SPIDE: Serial and Parallel Intertwined Speculative Decoding

Wenru Xu[1] **Peixuan Xu**[1] **Ziqi Yang**[1] **Ming Hu**[2] **Zihui Wang**[3]
Jianzhong Qi[4] **Rongshan Yu**[1] **Xiaoliang Fan**[1*] **Cheng Wang**[1]
[1] Xiamen University [2] East China Normal University
[3] Peng Cheng Laboratory [4] The University of Melbourne
{xuwenru, fanxiaoliang}@xmu.edu.cn

Abstract

Speculative Decoding (SD) reduces inference latency for Large Language Models (LLMs) by leveraging an efficient draft model to generate candidate tokens, which are subsequently verified by the target model. To enhance acceleration while reducing the LLM usage costs, we propose **S**erial and **P**arallel **I**ntertwined Speculative **DE**coding (SPIDE) — a novel training-free SD framework that orchestrates dynamic alternation combining serial dynamic drafting with parallel draft verification. We maintain a confidence-acceptance mapping table during the decoding process. In the serial dynamic drafting module, we leverage this table to evaluate the reliability of the draft sequence and adjust draft lengths adaptively. In the parallel draft verification module, we alleviate drafting-termination conflicts that compromise efficiency, and we update the mapping table synchronously. We conduct experiments on diverse model pairs and text generation tasks to assess the effectiveness of SPIDE. Compared with autoregressive decoding, SPIDE speeds up by **3.25×** on average and up to **4.56×**. Compared with vanilla SD, SPIDE only increases the LLM usage cost by **8.2%** on average, while bringing an additional **67.7%** speedup on average.

1 Introduction

Transformer-based Large Language Models (LLMs) rely on autoregressive decoding techniques, generating tokens sequentially often with noticeable latency issues. Speculative decoding (SD) has been investigated to mitigate LLM inference latency without compromising generation quality (Chen et al., 2023; Leviathan et al., 2023; Miao et al., 2023). SD employs a draft model to propose multiple speculative tokens per decoding step, and then runs the target model once to concurrently verify all candidate tokens. Since the draft model is much more efficient than the target model, SD significantly speeds up inference.

Early vanilla SD methods employ a static and sequential strategy (Chen et al., 2023; Leviathan et al., 2023). As shown in Figure 1(a), such methods draft a fixed number (γ) of tokens each time and execute drafting/verification in strict sequence, resulting in a limited speedup. To address the inefficiency caused by static drafting, dynamic drafting methods adaptively determine the draft length (Mamou et al., 2024; Zhang et al., 2024b). However, the serial "draft-then-verify" structure still limits the speedup of these methods.

Recently, parallel SD frameworks such as PEARL (Liu et al., 2025) optimize this workflow by introducing pre-verify/post-verify strategies, achieving better acceleration. As is shown in Figure 1(b), "pre-verify" checks the first token of the current draft, "post-verify" checks the previous draft. However, parallel SD faces two key challenges. **First**, when drafting is much faster than verification, acceleration is constrained by the need for a large value of γ, which is set to the drafting speed divided by the verification speed to minimize the wait time. For example, in Figure 1(b), only after eight tokens had been drafted it started to verify the wrong fourth token, which wasted the time for drafting the fifth to the eighth invalid tokens. **Second**, strict parallel execution forces continuous LLM engagement. In Figure 1(b), the target model M_p is repeatedly invoked up to five times without intermittency, verifying the first token of the current draft sequence or the tokens of the preceding draft sequence. Given that the target model is usually a large-scale LLM which is more expensive to invoke (Ong et al., 2024), this continuous engagement inevitably leads to high LLM usage costs.

To enhance acceleration while reducing the LLM usage costs, we propose an efficient and training-free framework, named **S**erial and **P**arallel

*Corresponding author

Findings of the Association for Computational Linguistics: ACL 2026, pages 20762–20779

