.

Acknowledgements

Yuhui Li and Chao Zhang are supported by the National Nature Science Foundation of China under Grant 62071013 and National Key R&D Program of China under Grant 2018AAA0100300. Hongyang Zhang is supported by the NSERC Discovery Grant RGPIN-2022-03215, DGECR-2022-00357.

Impact Statement

EAGLE preserves the output distribution of the original LLM, and hence, inherently carries no risk by itself. While the original LLM may generate inaccurate or harmful content, such issues are independent of EAGLE's functionality.

References

Ankner, Z., Parthasarathy, R., Nrusimha, A., Rinard, C., Ragan-Kelley, J., and Brandon, W. Hydra: Sequentially-dependent draft heads for medusa decoding. *arXiv preprint arXiv:2402.05109*, 2024.

Cai, T., Li, Y., Geng, Z., Peng, H., and Dao, T. Medusa: Simple framework for accelerating LLM generation with multiple decoding heads. `https://github.com/FasterDecoding/Medusa`, 2023.

Chen, C., Borgeaud, S., Irving, G., Lespiau, J.-B., Sifre, L., and Jumper, J. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023a.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Chen, Z., Yang, X., Lin, J., Sun, C., Huang, J., and Chang, K. C.-C. Cascade speculative drafting for even faster LLM inference. *arXiv preprint arXiv:2312.11462*, 2023b.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Dao, T., Fu, D., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

Fu, Y., Bailis, P., Stoica, I., and Zhang, H. Breaking the sequential dependency of LLM inference using lookahead decoding, November 2023. URL https://lmsys.org/blog/2023-11-21-lookahead-decoding/.

Gale, T., Elsen, E., and Hooker, S. The state of sparsity in deep neural networks.(2019). *arXiv preprint cs.LG/1902.09574*, 2019.

He, Z., Zhong, Z., Cai, T., Lee, J. D., and He, D. Rest: Retrieval-based speculative decoding. *arXiv preprint arXiv:2311.08252*, 2023.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Hooper, C., Kim, S., Mohammadzadeh, H., Genc, H., Keutzer, K., Gholami, A., and Shao, S. SPEED: Speculative pipelined execution for efficient decoding. *arXiv preprint arXiv:2310.12072*, 2023.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Quantized neural networks: Training neural networks with low precision weights and activations. *journal of machine learning research*, 18(187):1–30, 2018.

Jain, N., Chiang, P.-y., Wen, Y., Kirchenbauer, J., Chu, H.-M., Somepalli, G., Bartoldson, B. R., Kailkhura, B., Schwarzschild, A., Saha, A., et al. NEFTune: Noisy embeddings improve instruction finetuning. *arXiv preprint arXiv:2310.05914*, 2023.

Kim, S., Gholami, A., Yao, Z., Mahoney, M. W., and Keutzer, K. I-bert: Integer-only bert quantization. In *International conference on machine learning*, pp. 5506–5518. PMLR, 2021.

Kim, S., Mangalam, K., Moon, S., Malik, J., Mahoney, M. W., Gholami, A., and Keutzer, K. Speculative decoding with big little decoder. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Liu, F., Tang, Y., Liu, Z., Ni, Y., Han, K., and Wang, Y. Kangaroo: Lossless self-speculative decoding via double early exiting. *arXiv preprint arXiv:2404.18911*, 2024.

Liu, X., Hu, L., Bailis, P., Stoica, I., Deng, Z., Cheung, A., and Zhang, H. Online speculative decoding. *arXiv preprint arXiv:2310.07177*, 2023.

Miao, X., Oliaro, G., Zhang, Z., Cheng, X., Wang, Z., Wong, R. Y. Y., Chen, Z., Arfeen, D., Abhyankar, R., and Jia, Z. SpecInfer: Accelerating generative LLM serving with speculative inference and token tree verification. *arXiv preprint arXiv:2305.09781*, 2023.

Patterson, D. A. Latency lags bandwith. *Communications of the ACM*, 47(10):71–75, 2004.

PyTorch Labs. gpt-fast. https://github.com/pytorch-labs/gpt-fast/, 2023.

Sanh, V., Wolf, T., and Rush, A. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in Neural Information Processing Systems*, 33:20378–20389, 2020.

