Speculative Decoding via Early-exiting for Faster LLM Inference with Thompson Sampling Control Mechanism

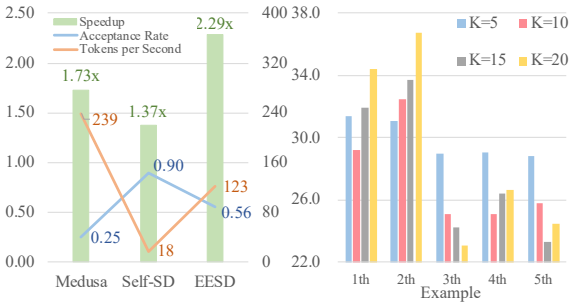
Anonymous ACL submission

Abstract

The recent advancements in large language models (LLMs) have been extraordinary, yet the escalating inference costs associated with them present challenges in real-world applications. To address these challenges, we propose a novel approach called Early-exiting Speculative Decoding (EESD) with lossless acceleration. Specifically, EESD utilizes a segment of the LLM to generate draft tokens, incorporating Early-exiting structures after the first N layers. To enhance the quality of draft tokens, a self-distillation method is integrated. This early-exiting design not only reduces deployment and training costs but also significantly accelerates the token generation speed. Moreover, we introduce a novel sampling mechanism that leverages Thompson Sampling to regulate the generation processes, automatically determining the quantity of draft tokens in each round. The original LLM is then employed to validate these draft tokens through a single forward pass, and thus guarantees that the final output text maintains a distribution consistent with vanilla auto-regressive decoding. The experimental results on both 13B and 70B models demonstrate that our approach decodes tokens at a markedly accelerated rate compared to prior methods, showing the effectiveness of our approach.

1 Introduction

Large Language Models (LLMs) excel in various NLP tasks due to their immense parameters and complex network (OpenAI, 2023; Chowdhery et al., 2023; Touvron et al., 2023a,b). However, these models generate tokens one-by-one in an auto-regressive manner during inference, making the generation exceedingly resource-intensive and time-consuming. To overcome this bottleneck, researchers have introduced an effective decoding technique - Speculative Decoding (SD) (Leviathan et al., 2023; Chen et al., 2023; Miao et al., 2023). SD essentially introduces two models, a small model (the draft model) which is used to concurrently generate multiple draft tokens, and the original LLM (the target model) which is employed for draft token verification. In this way, SD maintains the same performance as the auto-regressive decoding while boosting the inference speed.



(a) End-to-End Speedup (b) Cost vs. drafting steps

Figure 1: Experimental results using LLaMA-2-70B on the Gsm8k. (a) Speedup comparison with Medusa (Cai et al., 2023) and Self-SD (Zhang et al., 2023b). EESD achieves the highest speedup with the best tradeoff between token generation speed and acceptance rate. (b) Generation costs (seconds) with different drafting steps (K) in randomly select five samples from Gsm8k. The optimal value of K varies across different samples, indicating that a fixed K value for all samples is not ideal.

Compared to vanilla Speculative Decoding (Leviathan et al., 2023; Chen et al., 2023), several advanced models such as Medusa (Cai et al., 2023) and Self-SD (Zhang et al., 2023b) have been introduced, which only require deploying one LLM instead of two models, resulting in fewer resources required for both training and deployment. While these approaches achieve promising results, there are two main limitations. *First*, they fail to optimize the trade-off between the quality and speed of the draft token generation. For example, as shown in Figure 1a, while Medusa can generate draft tokens rapidly, the quality[1] of

[1]We use the acceptance rate to represent the quality of the draft tokens, which is percent of draft tokens are accepted by the target model during the verification.

these tokens tends to be subpar. On the other hand, Self-SD manages to produce high quality draft tokens but does so at a much slower speed, resulting in lower overall speedup. *Second*, most SD approaches commence verification after generating a pre-defined length of draft tokens (referred to as drafting steps K). The choice of K significantly influences the acceleration of the inference process. Typically, larger drafting steps result in faster end-to-end generation, but there is a potential trade-off as the acceptance rate may decrease if the quality of the longer draft sequence is not high. As illustrated in Figure 1b, the optimal value of K varies across different examples. This variation suggests that utilizing a fixed K may not yield the most effective strategy. Instead, an adaptive method is preferable to determine when to terminate the drafting process.

To address these challenges, in this paper, we propose a novel Early-Exiting Speculative Decoding method, named EESD, to facilitate efficient and qualitative generation of draft tokens. Specifically, EESD introduce an Early-exiting layer that is superimposed on the first-N layers of the LLM, which has shown powerful predictive potential in previous research (Bae et al., 2023; Schuster et al., 2022). A self-distillation method is further employed to enhance the learning of the Early-exiting layer. To identify the optimal drafting steps, we reformulate the task of determining the length of draft token generation as a multi-armed bandit (MAB) problem, and propose a novel Control Mechanism based on Thompson Sampling (TS) that is well-studied for estimating unknown parameters and facilitating optimal decision making. Comprehensive evaluations on both 13B and 70B models demonstrate the superior performances of our approach over several baselines. The main contributions of this paper are summarized as follows:

- We introduce a novel Early-exiting framework for generating draft tokens, which allows a single LLM to fulfill the drafting and verification stages. We train it using self-distillation. Our investigations indicate that this framework strikes an excellent balance between the quality and speed of draft token generation.
- We conceptualize the generation length of draft tokens as a MAB problem and propose a novel control mechanism based on Thompson Sampling, which leverages sampling to devises an optimal strategy.
- We conducted extensive experiments on three benchmarks. The results affirm that EESD can significantly improve the model's inference speed, outperforming existing SD methods.

2 Related Work

LLM Compression The central objective of model compression is to alleviate computational demands and enhance inference speed. The research on LLM compression mainly includes three directions, including knowledge distillation (Zhang et al., 2023a; Li et al., 2023; Gu et al., 2023), network pruning (Ma et al., 2023; Xia et al., 2023; Frantar and Alistarh, 2023) and quantization (Xiao et al., 2023; Liu et al., 2023; Frantar et al., 2022; Lin et al., 2023). The methods mentioned above work by reducing the model's footprint, thereby decreasing memory demand and enhancing the speed of inference. However, these methods sacrifices a degree of LLM's capability.

Efficient Decoding Leviathan et al. (2023); Chen et al. (2023) propose a method that uses a small model to generate draft tokens and then uses LLM for verification, which accelerates the decoding process while guaranteeing lossless outputs, named as Speculative Decoding. However, some researchers suggest that the extra small model is not essential for SD. For instance, Medusa (Cai et al., 2023) generates draft tokens by leveraging additional parameters instead of small model, while Self-SD (Zhang et al., 2023b) uses the substructure of LLM to generate draft tokens. In addition, He et al. (2023) unveil a method that replaces the generation of draft tokens with a large text database. In the other hand, some researchers introduce an Early-exiting method. This method dynamically modifies the depth of the decoder for each token generation, making predictions at an intermediate layer (Teerapittayanon et al., 2016; Elbayad et al., 2020). Furthermore, Bae et al. (2023) propose a new Early-exiting method that incorporates a shallow-deep module and synchronized parallel decoding.