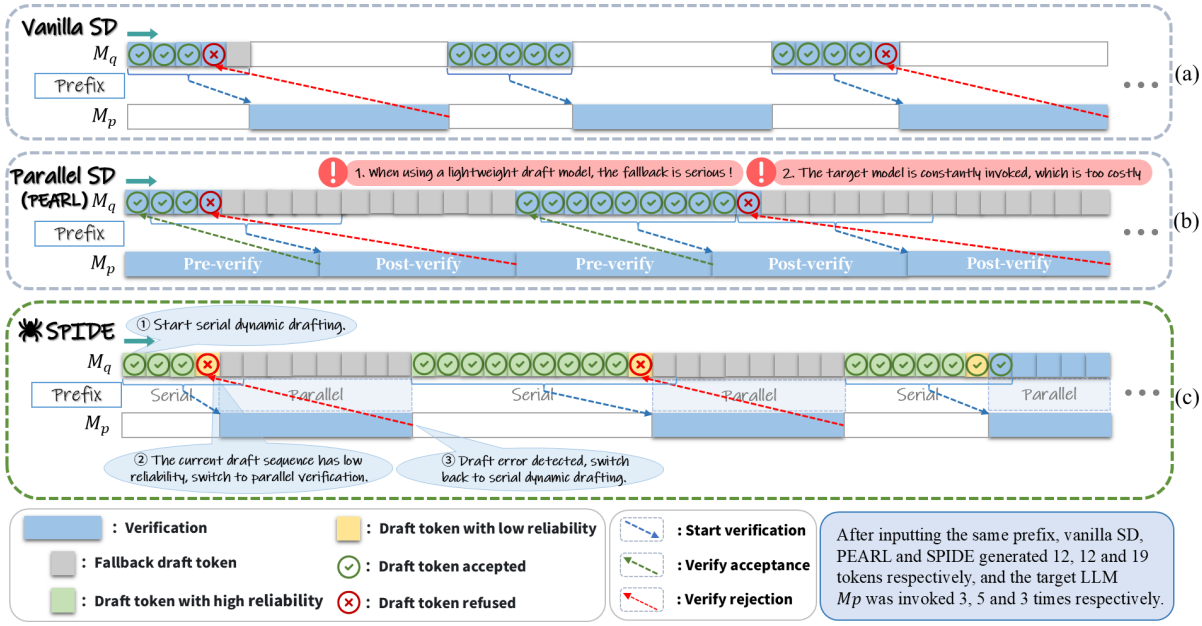


Figure 1: Comparison between vanilla SD, parallel SD, and our SPIDE (M_q and M_p stand for the draft model and the target model, respectively.): (a) Vanilla SD drafts a fixed number of tokens each time and sequentially execute drafting and verification, often with limited speedup. (b) Parallel SD (i.e., PEARL (Liu et al., 2025)) adopts the "pre-verify" strategy to verify the first token of the current draft sequence, and the "post-verify" strategy to verify the previous draft sequence. (c) Our SPIDE dynamically orchestrates serial dynamic drafting and parallel draft verification. This intertwined design achieves superior acceleration while reducing LLM usage costs for verifications.

Intertwined Speculative **DE**coding (SPIDE), with a *serial dynamic drafting* module and a *parallel draft verification* module. We divide the confidence score of draft tokens into intervals, record the acceptance rate of draft tokens in each interval, and maintain a *confidence-acceptance rate mapping table* during the decoding process. In the *serial dynamic drafting* module, we use this mapping table to evaluate the reliability of the current draft sequence after each drafting step. When the reliability is lower than our preset threshold, we utilize the *parallel draft verification* module to mitigate the inefficiency from premature draft termination. The mapping table is iteratively updated after each verification. In summary, our contributions are as follows:

- We propose a novel training-free decoding framework that dynamically orchestrates serial dynamic drafting tokens and parallel draft verification. Unlike existing solutions that parallelize the execution of drafting and verification constantly, SPIDE's intertwined design achieves superior acceleration while reducing LLM usage costs.
- We introduce a dynamic mapping table that maps the token confidence to the acceptance rate during the decoding process, and use this table to guide the setting of the drafting length dynamically.
- We conduct comprehensive experiments evaluating SPIDE against baselines across multiple model pairs and different text generation tasks. The results show that SPIDE outperforms auto-regressive decoding and vanilla SD, accelerating token generation by up to **4.56**× and **2.12**×, respectively. Compared to existing parallel SD framework, SPIDE achieves a faster speedup of up to **19.6%** while reducing the additional LLM usage cost of parallelism by **83.7**% on average.