Santilli, A., Severino, S., Postolache, E., Maiorca, V., Mancusi, M., Marin, R., and Rodola, E. Accelerating transformer inference for translation via parallel decoding. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12336–12355, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.689. URL https://aclanthology.org/2023.acl-long.689.

Shazeer, N. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

Shen, S., Dong, Z., Ye, J., Ma, L., Yao, Z., Gholami, A., Mahoney, M. W., and Keutzer, K. Q-bert: Hessian based ultra low precision quantization of bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8815–8821, 2020.

Spector, B. and Re, C. Accelerating LLM inference with staged speculative decoding. *arXiv preprint arXiv:2308.04623*, 2023.

Stern, M., Shazeer, N., and Uszkoreit, J. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31, 2018.

Sun, X., Ge, T., Wei, F., and Wang, H. Instantaneous grammatical error correction with shallow aggressive decoding. *arXiv preprint arXiv:2106.04970*, 2021.

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B.

Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. LlAMA 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Xia, H., Ge, T., Wang, P., Chen, S.-Q., Wei, F., and Sui, Z. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3909–3925, 2023.

Yang, N., Ge, T., Wang, L., Jiao, B., Jiang, D., Yang, L., Majumder, R., and Wei, F. Inference with reference: Lossless acceleration of large language models. *arXiv preprint arXiv:2304.04487*, 2023a.

Yang, S., Lee, G., Cho, J., Papailiopoulos, D., and Lee, K. Predictive pipelined decoding: A compute-latency trade-off for exact llm decoding. *arXiv preprint arXiv:2307.05908*, 2023b.

Zadeh, A. H., Edo, I., Awad, O. M., and Moshovos, A. Gobo: Quantizing attention-based nlp models for low latency and energy efficient inference. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 811–824. IEEE, 2020.

Zafrir, O., Boudoukh, G., Izsak, P., and Wasserblat, M. Q8bert: Quantized 8bit bert. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*, pp. 36–39. IEEE, 2019.

Zhang, J., Wang, J., Li, H., Shou, L., Chen, K., Chen, G., and Mehrotra, S. Draft & verify: Lossless large language model acceleration via self-speculative decoding. *arXiv preprint arXiv:2309.08168*, 2023.

Zhang, P., Zeng, G., Wang, T., and Lu, W. TinyLlama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023.

Zhou, Y., Lyu, K., Rawat, A. S., Menon, A. K., Rostamizadeh, A., Kumar, S., Kagy, J.-F., and Agarwal, R. DistillSpec: Improving speculative decoding via knowledge distillation. *arXiv preprint arXiv:2310.08461*, 2023.

A. Background: Speculative Sampling

In this section, we introduce standard speculative sampling (Leviathan et al., 2023; Chen et al., 2023a) to help readers unfamiliar with this field better understand EAGLE. We use t_i to denote the i-th token and $T_{a:b}$ to represent the token sequence $t_a, t_{a+1}, \cdots, t_b$. We use M_o to denote the original LLM and M_d to represent the draft model.

Modern LLMs generate text autoregressively, requiring the full model weights to be transferred from memory to cores for each token generation. The time cost of accessing weights far exceeds the computation cost. Therefore, LLM inference is memory-bound. In text generation tasks, the difficulty of generating different tokens varies, and a smaller model can generate some simple tokens. This observation has inspired a class of methods, represented by speculative sampling, which generate multiple tokens in one forward pass to accelerate LLM inference. The core idea of speculative sampling methods is to first draft and then verify: quickly generate a potentially correct draft and then check which tokens in the draft can be accepted.

Consider a prefix $T_{1:j}$, speculative sampling alternates between drafting and verification stages. In the drafting stage, speculative sampling invokes a draft model M_d (a smaller LLM than M_o) to generate a draft $\hat{T}_{j+1:j+k}$ with $T_{1:j}$ as the prefix. In the verification stage, speculative sampling calls the original LLM M_o to check the draft $\hat{T}_{j+1:j+k}$ and concatenates the correct parts into the prefix.