3 Methodology

The overall model architecture of EESD is illustrated in Figure 2. Essentially, our model is composed of three key components. 1) the Early-exiting layer that built on top of the first few layers of the LLM as a draft model; 2) the self-distillation method to enhance the learning of the draft model

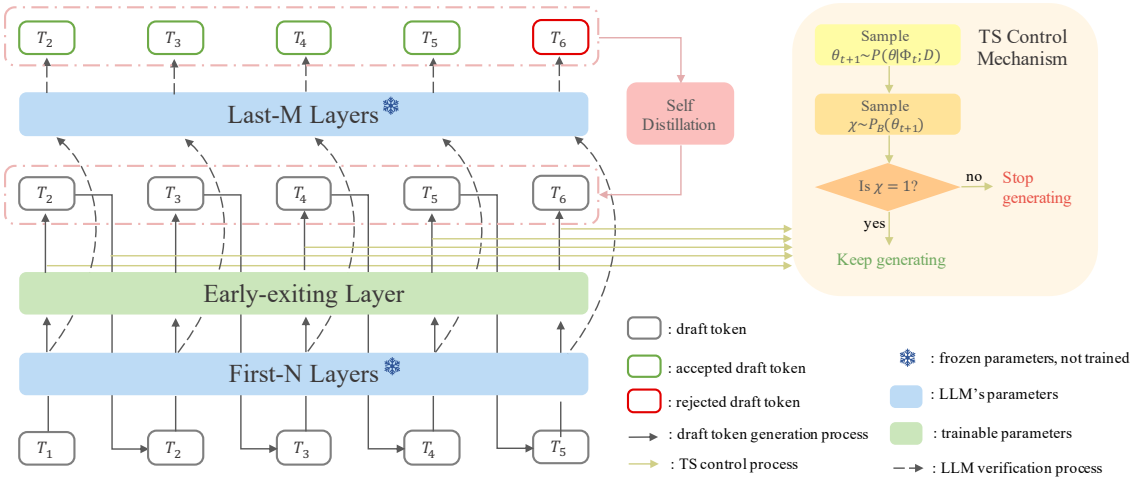


Figure 2: The framework of EESD which consists of three components: (1) Early-exiting layer which generate draft tokens efficiently and effectively; (2) Self-distillation which distills knowledge from the LLM (the target model); (3) TS control mechanism which can predict the optimal timing of terminating the draft token generation in each round. We divide the LLM (the target model) into two parts: the first-N layers and the last-M layers.

and boost the text generation quality; and 3) the Thompson Sampling control mechanism that adaptively determines the length of the draft tokens conditioned on its quality. We present the details of these components in the following sections.

3.1 Early-exiting Layer

Most previous methods (Bae et al., 2023; Kavehzadeh et al., 2023) use non-continuous subnetwork of original LLM (target model) as their draft model. In this work, we utilize the continuous first-N layers approach, which yields one significant advantage: the kv-cache of the draft model and the target model can share the first-N layers, thereby trimming redundant computation. Concreately, we formulated an Early-exiting Layer with the computation process as elucidated below,

$$p(y_t) = softmax(W^T Transformer^e(H_t^N)), \quad (1)$$

where H_t^N represents the hidden state of the N-th layer, which is calculated from the first-N layers of the origin LLM. And t represents t-th token. $p(y_t)$ is obtained from H_t^N through one layer of transformer. For LLaMA model, we also add RMSNorm layer before the output prediction head, which is $RMSNorm(Transformer^e(H_t^N))$.

As mentioned above, we incorporate an learnable Transformer layer subsequent to the first-N layers, and train this layer and W in Eq.(1) (RMSNorm parameters are trained as well for LLaMA model). In order to speed up the model convergence, we initialize the $Transformer^e$ and W with the last layer and predict head of the original LLM respectively. Since the training is confined to only a single Transformer layer and W, with the first-N layers being frozen, this approach dramatically reduce the computational resources.

3.2 Self-Distillation

To further enhance the effectiveness of the draft model, we employ self-distillation to learn the knowledge from the LLM. The key idea is that there is a large amount of valuable data used during the training of the LLM. However, it is usually impossible to obtain these original data as most of them are not directly accessible. Therefore, we propose to bridge the gap with self-distillation, which guides the learning of the early-exiting layer by transfering the knowledge from the generated text of the LLM. Specifically, Liu et al. (2023) suggest that a hybrid generation strategy of greedy and sampling is effective, and we adopt this approach for text generation from LLM. It is worth noting that the text generated by the LLM may contain certain lower-quality samples. We thus retain a subset of the open-source data for training purposes. The parameters of the Early-exiting Layer are trained utilizing an amalgamation of data generated by the LLM and open-source datasets, with the cross-entropy loss between the prediction of it and the ground truth of mixed datasets.

3.3 Thompson Sampling Control Mechanism

The above methods can effectively improve the quality and speed of draft token generation. However, as we mentioned in the introduction, a predetermined drafting step is not a good strategy. Consequently, we view the controlling draft model

generation as a MAB problem. Specifically, we view it as a Bernoulli process, where the model independently determines whether to continue draft token generating, denoted as $P_B(\theta)$. And the probability θ is uncertain, related to the input sample. The Thompson Sampling (TS) method can better estimate unknown variables through balancing exploration and exploitation (Slivkins, 2019). As illustrated in Algorithm 1, we utilize TS algorithm to adaptively estimate θ. The crux of the TS algorithm involves modeling uncertain parameters θ as a posterior distribution using Bayesian theory, i.e. $P(\theta|D)$, with D representing observed environmental samples. The core of this algorithm lies in designing a reasonable posterior distribution, which we will detail in the following chapters.

TS with Beta Distribution Considering that the sample adheres to the Bernoulli distribution, we adopt the conjugate distribution approach and select the Beta distribution as posterior distribution. This setup means the prior and posterior distributions share the same distribution function but with different parameters, which greatly simplifies the computational process. The probability density function of the Beta distribution is as follows:

$$Beta(\theta;\alpha,\beta) = \frac{1}{B(\alpha,\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}, \tag{2}$$

where $B(\alpha,\beta)$ is a standardization function. The Beta distribution has two parameters, α and β, so $\Phi = \{\alpha,\beta\}$ and $\Phi_0 = \{\alpha_0,\beta_0\}$ in Algorithm 1. According to Bayesian inference, we can update parameters α and β according to the following formula, which is the 18-th step in Algorithm 1,

$$\alpha_t = \alpha_{\{t-|Q_v|\}} + r, \tag{3}$$

$$\beta_t = \beta_{\{t-|Q_v|\}} + (n-r), \tag{4}$$

where r represents the number of successful experiments in the observed samples, and n represents the total number of experiments. And in Algorithm 1, the value of r is set to $|Q_v|-1$, indicating that the draft model should continue to generate on this set of tokens (i.e., $\chi = 1$). And the value of n is set to $min(|Q_v|+1, |Q_d|)$, indicating the number of tokens that have been validated by the target model. Because we stop verifying when we encounter an inconsistent token, subsequent tokens are not considered.

TS with Calibration In the prior segment, we introduce a TS algorithm with Beta distribution to improve the estimation of θ. However, according to MAB theory, initial phases are more exploration-focused, which may result in less accurate θ estimations (Ou et al., 2019; Peng et al., 2019). To alleviate this issue, we propose a novel hybrid method that combines *Model Prediction* and *Sampling Prediction*. We rely more on model prediction to mitigate inaccuracies from initial exploration. As the sampling prediction begins to converge later, we calibrate the model prediction with sampling prediction to achieve a more precise θ.

We train a single-layer to predict the value of θ. The computation formula for this is as follows,

$$\theta_M^{t+i} = Sigmoid(W_p(W^i H_t^{(T)}, H_{t+i}^{(D)})), \tag{5}$$

where t is the number of tokens that have already been generated, and i is the number of new tokens generated by the draft model in the current loop. $H_t^{(T)}$ represents the hidden state of the LLM (target model) at position t in the last layer, while the corresponding $H_{t+i}^{(D)}$ represents the hidden state of the draft model at position $t+i$ in the last layer. W^i is the transformation matrix at the i-th position for target model, and considering that i might be very large, we have restricted the number of $W^i \in \mathbb{R}^{d\times d}$, i.e. $i = min(i, 10)$. We sample a portion of training dataset to train the parameters

Algorithm 1 TS Control Algorithm

Require: Target Model M_t; Draft Model M_d; Max Generation Length L; Hyperparameters Φ_0; Input Prompt $\{x_0, ..., x_n\}$.