2 Related Works

LLM Inference Acceleration Accelerating LLM inference is a trending topic with practical significance. Key solutions include: (1) Efficient structure design (Chitty-Venkata and Somani, 2022), leveraging techniques such as Mixture-of-Experts (MoE) (Shazeer et al., 2017) and low-rank decomposition (Kiruthika et al., 2025) to reduce computational complexity and memory

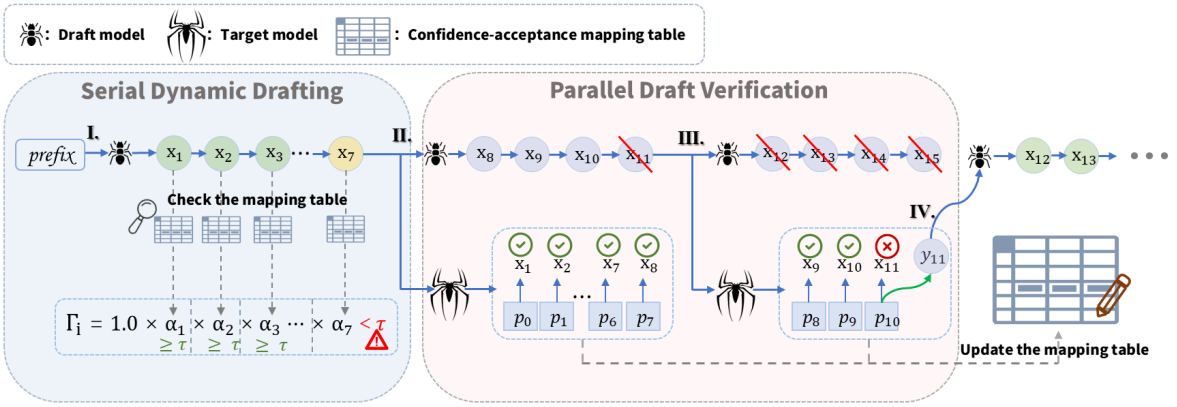


Figure 2: Illustration of our SPIDE: I. After inputting the prefix, the draft model M_q drafts $[x_1, x_2, \ldots, x_7]$ sequentially. Each time x_i is generated, we use the mapping table to map the confidence of x_i to an acceptance probability α_i, then use α_i to update the reliability score Γ_i of the draft sequence. II. After drafting x_7, the sequence reliability is lower than a preset threshold τ, we start parallel draft verification. While M_q continues to draft $[x_8, x_9, x_{10}, x_{11}]$, M_p verifies $[x_1, x_2, \ldots, x_8]$. III. $[x_1, x_2, \ldots, x_8]$ is fully accepted by M_p, we update the mapping table and continue with the parallel draft verification. M_q then drafts $[x_{12}, x_{13}, x_{14}, x_{15}]$ and M_p verifies $[x_9, x_{10}, x_{11}, x_{12}]$. IV. x_{11} is rejected by M_p, we update the mapping table and correct sampling y_{11}, then switch back to serial dynamic drafting module.

footprint; (2) Model compression (Wang et al., 2024), applying quantization (Lang et al., 2024), pruning (Liu et al., 2018) or knowledge distillation (Hinton et al., 2015) to reduce model size and computational requirements while preserving accuracy; and (3) Inference engine optimization (Park et al., 2025), dedicated to accelerating LLMs' forward pass execution.

Speculative Decoding Building on the success of SD in reducing LLM inference latency (Chen et al., 2023; Leviathan et al., 2023; Spector and Re, 2023; Zhao et al., 2024b), substantial efforts have been put into optimizing SD implementation. A few studies focus on the draft approach: (1) MCSD (Yang et al., 2024) proposes to sample multiple draft tokens and organize them in batches for verification; (2) GLIDE (Du et al., 2024) improves the draft model by reusing cached keys and values in the target model; (3) CS-Drafting (Chen et al., 2024) improves efficiency by combining vertical and horizontal cascading of draft models; (4) MEDUSA (Cai et al., 2024) accelerates by adding extra decoding heads to predict multiple subsequent tokens in parallel; (5) EAGLE (Li et al., 2024) conducts autoregressive prediction in the feature layer of the LLM and introduces a token sequence with offset as input to address feature uncertainty, thereby accelerating inference; (6) Other methods employ lightweight variants of the target model as the draft model, including quantization (Zhao et al., 2024a), early exit (Elhoushi et al., 2024), and forward filling (Monea et al., 2023).

Further enhancements target the dynamics of the inference process. This line of works is the most relevant to ours. Methods such as SVIP (Zhang et al., 2024b) adapt the draft length based on token entropy, while PEARL (Liu et al., 2025) achieves adaptive length and reduces pipeline stalls by parallelizing drafting and verification. Draft & Verify (Zhang et al., 2023) proposes an adaptive draft-exiting mechanism based on dynamic confidence threshold. SpecDec++ (Huang et al., 2024) and DISCO (Mamou et al., 2024) respectively train an acceptance prediction head and a basic classifier to achieve dynamic drafting. A comprehensive review over the works on enhancing SD can be found from (Xia et al., 2024).