Drafting Stage. We use the draft model M_d to autoregressively generate k tokens while recording the corresponding distributions $\hat{p}$:

$$\hat{t}_{j+1} \sim \hat{p}_{j+1} = M_d(T_{1:j}),$$
$$\hat{t}_{j+i} \sim \hat{p}_{j+i} = M_d(\text{concat}(T_{1:j}, \hat{T}_{j:j+i-1})), i = 2, \cdots, k,$$

where concat$(\cdot, \cdot)$ denotes the concatenation of two sequences. The draft $\hat{T}_{j+1:j+k}$ generated by M_d has a lower computational cost while having a certain probability of being consistent with the generation results of M_o.

Verification Stage. The verification stage checks the draft $\hat{T}_{j+1:j+k}$ and keeps the parts consistent with M_o. We leverage the parallelism of LLMs. Given the input sequence concat$(T_{1:j}, \hat{T}_{j+1:j+k})$, one forward pass of the LLM can compute $k+1$ distributions:

$$p_{j+1} = M_o(T_{1:j}),$$
$$p_{j+i} = M_o(\text{concat}(T_{1:j}, \hat{T}_{j:j+i-1})), i = 2, \cdots, k+1.$$

Then, we decide whether to accept each token in the draft from front to back. For token $\hat{t}_{j+i}$, the probability of it being accepted is $\min(1, p_{j+i}(\hat{t}_{j+i})/\hat{p}_{j+i}(\hat{t}_{j+i}))$. If $\hat{t}_{j+i}$ is accepted, we continue checking the next token; otherwise, we sample a token from the distribution norm$(\max(0, p_{j+i} - \hat{p}_{j+i}))$ to replace $\hat{t}_{j+i}$ and discard the remaining tokens in the draft. The result of this sampling method is exactly consistent with directly sampling from the distribution p computed by M_o. The proof can be found in Appendix A.1 of (Leviathan et al., 2023).

We concatenate the accepted draft to $T_{1:j}$ to form a new prefix, and then start the next round of drafting and verification.

B. Implementation Details

B.1. Tree Structure

Utilizing tree attention, EAGLE generates a tree-structured draft. The left side of Figure 10 illustrates the tree structure of the draft, while the right side depicts the corresponding chain-structured draft when tree attention is not used (as utilized in the ablation study detailed in Section 4.3.1). In a greedy setting, we select the top k tokens with the highest probabilities as child nodes. In a non-greedy setting, we sample k tokens. The number of child nodes, k, can be inferred from Figure 9; for instance, $k = 4$ at the root node. Regardless of employing a tree-structured or chain-structured draft, the draft model undergoes 5 forward passes during the draft phase. During the verification phase, each token's probability is obtained through a single forward pass by the target LLM.

Why do we use such a tree structure? The choice of the tree structure, as depicted in Figure 9, was not rigorously optimized but rather based on intuition: branches of higher-probability tokens should be deeper and wider. For this paper, all models across all experiments utilized the draft structure shown in Figure 9. However, the optimal tree structure is likely context-dependent. For instance, as batch size increases and redundant computational resources decrease, a smaller tree might be preferable. Tuning the draft structure could potentially lead to improved performance.

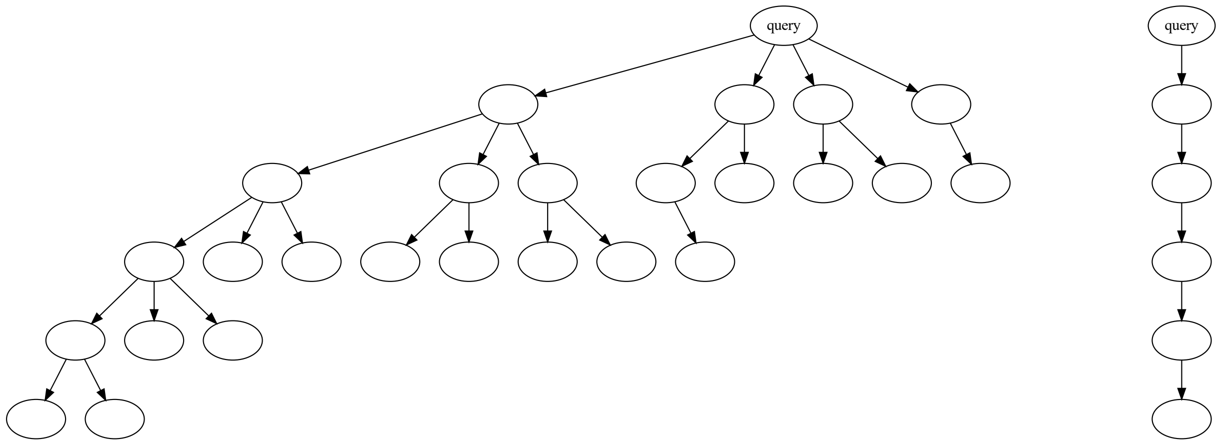

Figure 10: Structure of EAGLE's draft. The left side shows the draft structure when tree attention is employed, while the right side depicts the draft structure without the use of tree attention.