Initialize prior probability $P(\theta|\Phi_0)$ according to user-set hyperparameters Φ_0.
Initialize the result set $Q_g \leftarrow \{x_0, ..., x_n\}$ and $t \leftarrow 0$.
while t < L **do**
  Initialize the draft model result set and i, $Q_d \leftarrow Null, i \leftarrow 0$.
  while t + i < L **do**
    Get now token $x_i \leftarrow M_d(Q_g \bigcup Q_d)$.
    Add token x_i to set Q_d.
    Sample θ_{t+i} from $P(\theta|\Phi_t; D)$.
    Sample $\chi \in \{0,1\}$ from Bernoulli distribution $P_B(\theta_{t+i})$.
    $i \leftarrow i+1$.
    if $\chi = 0$ **then**
      Break
    end if
  end while
  Verify the results Q_d by Target Model M_t and get Q_v received by M_t, $Q_v \subseteq Q_d$.
  Add the set Q_v to Q_g, $Q_g \leftarrow Q_g \bigcup Q_v$.
  Update t according to length of Q_v, $t \leftarrow t + |Q_v|$.
  Update the parameters Φ_t of the posterior distribution.
end while
return Q_g

$\{W^1, W^2, ..., W^{10}\}$ and $W_p \in \mathbb{R}^{2\times 2d}$. The labels for this data are acquired by comparing the tokens produced by both the target and draft models, signifying the true value of χ. Following this, we update the parameters using cross-entropy loss.

According to the central limit theorem, when the sample size is sufficiently large, the sample mean adheres to a Gaussian distribution. Therefore, we make an assumption that sample mean $\widetilde{\chi}$ of χ in one drafting round follows a Gaussian distribution, i.e. $\widetilde{\chi} \sim \mathcal{N}(\mu, \sigma_S^2)$. As supported by Bayesian theory, when the random variable follows a Gaussian distribution with a known variance but an unknown mean and the prior distribution is also a Gaussian distribution, it satisfies the conjugate distribution. Consequently, we define μ to follow a Gaussian distribution with the model's predict score as mean and predict error as variance, $\mu \sim \mathcal{N}(\theta_M, \sigma_M^2)$. In Algorithm 1, we set $\Phi_0 = \{\sigma_M, \sigma_S, \hat{\theta_0}\}$, where σ_M , σ_S and $\hat{\theta_0}$ are hyperparameters set by the user. In Step 8 of Algorithm 1, we sample θ value from Gaussian distribution, $\theta_{t+i} \sim \mathcal{N}(\mu_{t+i}, \sigma_{t+i}^2)$, and compute the values of μ and σ using the formula provided,

$$\mu_{t+i} = \frac{\sigma_S^2}{n\sigma_M^2 + \sigma_S^2}\theta_M^{t+i} + \frac{n\sigma_M^2}{n\sigma_M^2 + \sigma_S^2}\hat{\theta}_t, \tag{6}$$

$$\frac{1}{\sigma_{t+1}^2} = \frac{1}{\sigma_M^2} + \frac{n}{\sigma_S^2}, \tag{7}$$

Where n is verification times. We update parameter Φ based on the following formula in step 18,

$$\hat{\theta}_t = \frac{\hat{\theta}_{\{t-|Q_v|\}} * (t - |Q_v| + 1) + |Q_v|}{t+1}, \tag{8}$$

For a more detail, please refer to Appendix B.

4 Experiment

4.1 Setup

Training stage We randomly extract 100,000 samples from the SlimPajama (Soboleva et al., 2023) to train LLaMA-2-70B, LLaMA-2-13B and CodeLLaMA-2-13B. And use the ShareGPT[2] dataset to train LLaMA-2-70B-chat and Vicuna-13B (Zheng et al., 2023). We choose the first-5 layers as the draft model for 70B models and first-3 for 13B models. For fair comparison, we train our model and Medusa (Cai et al., 2023) using the same data and set Medusa head at 4. More training details can be found in Appendix C

[2]https://huggingface.co/datasets/Aeala/ShareGPT_Vicuna_unfiltered

Evaluation stage We conduct experiments on three benchmarks under 1-shot setting: Gsm8k (Cobbe et al., 2021), XSum (Narayan et al., 2018) and Humaneval (Chen et al., 2021). We randomly select 500 instances from the test set for evaluation. And we set final output length at 512 and batch size at 1. We set the drafting step K at 10 for Vanilla SD (Chen et al., 2023) and Self SD (Zhang et al., 2023b). The reported results are the average of 10 different runs. We have only conducted on greedy generation[3], as the findings from the top-p sampling exhibit similar trends.

Metrics We propose a Harmonic Mean (HM) to assess the quality of the draft tokens and strategy for generating them, while the specific calculation formula is as $S = \frac{2*v_d*r_d}{v_d+r_d} * 100\%$, where v_d indicates the percent of draft tokens that are accepted by the target model, and r_d represents the proportion of tokens that come from the draft model. More detailed explanation in Appendix D. Due to the verification process, all baseline and EESD methods can assure that the generation results are identical to the original LLM, hence we only need to compare their speedup.

All experiments are conducted on NVIDIA A100-80GB GPUs.

4.2 Main Results

We report evaluation results for Gsm8k and XSum in Table 1, and for Humaneval in Table 2. As shown in Table 1 and 2, it is clear that EESD significantly outperforms the previous methods on both 13B and 70B models, especially on LLaMA-2-70B, which demonstrates the effectiveness of our approach. There are several key observations from these results. **First**, we observe that EESD can yield 2.45× times speedup on CodeLLaMA-2-13B for coding task, suggesting our method exhibits particular effectiveness within this domain. **Second**, compared to Vanilla SD and Medusa, EESD shows superior results with fewer training and deployment parameters. For instance, EESD achieves up to 2.13× and 1.80× times faster speeds on llama-2-70b model with just 1.12B parameters being trained. While we introduce an additional training process as compared to Self-SD, we manage to significantly improve speed effectiveness, utilizing minimal training resources. **Third**, we discover that a stronger

[3]For all experiments, we only generate one top-1 draft token candidate in Step 6 of Algorithm 1, and retain the result consistent with the target model's top-1 token on verifying at Step 15 of Algorithm 1.

Model	Method	Trainable Params	Deployment Params	Gsm8k		XSum	
				HM	Speedup	HM	Speedup
LLaMA-2-70B	Vanilla SD	7B†	77B	80.35	1.88×	67.30	1.46×
	Self SD	-	70B	78.64	1.37×	68.60	1.23×
	Medusa	1.32B	71.3B	33.75	1.73×	25.69	1.42×
	EESD (+Beta-TS)	1.12B	71.1B	58.79	2.13×	51.91	1.80×
	EESD (+Cali-TS)	1.12B+0.67B‡	71.8B	62.25	**2.29**×	53.41	**1.86**×
LLaMA-2-70B-chat	Vanilla SD	7B†	77B	63.90	1.44×	62.75	1.39×
	Self SD	-	70B	67.99	1.13×	68.38	1.16×
	Medusa	1.32B	71.3B	22.86	1.42×	17.02	1.20×
	EESD (+Beta-TS)	1.12B	71.1B	47.76	1.79×	40.73	1.51×
	EESD (+Cali-TS)	1.12B+0.67B‡	71.8B	48.23	**1.82**×	41.85	**1.55**×
LLaMA-2-13B	Vanilla SD	7B†	20B	84.59	0.96×	75.63	0.77×
	Self SD	-	13B	80.53	1.37×	77.61	1.35×
	Medusa	760M	13.8B	31.71	1.77×	25.03	1.53×
	EESD (+Beta-TS)	481M	13.5B	57.22	1.91×	55.46	1.84×
	EESD (+Cali-TS)	481M+262M‡	13.7B	58.97	**2.04**×	56.45	**1.92**×
Vicuna-13B	Vanilla SD	7B†	20B	64.22	0.69×	53.77	0.55×
	Self SD	-	13B	68.07	1.24×	60.38	1.12×
	Medusa	760B	13.8B	26.08	1.53×	15.55	1.21×
	EESD (+Beta-TS)	481M	13.5B	43.01	1.57×	32.23	1.25×
	EESD (+Cali-TS)	481M+262M‡	13.7B	43.53	**1.59**×	32.50	**1.27**×

Table 1: Evaluation on Gsm8k and XSum with different methods. **Speedup** signifies the acceleration effect in comparison with the auto-regression method. † For all Vanilla SD, we use the Homologous 7B model as the draft model, and we think that this 7B model needs to be trained. ‡ Model prediction requires the training of additional parameters. Results are statistically significant with respect to all baselines (all p-value < 0.005).