Our SPIDE framework distinguishes itself from the above-mentioned related works through two primary innovations: (1) Instead of directly using a confidence metric (i.e., entropy or raw probability), we convert confidence levels into an acceptance rate to control the draft lengths more precisely that alleviate the loss of efficiency caused by premature or delayed termination of drafting. Our method does not require training and only needs to maintain a mapping table. (2) Our framework moves away from rigid serial or parallel processing. It alternates between serial and parallel states during decoding based on the current context. This flexible switching mechanism effectively enhances the

overall efficiency while maintaining low calling costs to the target model.

3 Method

3.1 Preliminaries

Speculative Decoding SD is an inference paradigm designed to reduce the high latency of autoregressive decoding in LLMs. At each step, SD generates multiple draft tokens that are subsequently verified in parallel. Specifically, given a prefix $\mathbf{x}$, SD starts by invoking a draft model M_q to sample a draft sequence of tokens with length γ, denoted as $x_1, \cdots, x_\gamma$, where $x_i \sim q(x|x_1, \cdots, x_{i-1}, \mathbf{x})$. The draft tokens, along with the prefix, are then sent to the target model M_p to obtain their output distribution $p(x|x_1, \cdots, x_i, \mathbf{x})$ in parallel. Finally, the draft tokens are verified sequentially from x_1 to x_γ. To verify token x_i, a speculative sampling algorithm is employed to determine whether to accept x_i or not, based on $q(x|x, \cdots, x_{i-1}, \mathbf{x})$ and $p(x|x, \cdots, x_{i-1}, \mathbf{x})$. Once a token is rejected, the next verification terminates and the algorithm returns a new token as the endpoint. If all tokens are accepted, there is an additional token sampled from $p(x|x_1, \cdots, x_\gamma, \mathbf{x})$ as the endpoint.

Speculative Sampling The importance of speculative sampling lies in its guarantee of distributional equivalence: unconditional acceptance of draft tokens would violate the output distribution relative to the target model M_p. Speculative sampling employs an algorithm that accepts x with a probability of $\min(1, \frac{p(x)}{q(x)})$, resulting in an overall acceptance rate of $\sum_x \min(p(x), q(x))$. In the event that x is rejected, a new token is sampled for correction, i.e., sampled from the distribution defined by:

$$p'(x) = \frac{\max(0, p(x) - q(x))}{\sum_x \max(0, p(x) - q(x))}. \quad (1)$$

It has been shown that speculative sampling preserves the output distribution consistent with the target model (Leviathan et al., 2023; Chen et al., 2023).

3.2 SPIDE

The framework of SPIDE is shown in Figure 2. It consists of two main modules: serial dynamic drafting and parallel draft verification. In addition, we design a confidence-acceptance mapping table that can be updated in real time during the decoding process. The serial dynamic drafting module uses the mapping table to dynamically adjust the draft length, and the parallel draft verification module updates the mapping table immediately after each verification. Below, we will first explain the design of the confidence-acceptance mapping table, and then explain the modules in SPIDE. We include the full algorithm of SPIDE in Appendix A.

Confidence-Acceptance Mapping Table.

We maintain a confidence-acceptance mapping table during the decoding process, with the goal of effectively mapping the confidence of draft tokens to the acceptability of the tokens in real time, to guide SPIDE to set drafting lengths dynamically. We use the raw probability as the confidence of each draft token, which is calculated based on M_q predicting the maximum probability of the next token x_{i+1} given the previous context $x_{<i+1}$, and is represented as:

$$C_{i+1} = \max_{x_{i+1}} p(x_{i+1} \mid x_{<i+1}) \quad (2)$$

where $p(x_{i+1} \mid x_{<i+1})$ is the probability distribution of predicting the next token x_{i+1} given the previous context $x_{<i+1}$.

The acceptance rate α reflects the proportion of tokens generated by M_q that are accepted by M_p. We divide token confidence values into ten intervals and record both the portion of tokens and the acceptance rate of each confidence interval.