B.2. Multi-Round Speculative Sampling

Unlike the chain-structured draft of speculative sampling, EAGLE employs a tree-structured draft, necessitating modifications to the sampling algorithm. Single-round speculative sampling of standard speculative sampling takes as input a draft token t, the draft distribution $\hat{p}$, and the original LLM's distribution p. It accepts the draft token $\hat{t}$ with probability $\min(1, p(\hat{t})/\hat{p}(\hat{t}))$. If not accepted, it performs naive sampling from the distribution norm$(\max(0, p - \hat{p}))$. Multi-round speculative sampling simply modifies the naive sampling in the case of non-acceptance to recursively call single-round speculative sampling. The pseudocode for Multi-round speculative sampling is provided in Algorithm 1.

Algorithm 1 Multi-round speculative sampling

```
Input: Target distribution $p$, draft tokens $\hat{t}_i$ and distributions $\hat{p}_i$ for each $i$ from 1 to $k$, where $\hat{t}_i$ is sampled from $\hat{p}_i$,
Output: a sample $x \sim p$ ;
$i \leftarrow 1$
for $i \leq k$ do
    $r \leftarrow U(0,1)$
    if $r < p(\hat{t}_i)/\hat{p}_i(\hat{t}_i)$ then
        Return $\hat{t}_i$
    end if
    $p \leftarrow norm(max(0, p - \hat{p}_i))$
    $i \leftarrow i + 1$
end for
Sample $t \sim p$
Return $t$
```

C. Detailed experimental results

Table 8 displays the speedup ratio, average acceptance length τ and acceptance rate α of EAGLE on HumanEval, GSM8K, and Alpaca datasets.

Table 8: Speedup ratio, average acceptance length τ and acceptance rate α on HumanEval, GSM8K, and Alpaca at temperature = 0.

Dataset	Model	Speedup	τ	0-α	1-α	2-α	3-α	4-α
HumanEval	Vicuna 7B	3.33x	4.29	0.82	0.77	0.72	0.69	0.71
	Vicuna13B	3.58x	4.39	0.85	0.78	0.74	0.72	0.73
	Vicuna 33B	3.67x	4.28	0.83	0.77	0.74	0.70	0.70
	LLaMA2-Chat 7B	3.17x	4.24	0.81	0.76	0.73	0.74	0.72
	LLaMA2-Chat 13B	3.76x	4.52	0.85	0.80	0.78	0.76	0.75
	LLaMA2-Chat 70B	3.52x	4.42	0.84	0.79	0.75	0.73	0.74
GSM8K	Vicuna 7B	3.01x	4.00	0.79	0.71	0.70	0.71	0.70
	Vicuna13B	3.08x	3.97	0.79	0.71	0.67	0.68	0.64
	Vicuna 33B	3.25x	3.94	0.79	0.71	0.67	0.67	0.67
	LLaMA2-Chat 7B	2.91x	3.82	0.75	0.69	0.64	0.65	0.63
	LLaMA2-Chat 13B	3.20x	4.03	0.80	0.70	0.70	0.68	0.66
	LLaMA2-Chat 70B	3.03x	3.93	0.77	0.71	0.66	0.64	0.60
Alpaca	Vicuna 7B	2.79x	3.86	0.74	0.68	0.66	0.66	0.67
	Vicuna13B	3.03x	3.95	0.72	0.67	0.64	0.63	0.64
	Vicuna 33B	2.97x	3.61	0.70	0.64	0.64	0.63	0.64
	LLaMA2-Chat 7B	2.78x	3.71	0.73	0.66	0.62	0.64	0.62
	LLaMA2-Chat 13B	3.01x	3.83	0.75	0.67	0.64	0.63	0.63
	LLaMA2-Chat 70B	2.97x	3.77	0.76	0.68	0.65	0.61	0.62