Model	Method	Trainable Params	Humaneval	
			HM	Speedup
LLaMA-2-13B	Vanilla SD	7B	87.32	0.97×
	Self SD	-	79.93	1.36×
	Medusa	760M	26.67	1.61×
	EESD (+Beta-TS)	481M	61.43	2.08×
	EESD (+Cali-TS)	481M+262M	62.87	**2.15**×
CodeLLaMA-2-13B	Vanilla SD	7B	91.12	1.09×
	Self SD	-	83.51	1.38×
	Medusa	761M	49.14	1.97×
	EESD (+Beta-TS)	481M	68.94	2.21×
	EESD (+Cali-TS)	481M+262M	70.15	**2.45**×

Table 2: Evaluation on Humaneval with different speculative decoding methods. Results are statistically significant with respect to all baselines (all p-value < 0.005).

Method	Gsm8k (HM)	XSum (HM)
Vanilla SD	0.96×	0.77×
EESD (Beta-TS)	1.91× (57.22)	1.84× (55.46)
w/o Early-exiting Layer	1.18× (23.69)	1.15× (23.10)
w/o Self-Distillation	1.82× (54.12)	1.73× (51.84)
EESD (Cali-TS)	2.04× (58.97)	1.92× (56.45)
w/o Sampling-Prediction	1.82× (54.03)	1.78× (53.49)
w/o Model-Prediction	1.88× (57.10)	1.82× (55.38)
EESD w/o TS†	1.66× (44.76)	1.58× (41.77)

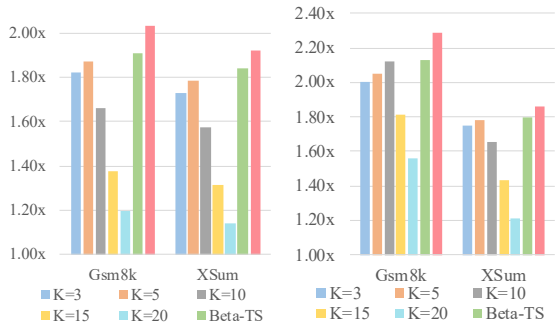
Table 3: Ablation studies of different components based on LLaMA-2-13B. We exhibit the Speedup on Gsm8k and XSum, and also release the HM value in parentheses. † We set the drafting step K at 10. Other models yield similar patterns to LLaMA-2-13B.

capability of the draft model, indicated by a higher HM value, does not necessarily result in higher speedup. It is essential to consider the generation speed of draft tokens, and our approach can strike an optimal balance between the two to achieve higher speedup (detailed in Appendix D).

5 Analysis and Discussion

5.1 Ablation Study

To elucidate the impact of different components within our approach, we conduct a series of ablation studies. In Table 3, we exhibit experimental results, and several significant insights can be inferred. **First**, we notice a substantial decrement in the model's performance when we replace the TS control with a fixed K value, which signifies the effectiveness of our proposed method for managing the generation of draft tokens. **Second**, similar to the prior approach, we introduce a trainable lm head just after the first-N layers, dispensing with the Early-exiting Layer. However, such a modification result in a significant decline in the model's performance, strongly indicating the fundamental role of the Early-exiting Layer in maintaining the quality of draft tokens. **Third**, a noteworthy observation is that our approach attains commendable results solely with only open-source data, especially on XSum. Furthermore, the performance can



(a) LLaMA-2-13B (b) LLaMA-2-70B

Figure 3: We evaluate the speedup in generating 512 tokens using the EESD method at varying K values.

Model	Method	Gsm8k	XSum
LLaMA-2-70B	Vanilla SD	1.88×	1.46×
	+ Beta-TS	2.02× (+0.14)	1.67× (+0.21)
	Self SD	1.37×	1.23×
	+ Beta-TS	1.44× (+0.07)	1.25× (+0.04)
LLaMA-2-13B	Vanilla SD	0.96×	0.77×
	+ Beta-TS	0.99× (+0.03)	0.85× (+0.08)
	Self SD	1.37×	1.35×
	+ Beta-TS	1.41× (+0.04)	1.40× (0.05)

Table 4: Speedup of other SD methods with TS control mechanism.

be improved with the addition of self-distillation, demonstrating the utility of data generated by original LLM. **Fourth**, within the Cali-TS approach, the role of sample prediction surpasses that of model prediction, and an integration of both can yield more optimal results.

5.2 Can the TS control mechanism predict the optimal drafting steps?

To investigate the ability of the TS control mechanism to automatically determine the quantity of draft tokens in each round, we conducted experiments on the effects of varying drafting steps K. As illustrated in Figure 3, the optimal K value differs across models and datasets, but the TS with Beta distribution consistently slightly exceeds the effect of the optimal K value. Moreover, with the boost from the model prediction, the TS with calibration can achieve a better acceleration effect. The experiment confirmed that the TS control mechanism can adaptively predict the optimal length for generating draft tokens in each round.

5.3 The generality of TS control mechanism

To verify the generality and effectiveness of the proposed TS control mechanism, we further apply it to other SD models instead of a pre-defined K. The

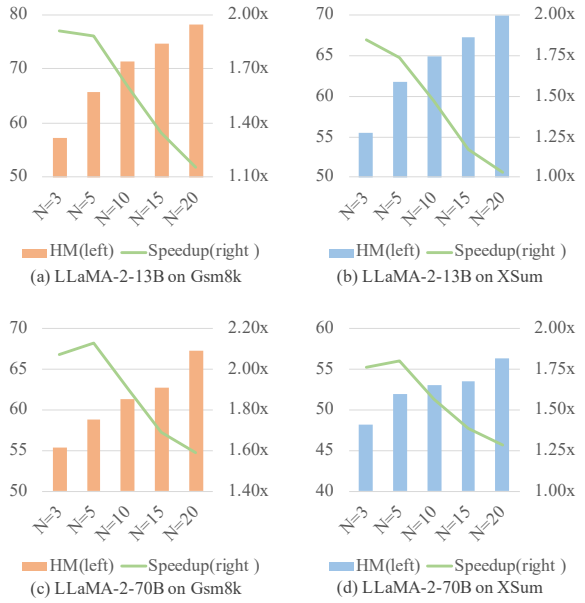
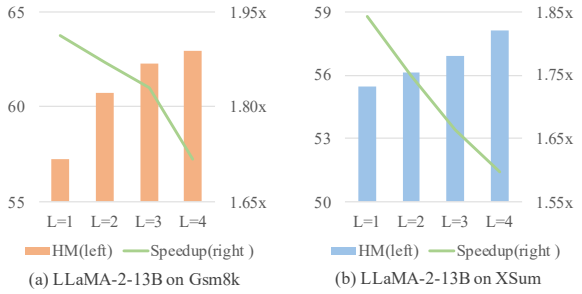


Figure 4: Effect of the different first-N layers. We valuate EESD (+Beta-TS) across varying N values of the first-N layers.

results are reported in Table 4. According to the results, we can observe that TS control mechanism could be easily integrated into other SD methods to lift their performances. Note that the results in Table 4 are different from the results of w/o TS in ablation study. In ablation study, we set K at 10, which is not a superior setting, and as shown in Figure 3, K=5 is a better setting for the EESD of LLaMA-2-13B. However, for vanilla SD and self SD, K=10 is a suitable setting.