Figure 3 shows example token confidence-acceptance values by intervals, obtained with DeepSeek-1.3B and DeepSeek-33B being the draft model and the target model, respectively. We observe that the confidence and acceptance values have a strong positive correlation. Moreover, draft tokens with confidence in the range of $(0.9, 1.0]$

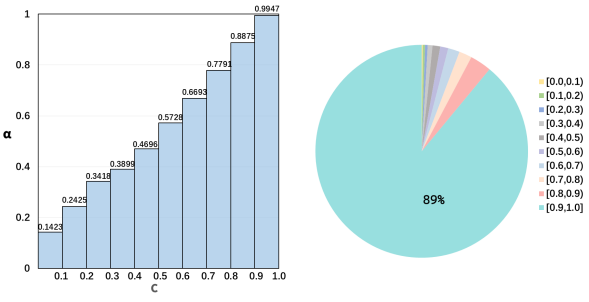


Figure 3: The relationship between the confidence intervals and acceptance rates of draft tokens (left), and the proportion of draft tokens in each confidence interval during the decoding process (right).

Statistics are updated in real time during the decoding process.

Confidence	Total Number (N_T)	Accepted Number (N_A)	Acceptance Rate (N_A/N_T)
[0, 0.1)	38	15	0.3947
[0.1, 0.2)	195	66	0.3385
⋮	⋮	⋮	⋮
[0.8, 0.9)	2365	2099	0.8875
[0.90, 0.91)	229	213	0.9301
[0.91, 0.92)	436	413	0.9472
⋮	⋮	⋮	⋮
[0.99, 1.0)	8270	8217	0.9936
[1.0]	55759	55729	0.9995

Figure 4: A snapshot of our confidence-acceptance mapping table maintained in real time by DeepSeek 1.3B & 33B during the decoding process.

are the vast majority, with a high acceptance rate close to 100%. Similar situations are observed in other model pairs, as reported in Appendix B.

Taking advantage of this positive correlations, it is intuitive to consider token confidence as the probability that the draft tokens will be accepted after verification. We take a step further and propose a more fine-grained strategy based on real-time statistics to align confidence and acceptance probability. As shown in Figure 4, we maintain a confidence-acceptance mapping table containing 20 entries, covering the confidence intervals of $[0, 0.1)$, $[0.1, 0.2)$, ... , $[0.8, 0.9)$, $[0.9, 0.91)$, $[0.91, 0.92)$, ... , $[0.99, 1.0)$ and 1.0. During the decoding process, the total number of draft tokens and the number of accepted tokens within each interval will be counted in real time, and the acceptance rate for each interval will be updated accordingly.

This strategy alleviates the deviation between the confidence of the draft tokens and the actual acceptance probability, so as to align the two in a more explainable and fine-grained manner. Compared with SpecDec++ (Huang et al., 2024) for training an acceptance prediction head or DISCO (Mamou et al., 2024) for training a simple classifier , our method does not require an additional training process, making it more dynamically adaptable to evolving output token confidence and acceptability.

It should be noted that, we compare the mapping tables formed by draft tokens generated at different time periods. The results shows that the differences between them are marginal. In addition, after the decoding process starts and a reasonably small number (e.g., 500) draft tokens are generated, the acceptance rate of each confidence interval can quickly stabilize. Thus, the impact of drifts in the relationship between the confidence interval and the acceptance rate, as well as cold start, can be considered negligible.

Serial Dynamic Drafting Module.

This module achieves better dynamic control of draft length by using the above mapping table. In serial dynamic drafting, we use M_q to generate draft tokens sequentially. After each token is drafted, we determine the corresponding acceptance rate from the current confidence-acceptance mapping table based on the confidence score of that token. This acceptance rate is then utilized as the probability of the draft token being accepted upon future verification. At the same time, a reliability value is updated, which represents the overall reliability of the current sequence of i draft tokens:

$$\Gamma_i = \prod_{k=1}^{i} \alpha_{b_k}, \tag{3}$$

where b_k represents the confidence interval to which the k-th token belongs, α_{b_k} represents the acceptance rate of the corresponding interval.

In this module, we decide the next action based on whether the overall reliability Γ_i of the current draft sequence is higher than a reliability threshold (a hyperparameter τ), formalized as the following function:

$$\text{DraftAction}(\Gamma_i, \tau) = \begin{cases} 1, & \Gamma_i > \tau, \\ 0, & else, \end{cases} \tag{4}$$

where 1 indicates the continuation of serial dynamic drafting, using M_q to generate the next draft token, and 0 indicates switching to parallel draft verification, activating M_p to verify the current draft sequence.