5.4 Effect of the first-N layers

Our experiments explore the impact of varying the number of first-N layers. As shown in Figure 4, using more layers improves the quality of the draft tokens, as measured by a higher HM value. However, end-to-end speedup does not correspondingly increase along with draft quality. This suggests that the extra time required to generate draft tokens with more layers offsets some of the end-to-end speedup. The results indicate that layer augmentation only leads to slight improvements in the quality of draft tokens. Therefore, utilizing fewer layers for generating draft tokens proves to be an effective strategy. In addition, for larger models, such as 70B, the value of N needs to be slightly larger. And it is empirically suggested that N should be 5%-10% of the total number of LLM layers.



(a) LLaMA-2-13B on Gsm8k (b) LLaMA-2-13B on XSum

Figure 5: Effect of varying the number of Early-exiting layers.

Model	Method	Seq.	Loaded Params	Trainable Params	Num. of GPU	Batch Size	Time per batch
70B	Vanilla SD	4k	7B	7B	8	64	4.1s
	Medusa	4k	70.0B	1.32B	8	64	30.0s
	EESD	4k	5.6B	1.12B	8	64	2.3s
13B	Vanilla SD	2k	7B	7B	2	64	15.5s
	Medusa	2k	13.6B	760M	2	64	9.5s
	EESD	2k	1.6B	481M	2	64	1.6s

Table 5: Training efficiency of three methods on A100-80GB for LLaMA-2-13B and LLaMA-2-70B.

5.5 One Transformer layer is best for Early-exiting layer?

As demonstrated in Table 3, it has been proven that adding one Transformer layer after the First-N layers significantly improves the draft model's performance. To further investigate, we evaluate the effect of increasing the number of Transformer layers. Figure 5 illustrates that augmenting the number of Transformer layers following the First-N layers does yield an improvement in draft token quality. However, because degree of this improvement is relatively small, it results in a reduction in the overall end-to-end speedup. Therefore, the experiment indicates that a single Transformer layer is enough to ensure the quality of draft tokens, and perfectly balance the quality and generation speed of draft tokens to achieve the optimal end-to-end speedup.

5.6 Training Efficiency

We compare training efficiency of three methods, which are tested on NVIDIA A100-80G GPUs. We set batch size to 64 and used SlimPajama datasets to train these models. As shown in Table 5, EESD only requires loading the parameters of First-N layers and Early-exiting layer, while Medusa requires loading all parameters of LLM and Medusa heads. Notably, during training, although both Medusa and EESD only update a portion of parameters, Medusa requires each sample to be computed across the whole LLM network. In contrast, EESD only needs to compute across First-N layers. Consequently, compared to Medusa and Vanilla SD, EESD significantly reduces in both training time and memory consumption.

Model	Method	Gsm8k	XSum
LLaMA-2-70B	EESD (Beta-TS)	2.13×	1.80×
	+ Tree Attention	2.31× (+0.18)	1.91× (+0.11)
	EESD (Cali-TS)	2.29×	1.86×
	+ Tree Attention	2.48× (+0.19)	1.96× (+0.10)
LLaMA-2-13B	EESD (Beta-TS)	1.91×	1.84×
	+ Tree Attention	2.04× (+0.13)	1.89× (+0.05)
	EESD (Cali-TS)	2.04×	1.92×
	+ Tree Attention	2.18× (+0.14)	2.12× (+0.20)

Table 6: Speedup of EESD with implementing tree attention. We generate multiple draft token candidates and only retain the result consistent with the target model's top-1 token on verifying process.

5.7 Implement Tree Attention

Tree attention has been a prevalent technique in inference acceleration (Miao et al., 2023; Spector and Ré, 2023). This technique functions by structuring numerous draft token candidates within a tree framework, allowing the LLM to concurrently verify several potential draft sequences through parallel decoding. It significantly increases the acceptance rate, thereby augmenting the overall speed of end-to-end generation. As shown in Table 6, we can easily implement the tree attention mechanism to EESD, resulting in significant increases in speed. It can achieve up to 2.48× and 1.96× times speedup on LLaMA-2-70B, as well as up to 2.18× and 2.12× times speedup on LLaMA-2-13B.

6 Conclusion

In this work, we propose EESD, a novel method designed to lossless accelerate LLM by leveraging its first-N layers for generating draft tokens and employing Thompson Sampling to regulate this process. Specifically, we introduce an Early-exiting layer after first-N layers and train it using self-distillation, which strike an optimal balance between efficiency and performance of draft token generation. Furthermore, we devise a novel hybrid method that effectively combines model prediction and sampling prediction, resulting in remarkable generation speed enhancement. After conducting exhaustive experiments, the results demonstrate that EESD not only achieves a significant speedup but also substantially reduces both training time and memory consumption, compared to previous speculative decoding methods.

Limitations

In this section, we discuss the limitations of our work as follows. First, while we have given an empirical suggestion for the setting of N value in the First-N layers, we have not thoroughly studied the function of these first layers and how they affect the final outputs. We believe that a more detailed investigation of this is helpful for choosing the optimal N value. As such, we will conduct this research in future work. Second, we propose a model for predicting whether the draft token is consistent with the LLM's token in Section 3.3. However, this model has a large number of parameters, which is not very friendly for training and deployment. Therefore, we plan to refine the model's structure to improve its efficiency in future work.

References

Sangmin Bae, Jongwoo Ko, Hwanjun Song, and Se-Young Yun. 2023. Fast and robust early-exiting framework for autoregressive language models with synchronized parallel decoding. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 5910–5924. Association for Computational Linguistics.

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

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. 2020. Depth-adaptive transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10323–10337. PMLR.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. GPTQ: accurate post-training quantization for generative pre-trained transformers. *CoRR*, abs/2210.17323.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2023. Knowledge distillation of large language models. *CoRR*, abs/2306.08543.

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

Parsa Kavehzadeh, Mojtaba Valipour, Marzieh Tahaei, Ali Ghodsi, Boxing Chen, and Mehdi Rezagholizadeh. 2023. Sorted llama: Unlocking the potential of intermediate layers of large language models for dynamic inference using sorted fine-tuning (soft). *CoRR*, abs/2309.08968.

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

Liunian Harold Li, Jack Hessel, Youngjae Yu, Xiang Ren, Kai-Wei Chang, and Yejin Choi. 2023. Symbolic chain-of-thought distillation: Small models can also "think" step-by-step. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 2665–2679. Association for Computational Linguistics.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. 2023. AWQ: activation-aware weight quantization for LLM compression and acceleration. *CoRR*, abs/2306.00978.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2023. LLM-QAT: data-free quantization aware training for large language models. *CoRR*, abs/2305.17888.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *CoRR*, abs/2305.11627.

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

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1797–1807. Association for Computational Linguistics.

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

Mingdong Ou, Nan Li, Cheng Yang, Shenghuo Zhu, and Rong Jin. 2019. Semi-parametric sampling for stochastic bandits with many arms. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 7933–7940. AAAI Press.

Yi Peng, Miao Xie, Jiahao Liu, Xuying Meng, Nan Li, Cheng Yang, Tao Yao, and Rong Jin. 2019. A practical semi-parametric contextual bandit. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 3246–3252. ijcai.org.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. 2022. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Aleksandrs Slivkins. 2019. Introduction to multi-armed bandits. *CoRR*, abs/1904.07272.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. 2023. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama.

Benjamin Spector and Christopher Ré. 2023. Accelerating LLM inference with staged speculative decoding. *CoRR*, abs/2308.04623.

Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *23rd International Conference on Pattern Recognition, ICPR 2016, Cancún, Mexico, December 4-8, 2016*, pages 2464–2469. IEEE.

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

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

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language

model pre-training via structured pruning. *CoRR*, abs/2310.06694.

Guangxuan Xiao, Ji Lin, Mickaël Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 38087–38099. PMLR.

Chen Zhang, Yang Yang, Jiahao Liu, Jingang Wang, Yunsen Xian, Benyou Wang, and Dawei Song. 2023a. Lifting the curse of capacity gap in distilling language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 4535–4553. Association for Computational Linguistics.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena.