As shown in Figure 2, after inputting a prefix, the serial dynamic drafting module starts. M_q autoregressively drafts a sequence $[x_1, x_2, \ldots, x_7]$. After generating x_i $(0 < i \leq 7)$, we will check the acceptance rate α_{b_i} corresponding to x_i in the confidence-acceptance mapping table, and then update Γ_i (the initial value is set to 1.0) with α_{b_i}. After x_7 is generated, Γ_7 is less than τ. M_p is activated at this point for parallel draft verification.

The update mechanism of Γ_i is an online Bayesian belief update process:

$$P(H_0|x_{1:i}) \propto \left(\prod_{k=1}^{i} \alpha_{b_k} \right) \cdot \pi_0, \tag{5}$$

	Vicuna 68M & 13B			Vicuna 160M & 33B			Pythia 160M & 12B			DeepSeek 1.3B & 33B			TinyLlama 1.1B & CodeLlama 34B			Qwen3 1.7B & 32B		
Methods	Speed	Cost	MAT	Speed	Cost	MAT	Speed	Cost	MAT	Speed	Cost	MAT	Speed	Cost	MAT	Speed	Cost	MAT
AD	1.00×	–	–	1.00×	–	–	1.00×	–	–	1.00×	–	–	1.00×	–	–	1.00×	–	–
SD	1.76×	1.00×	1.91	1.97×	1.00×	2.29	1.56×	1.00×	3.30	2.79×	1.00×	8.25	2.33×	1.00×	5.65	1.60×	1.00×	3.65
SVIP	1.86×	1.02×	1.84	2.10×	0.94×	2.43	1.70×	0.90×	3.82	3.04×	0.92×	9.10	2.77×	**0.64×**	9.57	1.53×	1.07×	3.45
D&V	1.88×	1.03×	1.86	2.22×	0.97×	2.55	1.88×	0.96×	3.69	3.16×	**0.90×**	9.44	2.89×	0.84×	9.44	1.76×	**0.95×**	3.88
PEARL	1.99×	1.27×	1.93	2.42×	1.30×	2.86	2.65×	1.35×	6.44	4.41×	1.77×	23.26	3.65×	1.58×	14.68	2.48×	1.80×	8.47
SPIDE	**2.38×**	**0.95×**	**2.54**	**2.66×**	**0.91×**	**3.52**	**2.83×**	**0.84×**	**7.89**	**4.52×**	1.26×	**24.83**	**3.86×**	1.20×	**14.94**	**2.64×**	1.26×	**9.14**

Table 1: Results on the HumanEval dataset, a code generation task. We bold **the best results** for each model combination and annotate the verification cost of two parallel methods (i.e., our SPIDE and PEARL) in gray. MAT stands for mean accepted tokens. "–" indicates that this indicator cannot be evaluated since AD does not involve drafting and verification.

	Vicuna 68M & 13B			Vicuna 160M & 33B			Pythia 160M & 12B			DeepSeek 1.3B & 33B			TinyLlama 1.1B & CodeLlama 34B			Qwen3 1.7B & 32B		
Methods	Speed	Cost	MAT	Speed	Cost	MAT	Speed	Cost	MAT	Speed	Cost	MAT	Speed	Cost	MAT	Speed	Cost	MAT
AD	1.00×	–	–	1.00×	–	–	1.00×	–	–	1.00×	–	–	1.00×	–	–	1.00×	–	–
SD	1.64×	**1.00×**	2.08	2.27×	1.00×	4.26	1.35×	1.00×	2.79	2.30×	**1.00×**	6.59	2.52×	1.00×	7.04	1.34×	1.00×	3.09
SVIP	1.65×	1.05×	1.95	2.57×	**0.99×**	4.43	1.42×	0.87×	3.39	2.56×	1.03×	6.50	2.91×	**0.53×**	14.52	1.46×	**0.98×**	3.34
D&V	1.60×	1.09×	1.88	2.44×	1.07×	4.10	1.58×	0.91×	3.78	2.61×	1.09×	6.81	2.76×	0.86×	12.89	1.52×	1.03×	3.88
PEARL	2.01×	1.33×	2.13	3.22×	1.47×	5.41	2.62×	1.23×	5.90	3.77×	1.68×	14.05	4.45×	1.57×	42.57	2.22×	1.65×	5.11
SPIDE	**2.41×**	1.02×	**2.98**	**3.76×**	1.09×	**6.89**	**2.86×**	**0.75×**	**7.65**	**3.99×**	1.25×	**15.11**	**4.56×**	1.40×	**44.96**	**2.53×**	1.05×	**5.91**

Table 2: Results on the GSM8K dataset, a arithmetic reasoning task (same result presentation style as in Table 1).

where H_0 means "the current draft sequence is overall reasonable". We set the prior $P(H_0) = \pi_0$ (1.0 in the experiments), and continuously update the posterior through the draft token generated at each step. When Γ_i is lower than τ, we believe that the overall reliability of the current draft sequence is lower than our expectation, the risk of rejection is high, and it is time to switch to parallel draft verification.

Parallel Draft Verification Module.

The parallel draft verification module contains two workflows that run in parallel: verification of the previous draft sequence and continued drafting based on the previous draft sequence. It aims to simultaneously mitigate efficiency degradation caused by rollback and premature draft termination.

As shown in Figure 2, after switching to parallel draft verification, M_q continues to draft $[x_8, x_9, x_{10}, x_{11}]$ based on the assumption that $[x_1, x_2, \ldots, x_7]$ will eventually be fully accepted by M_p. Meanwhile, the previously idle M_p receives the prefix $\mathbf{x}$ and $[x_1, x_2, \ldots, x_7]$, and it performs a forward propagation. Using the output logits $[p_0, p_1, \ldots, p_7]$, we verify $[x_1, x_2, \ldots, x_7, x_8]$ in parallel. In the example, $[x_1, x_2, \ldots, x_7]$ is fully accepted by M_p. Therefore, we continue with another round of parallel draft verification over $[x_9, x_{10}, x_{11}, x_{12}]$ while M_q continues to draft $[x_{12}, x_{13}, x_{14}, x_{15}]$, this time x_{11} is rejected, so $[x_{11}, x_{12} \ldots, x_{15}]$ is discarded. After y_{11} is sampled for correction, we switch back to serial dynamic drafting. After each verification, the mapping table is updated accordingly.

Moreover, to minimize the wait time between M_p and M_q in parallel draft verification (i.e., when M_p finishes verifying the previous batch of drafts, M_q also finished drafting the new s draft tokens), we set:

$$s = \left\lfloor \frac{v_q}{v_p} \right\rfloor, \tag{6}$$

where v_q is the running speed of M_q, and v_p is the running speed of M_p.

4 Experiments

4.1 Experimental Setup

Models We use six pairs of draft and target models to evaluate the effectiveness of SPIDE, including Vicuna-68M & 13B, Vicuna-160M & 33B (Chiang et al., 2023), Pythia-160M & 12B (Biderman et al., 2023), TinyLlama-1.1B & CodeLlama-34B (Zhang et al., 2024a; Roziere et al., 2023), DeepSeek-1.3B & 33B (Guo et al., 2024) and

Methods	Bengali	German	English	Spanish	French	Japanese	Russian	Swahili	Tegulu	Thai	Chinese	Avg.	Cost	MAT
AD	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	-	-
SD	2.40×	1.89×	1.55×	1.65×	1.66×	1.51×	1.50×	2.17×	1.74×	1.77×	1.66×	1.77×	**1.00×**	1.52
SVIP	2.16×	2.14×	1.69×	1.56×	1.77×	1.57×	1.56×	2.49×	1.64×	1.79×	1.67×	1.82×	1.06×	1.66
D&V	2.26×	2.33×	1.65×	1.76×	1.67×	1.45×	1.87×	2.56×	1.67×	1.67×	1.56×	1.76×	1.04×	1.59
PEARL	3.06×	2.86×	1.75×	1.87×	2.14×	1.58×	2.43×	3.95×	1.80×	1.97×	1.88×	2.30×	1.34×	2.06
SPIDE	**3.94×**	**3.02×**	**2.40×**	**2.12×**	**2.41×**	**1.91×**	**2.92×**	**4.02×**	**2.37×**	**3.28×**	**2.23×**	**2.79×**	1.09×	**2.54**

Table 3: Results of Vicuna 68M & 13B on the MGSM dataset, a multilingual arithmetic reasoning task. We bold **the best results** of each model combination.