A Thompson Sampling with Beta Distribution

Given that the samples follow a Bernoulli distribution, we can infer using Bayes' theorem that when the prior distribution is Beta, the posterior distribution is also Beta. This phenomenon, known as a conjugate distribution, means that the prior and posterior share the same distribution function but with different parameters. Using a conjugate distribution greatly simplifies the computational derivation, leading us to select the Beta distribution as the prior. As shown in Algorithm 2, we implement Thompson Sampling algorithm with Beta distribution to iteratively estimate the value of θ. The prior distribution of θ is Beta, and as previously described, the posterior distribution of θ is also Beta.

Algorithm 2 Thompson Sampling with Beta Distribution Algorithm

Require: Target Model M_t; Draft Model M_d; Max Generation Length L; Hyperparameters α_0,β_0; Input Prompt $\{x_0, ..., x_n\}$.

Initialize prior probability $Beta(\theta; \alpha_0, \beta_0)$.
Initialize the result set $Q_g \leftarrow \{x_0, ..., x_n\}$ and $t \leftarrow 0$.
while t < L **do**
  Initialize the draft model result set and i, $Q_d \leftarrow Null, i \leftarrow 0$.
  while t + i < L **do**
    Get now token $x_i \leftarrow M_d(Q_g \bigcup Q_d)$.
    Add token x_i to set Q_d.
    Sample θ_{t+i} from $Beta(\theta; \alpha_t, \beta_t)$.
    Sample $\chi \in \{0, 1\}$ from Bernoulli distribution $P_B(\theta_{t+i})$.
    $i \leftarrow i + 1$.
    if $\chi = 0$ **then**
      Break
    end if
  end while
  Verify the results Q_d by Target Model M_t and get Q_v received by M_t, $Q_v \subseteq Q_d$.
  Add the set Q_v to Q_g, $Q_g \leftarrow Q_g \bigcup Q_v$.
  Update t according to length of Q_v, $t \leftarrow t + |Q_v|$.
  Calculate $r \leftarrow |Q_v| - 1$.
  Calculate $n \leftarrow min(|Q_v| + 1, |Q_d|)$
  Update α_t, $\alpha_t \leftarrow \alpha_{\{t-|Q_v|\}} + r$.
  Update β_t, $\beta_t \leftarrow \beta_{\{t-|Q_v|\}} + (n - r)$.
end while
return Q_g

B Thompson Sampling with Calibration

In the previous section, we unveiled a Thompson Sampling algorithm with Beta distribution (Beta-TS). This method progressively update the parameters of Beta distribution to enhance the precision of the estimated θ. However, based on the Multi-Arm Bandit (MAB) theory, the early phase is more exploration-oriented, this predisposition can lead to a less accurate initial estimation of the θ. To escalate the efficiency of the Thompson Sampling method, we further propose a hybrid approach of model prediction and sampling prediction. In early stage, we rely more on model prediction to curtail the inaccuracies introduced by exploration. In later stages, as the sampling prediction converges, we calibrate the model prediction with the result of sampling to obtain an accurate estimate of θ. The details are illustrated in Algorithm 3.

Algorithm 3 Thompson Sampling with Calibration Algorithm

Require: Target Model M_t; Draft Model M_d; Max Generation Length L; Hyperparameters σ_M,σ_S,μ_0,σ_0; Input Prompt $\{x_0, ..., x_n\}$.

Initialize prior probability $\mathcal{N}(\mu_0, \sigma_0^2)$.
Initialize the result set $Q_g \leftarrow \{x_0, ..., x_n\}$, $n \leftarrow 0$ and $t \leftarrow 0$.
while t < L **do**
  Initialize the draft model result set and i, $Q_d \leftarrow Null, i \leftarrow 0$.
  while t + i < L **do**
    Get now token $x_i \leftarrow M_d(Q_g \bigcup Q_d)$.
    Add token x_i to set Q_d.
    Get model predict score, $\theta_M^{t+i} \leftarrow Sigmoid(MLP(W^i H_t^{(T)}, H_{t+i}^{(D)}))$.
    Calculate $\mu_{t+i} \leftarrow \frac{\sigma_S^2}{n\sigma_M^2+\sigma_S^2}\theta_M^{t+i} + \frac{n\sigma_M^2}{n\sigma_M^2+\sigma_S^2}\hat{\theta}_t$.
    Calculate $\sigma_{t+i}^2 \leftarrow \frac{\sigma_M^2\sigma_S^2}{\sigma_S^2+n\sigma_M^2}$.
    Sample θ_{t+i} from Gaussian distribution $\mathcal{N}(\mu_{t+i}, \sigma_{t+i}^2)$.
    Sample $\chi \in \{0, 1\}$ from Bernoulli distribution $P_B(\theta_{t+i})$.
    $i \leftarrow i + 1$.
    if $\chi = 0$ **then**
      Break
    end if
  end while
  Verify the results Q_d by Target Model M_t and get Q_v received by M_t, $Q_v \subseteq Q_d$.
  Add the set Q_v to Q_g, $Q_g \leftarrow Q_g \bigcup Q_v$.
  Update t according to length of Q_v, $t \leftarrow t + |Q_v|$.
  Update $n \leftarrow n + 1$.
  Update $\hat{\theta}_t$, $\hat{\theta}_t \leftarrow \frac{\hat{\theta}_{\{t-|Q_v|\}} * (t-|Q_v|+1) + |Q_v|}{t+1}$.
end while
return Q_g

According to the central limit theorem, when the sample size is sufficiently large, the sample mean adheres to a Gaussian distribution. Therefore, we make an assumption that sample mean $\widetilde{\chi}$ of χ in one drafting iteration follows a Gaussian distribution, i.e $\widetilde{\chi} = \frac{\chi_1+\chi_2+\chi_3+...+\chi_k}{k} \sim \mathcal{N}(\mu, \sigma_S^2)$. And According to the central limit theorem, σ_S equals $\sigma_\chi / sqrt(k)$, where σ_χ is the standard deviation of the random variable χ and we assume σ_χ

is known. In this case, we make the assumption that k is a fixed value and pre-determined by the user, which guarantees that $\widetilde{\chi}$ is independently and identically distributed. Furthermore, we employ a model to estimate the value of μ. It can be posited that μ follows a Gaussian distribution, characterized by the model's predicted value as the mean and the model's predicted error as the variance, i.e. $\mu \sim \mathcal{N}(\theta_M, \sigma_M^2)$, where θ_M is model's predicted value. Here, we presume that the model's predicted error is a known entity. As supported by Bayesian theory, when the random variable follows a Gaussian distribution with a known variance but an unknown mean and the prior distribution is also a Gaussian distribution, it satisfies the conjugate distribution. Therefore, the posterior distribution is following,

$$\begin{aligned} &P(\mu|D) \\ &\propto P(D|\mu, \sigma_S^2) P(\mu|\theta_M, \sigma_M^2) \\ &\propto \mathcal{N}(\frac{\sigma_S^2}{n\sigma_M^2 + \sigma_S^2}\theta_M^{t+i} + \frac{n\sigma_M^2}{n\sigma_M^2 + \sigma_S^2}\hat{\theta}_t, \frac{\sigma_M^2 \sigma_S^2}{\sigma_S^2 + n\sigma_M^2}) \end{aligned} \tag{9}$$

where n is the number of verification by LLM, $\{\sigma_M, \sigma_S, \hat{\theta_0}\}$ is pre-determined by the user and $\hat{\theta_t}$ is the observed sample mean of the random variable $\widetilde{\chi}$. Due to k is a constant, we set $\hat{\theta_t}$ to observed sample mean of the random variable χ, i.e.

$$\hat{\theta_t} = \frac{\hat{\theta}_{\{t-|Q_v|\}} * (t - |Q_v| + 1) + |Q_v|}{t+1}, \tag{10}$$

Carefully thinking Eq.(6) and Eq.(9), we can observe that when n is small, the mean μ tends to θ_M^{t+i}, and when n is large, μ tends to $\hat{\theta_t}$. This achieves the reduction of uncertainty in sampling prediction through model prediction in the early exploration stage. In the later stage, as the number of observed samples increases, the accuracy of $\hat{\theta_t}$ markedly enhances. Concurrently, μ draws closer to $\hat{\theta_t}$, thereby yielding more accurate prediction. Therefore, our proposed method of mixing model prediction and sampling prediction can outperform the Beta-TS algorithm.

C Training Details

We implement all experiments with the deep learning framework PyTorch on NVIDIA A100-80G GPUs. We set the learning rate to 1e-3 and the batch size to 64, for training EESD and Medusa. The hyperparameter settings we adopt are shown in Table 7

D Harmonic Mean Metrics

We believe that two indicators, the acceptance rate and the proportion of draft tokens, will affect the end-to-end acceleration effect. The acceptance rate, denoted as v_d, indicates the percentage of draft tokens that are accepted by the target model. It is calculated as follows,

$$v_d = \frac{N_{right}}{N_{all\_draft}}, \tag{11}$$

where N_{right} is the number of draft tokens that are accepted by the target model, and N_{all_draft} denotes the total count of tokens generated by the draft model. The proportion of draft tokens, denoted as r_d, represents the proportion of tokens that come from the draft model, which is calculated as follows,

$$r_d = \frac{N_{right}}{L}, \tag{12}$$

where L is the total number of tokens in the final output sequence.

Once we have computed the aforementioned two metrics, we can infer the speedup. The inference time of end-to-end generation can be calculated according to the following formula,

$$T = \frac{r_d * L}{v_d} T_d + (1 - r_d) * L * T_t, \tag{13}$$

Where T_d represents the time taken by the draft model to generate one token, and T_t represents the time taken by the target model to generate one token. We can compute speed of the method by $sp = \frac{L}{T}$ and speedup by $speedup = \frac{sp}{T_t}$. Therefore, by integrating Eq.(13), we obtain the following formula,

$$speedup = \frac{v_d}{(\alpha - v_d) * r_d + v_d}, \tag{14}$$

where α equals $\frac{T_d}{T_t}$. To achieve speedup greater than 1.00×, the term $\alpha - v_d$ must be less than zero, given that both v_d and r_d are positive values. The speedup of the method is influenced by the factors α, v_d and r_d. Under the premise that $\alpha < v_d$, ideally, α should be as low as possible, while v_d and r_d should be as high as possible. Both v_d and r_d are influenced by the quality of the draft tokens and the strategy for generating the draft tokens. A well-devised strategy for draft token generation can increase the values of v_d and r_d, but the upper bound is restricted by the inherent quality of the draft tokens themselves. Therefore, we use the harmonic mean of v_d and r_d to assess

Model	Method	Train Dataset	Seq.	# Epoch	Learning Rate	Batch Size	# GPUs
LLaMA-2-70B	Medusa	SlimPajama	4k	6	1e-3	64	8
	EESD	SlimPajama	4k	6	1e-3	64	8
LLaMA-2-70B-chat	Medusa	ShareGPT	4k	6	1e-3	64	8
	EESD	ShareGPT	4k	6	1e-3	64	8
LLaMA-2-13B	Medusa	SlimPajama	2k	4	1e-3	64	2
	EESD	SlimPajama	2k	4	1e-3	64	2
Vicuna-13B	Medusa	ShareGPT	2k	4	1e-3	64	2
	EESD	ShareGPT	2k	4	1e-3	64	2
CodeLLaMA-2-13B	Medusa	SlimPajama	16k	4	1e-3	64	8
	EESD	SlimPajama	16k	4	1e-3	64	8

Table 7: The hyperparameter values for EESD and Medusa training.

both the quality of the draft tokens and the strategy for generating them.

As shown in Table 8, 9, and 10, we present the v_d and r_d scores of each baseline method as well as our method across three benchmark evaluations, as detailed explanations of Table 1 and 2.

E Inference Time and Speed up

We present the comprehensive results of end-to-end inference time and tokens generated per second for the auto-regressive method, three speculative decoding methods and EESD. These results provide detailed explanations of the data shown in Table 1 and 2. These results, gathered from evaluations using the Gsm8k benchmark, are detailed in Table 8. Furthermore, we provide the additional results from the XSum dataset in Table 9, and from the Humaneval dataset in Table 10.

F Breakdown of Computation

Table 11 presents an analysis of the computational time required for EESD generating on 200 instances randomly selected from XSum. The results indicate that Cali-TS exhibits higher time consumption compared to Beta-TS during the sampling phase. However, Cali-TS significantly diminishes the time usage in the drafting and verification stages, due to its superior control over the draft token generation process. Consequently, Cali-TS can yield a lower total time consumption.

G Case Study

As shown in Figure 6, we demonstrated two Xsum samples. As described in Section 4.1, we adopt a 1-shot setting and use a greedy generation strategy. We observe that during the generation process of EESD, those draft tokens that are inconsistent with the original LLM's output will be discarded. This ensures that the final result generated by EESD is the same as auto-regression. Furthermore, we find that for samples with a high draft token acceptance rate, the EESD tends to generate longer draft sequence in one drafting round, as shown in example 1. Conversely, for samples with a lower acceptance rate, the EESD displays a tendency to generate shorter draft sequence, minimizing the quantity of discarded draft tokens, as shown in example 2. The examples in Figure 6 shows that our method is effective in adaptively determining the length of draft token generation, leading to significant improvement in the final end-to-end generation speed.

[INPUT]
Article: Masers were invented before the laser, but have languished in obscurity because they required high magnetic fields and difficult cooling schemes.
……
this type of maser could be used to detect some extraterrestrial intelligence that hasn't been detected."
Summary: Researchers have shown off a microwave-emitting version of the laser, called a maser, that works at room temperature.
Article: The 26-year-old England tight-head prop, who was sent off, pleaded guilty at a disciplinary hearing on Tuesday.
……
Brookes was dismissed in the 38th minute of Saints' 22-16 defeat when he charged into the ruck and struck the head of Newcastle hooker Scott Lawson with his shoulder.
Summary:

[Auto-regression OUTPUT]
The 26-year-old England tight-head prop, who was sent off, pleaded guilty at a disciplinary hearing on Tuesday. </s>

[EESD Generation Process]
- ROUND 1: The 26-year-old England
- ROUND 2: -head prop, who was dismissed
- ROUND 3: off for a year in the 201
- ROUND 4: pleaded guilty to a misdemean
- ROUND 5: the London
- ROUND 6: disciplinary hearing on Tuesday. </s>

[EESD Final OUTPUT]
The 26-year-old England tight-head prop, who was sent off, pleaded guilty at a disciplinary hearing on Tuesday. </s>

(a) EXAMPLE 1

[INPUT]
Article: Masers were invented before the laser, but have languished in obscurity because they required high magnetic fields and difficult cooling schemes.
……
this type of maser could be used to detect some extraterrestrial intelligence that hasn't been detected."
Summary: Researchers have shown off a microwave-emitting version of the laser, called a maser, that works at room temperature.
Article: Liberal Democrat AM Eluned Parrott said the Welsh government had "wasted" more than £52,000 on them at railway stations in south Wales.
She accused ministers of spending public money to "promote themselves before an election".
……
Formal consultation for the £600m programme to develop an integrated network of rail, bus and light rail services begins in 2016.
Summary:

[Auto-regression OUTPUT]
The Welsh government has been accused of wasting more than £52,000 on posters promoting the south Wales Metro. </s>

[EESD Generation Process]
- ROUND 1: The UK's largest broadband service provider
- ROUND 2: government has been working on
- ROUND 3: of spending £100m
- ROUND 4: taxpay
- ROUND 5: than £52,000 of
- ROUND 6: the rail system
- ROUND 7: reforms. </s>
- ROUND 8: the idea of a
- ROUND 9: Wales Government's plans
- ROUND 10: . </s>

[EESD Final OUTPUT]
The Welsh government has been accused of wasting more than £52,000 on posters promoting the south Wales Metro. </s>

(b) EXAMPLE 2

Figure 6: A visualization of the generation process of EESD with Cali-TS on LLaMA-2-70B. We present two examples from the XSum dataset, and demonstrate the input text, the text generated by the original LLM using auto-regression strategy, the generation process of EESD, and the final result generated by EESD. In EESD generation process, the green color represents the draft token that are accept by LLM, and the red color represents the rejected draft tokens, and gray color represents the draft tokens that will be discarded after the rejected token. And in the EESD final output, the green color represents tokens generated by the draft model, and the red color represents the tokens generated by the original LLM.

Model	Method	Harmonic Mean v_d	Harmonic Mean r_d	Harmonic Mean HM	Inference Time (/s)	Speed (token/s)	Speedup
LLaMA-2-70B	Auto-regressive	-	-	-	56.32	9.10	1.00×
	Vanilla SD	0.74	0.88	80.35	29.89	17.13	1.88×
	Self SD	0.90	0.70	78.64	41.09	12.46	1.37×
	Medusa	0.25	0.50	33.75	32.56	15.72	1.73×
	EESD (+Beta-TS)	0.52	0.68	58.79	26.44	19.36	2.13×
	EESD (+Cali-TS)	0.56	0.71	62.25	24.61	20.80	**2.29**×
LLaMA-2-70B-chat	Auto-regressive	-	-	-	56.53	9.06	1.00×
	Vanilla SD	0.52	0.83	63.90	39.17	13.07	1.44×
	Self SD	0.66	0.70	67.99	49.89	10.26	1.13×
	Medusa	0.16	0.39	22.86	39.97	12.81	1.41×
	EESD (+Beta-TS)	0.39	0.62	47.76	31.57	16.22	1.79×
	EESD (+Cali-TS)	0.39	0.62	48.23	31.04	16.49	**1.82**×
LLaMA-2-13B	Auto-regressive	-	-	-	21.93	23.35	1.00×
	Vanilla SD	0.81	0.89	84.59	22.78	22.48	0.96×
	Self SD	0.90	0.73	80.53	16.04	31.92	1.37×
	Medusa	0.24	0.47	31.71	12.42	41.22	1.77×
	EESD (+Beta-TS)	0.50	0.67	57.22	11.46	44.68	1.91×
	EESD (+Cali-TS)	0.52	0.69	58.97	10.77	47.54	**2.04**×
Vicuna-13B	Auto-regressive	-	-	-	22.26	23.00	1.00×
	Vanilla SD	0.52	0.83	64.22	32.40	15.80	0.69×
	Self SD	0.76	0.62	68.07	17.91	28.59	1.24×
	Medusa	0.19	0.42	26.08	14.57	35.14	1.53×
	EESD (+Beta-TS)	0.34	0.58	43.01	14.16	36.16	1.57×
	EESD (+Cali-TS)	0.35	0.59	43.53	14.00	36.57	**1.59**×

Table 8: The detailed evaluation results on Gsm8k with different methods of Table 1. We present the result from our assessment of Harmonic Mean, inference time, and the speed of end-to-end generation. Additionally, we also present the speedup in comparison with the auto-regression method.

Model	Method	Harmonic Mean v_d	Harmonic Mean r_d	Harmonic Mean HM	Inference Time (/s)	Speed (token/s)	Speedup
LLaMA-2-70B	Auto-regressive	-	-	-	62.59	8.18	1.00×
	Vanilla SD	0.57	0.83	67.30	42.85	11.95	1.46×
	Self SD	0.80	0.60	68.60	51.72	9.90	1.21×
	Medusa	0.19	0.42	25.69	44.21	11.58	1.42×
	EESD (+Beta-TS)	0.44	0.63	51.91	34.81	14.71	1.80×
	EESD (+Cali-TS)	0.46	0.65	53.41	33.58	15.25	**1.86**×
LLaMA-2-70B-chat	Auto-regressive	-	-	-	62.57	8.18	1.00×
	Vanilla SD	0.70	0.66	62.75	44.98	11.38	1.39×
	Self SD	0.66	0.70	68.38	53.93	9.49	1.16×
	Medusa	0.12	0.31	17.02	52.01	9.85	1.20×
	EESD (+Beta-TS)	0.32	0.55	40.73	41.32	12.39	1.51×
	EESD (+Cali-TS)	0.33	0.57	41.85	40.43	12.66	**1.55**×
LLaMA-2-13B	Auto-regressive	-	-	-	22.46	22.80	1.00×
	Vanilla SD	0.67	0.86	75.63	29.09	17.60	0.77×
	Self SD	0.87	0.70	77.61	16.63	30.79	1.35×
	Medusa	0.18	0.41	25.03	14.67	34.90	1.53×
	EESD (+Beta-TS)	0.48	0.66	55.46	12.18	42.04	1.84×
	EESD (+Cali-TS)	0.49	0.67	56.45	11.69	43.80	**1.92**×
Vicuna-13B	Auto-regressive	-	-	-	22.82	22.44	1.00×
	Vanilla SD	0.41	0.79	53.77	40.71	12.58	0.55×
	Self SD	0.69	0.54	60.38	20.14	25.42	1.12×
	Medusa	0.11	0.29	15.55	18.28	28.01	1.21×
	EESD (+Beta-TS)	0.24	0.47	32.23	17.96	28.51	1.25×
	EESD (+Cali-TS)	0.25	0.48	32.50	17.75	28.85	**1.27**×

Table 9: The detailed evaluation results on XSum with different methods of Table 1. We present the result from our assessment of Harmonic Mean, inference time, and the speed of end-to-end generation. Additionally, we also present the speedup in comparison with the auto-regression method.

Model	Method	Harmonic Mean v_d	Harmonic Mean r_d	Harmonic Mean HM	Inference Time (/s)	Speed (token/s)	Speedup
LLaMA-2-13B	Auto-regressive	-	-	-	21.83	23.45	1.00×
	Vanilla SD	0.86	0.89	87.32	22.60	22.65	0.97×
	Self SD	0.89	0.72	79.93	16.09	31.82	1.36×
	Medusa	0.19	0.42	26.67	13.57	37.73	1.61×
	EESD (+Beta-TS)	0.54	0.72	61.43	10.52	48.67	2.08×
	EESD (+Cali-TS)	0.55	0.73	62.87	10.14	50.49	**2.15**×
CodeLLaMA-2-13B	Auto-regressive	-	-	-	22.82	22.44	1.00×
	Vanilla SD	0.92	0.90	91.12	20.84	24.57	1.09×
	Self SD	0.92	0.76	83.51	16.58	30.88	1.38×
	Medusa	0.45	0.55	49.14	11.56	44.29	1.97×
	EESD (+Beta-TS)	0.64	0.75	68.94	10.31	49.66	2.21×
	EESD (+Cali-TS)	0.65	0.76	70.15	9.32	54.94	**2.45**×

Table 10: The detailed evaluation results on Humaneval with different methods of Table 2. We present the result from our assessment of Harmonic Mean, inference time, and the speed of end-to-end generation. Additionally, we also present the speedup in comparison with the auto-regression method.

Model	Method	Drafting	Verification	Sampling	Others
LLaMA-2-70B	EESD(+Beta-TS)	5.517s	25.203s	0.017s	4.076s
	EESD(+Cali-TS)	5.022s	24.106s	0.426s	4.023s
LLaMA-2-13B	EESD(+Beta-TS)	3.656s	7.986s	0.014s	0.526s
	EESD(+Cali-TS)	3.436s	7.392s	0.355s	0.507s

Table 11: Breakdown of computational time (seconds) for EESD on 200 instances randomly sampled from XSum. We set final output sequence length at 512.