Methods	Writing	Roleplay	Reasoning	Math	Coding	Extraction	Stem	Humanities	Avg.	Cost	MAT
AD	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	-	-
SD	1.55×	1.41×	1.58×	1.70×	1.60×	1.32×	1.47×	1.53×	1.52×	1.00×	1.68
SVIP	1.57×	1.49×	1.71×	1.57×	1.50×	1.31×	1.44×	1.53×	1.51×	**0.98×**	1.61
D&V	1.68×	1.45×	1.76×	1.51×	1.45×	1.40×	1.48×	1.52×	1.65×	1.07×	1.87
PEARL	1.73×	1.72×	2.38×	2.38×	1.89×	1.55×	1.53×	1.56×	1.84×	1.29×	2.05
SPIDE	**2.06×**	**2.13×**	**2.58×**	**2.54×**	**2.20×**	**1.89×**	**1.98×**	**1.97×**	**2.17×**	1.09×	**2.58**

Table 4: Results of Vicuna 68M & 13B on the MT-bench dataset, a multi-round dialogue task. We bold **the best results** of each model combination.

Qwen3-1.7B & 32B (Yang et al., 2025). All models are run on a single-node server with four 3090 GPUs. We detail model configurations in Appendix C.1.

Datasets We conduct experiments on four representative text generation tasks to evaluate the effectiveness of our SPIDE, including HumanEval (code generation task) (Chen et al., 2021), GSM8K (multilingual arithmetic reasoning task) (Cobbe et al., 2021), MGSM (the multilingual translation of GSM8K) (Shi et al., 2022) and MT-bench (multi-round dialogue task) (Zheng et al., 2023). Since we do not observe any significant difference between sampling with temperature 1 and greedy decoding in previous speculative decoding experiments (Leviathan et al., 2023), and to ensure our experiments are fully reproducible, we perform sampling at temperature 0, i.e., using greedy decoding by default. We detail dataset configurations in Appendix C.2. All models and datasets used are publicly available under open-source licenses that permit research use and redistribution.

Baseline Methods We implement five training-free decoding methods as our baselines. (i) **Autoregressive Decoding (AD):** the standard paradigm for LLMs text generation. (ii) **Speculate decoding (SD):** the vanilla SD approach, which uses a draft model to draft tokens and then verifies them in parallel (Leviathan et al., 2023; Chen et al., 2023). (iii) **SVIP:** a dynamic drafting method, where the length of the draft sequence is adaptively determined based on the entropy of each draft token distribution (Zhang et al., 2024b). (iv) **D&V(Draft&Verify):** another dynamic drafting method, which stop drafting when the confidence of each draft token falls below a dynamically adjusted threshold. (v) **PEARL:** a parallel SD method, where the drafting phase and the verification phase are carried out in parallel (Liu et al., 2025).

Evaluation Metrics We use three metrics to evaluate the methods: (i) **Walltime speedup ratio (Speed):** the speedup of each method relative to autoregressive decoding; (ii) **Verification overhead ratio (Cost):** the ratio of the number of calls to the target model under the same task relative to that of the vanilla SD; (iii) **Mean accepted tokens (MAT):** the average number of tokens generated by the draft model that are successfully accepted by the target model during the decoding process. This metric directly affects the efficiency speedup of decoding. We provide additional evaluation details in Appendix C.3.

4.2 Main Results

As shown in Table 1 and Table 2, SPIDE outperforms vanilla SD, SVIP, D&V and PEARL in terms of speedup and MAT on all model pairs in the HumanEval and GSM8K datasets. Compared

Methods	HumanEval						GSM8K	
	Vicuna 68M & 13B	Vicuna 160M & 33B	Pythia 160M & 12B	DeepSeek 1.3B & 33B	TinyLlama 1.1B & CodeLlama 34B	Qwen3 1.7B & 32B	Pythia 160M & 12B	DeepSeek 1.3B & 33B
SPIDE	2.38×	2.66×	2.83×	4.52×	3.86×	2.64×	2.86×	3.99×
SPIDE *w/o dyna-draft*	2.15×	2.36×	2.48×	3.83×	3.16×	2.08×	2.45×	3.29×
SPIDE *w/o para-verify*	2.05×	2.27×	1.96×	3.41×	3.07×	1.94×	1.63×	2.92×
SD	1.76×	1.97×	1.56×	2.79×	2.33×	1.60×	1.35×	2.30×
SVIP	1.99×	2.10×	1.70×	3.04×	2.77×	1.53×	1.42×	2.56×
D&V	1.88×	2.22×	1.88×	3.16×	2.89×	1.76×	1.49×	2.47×

Table 5: Ablation results of SPIDE on HumanEval and GSM8K datasets. We annotate the comparison of the three serial frameworks in gray.

with autoregressive decoding, SPIDE is speeded up by **3.25×** on average and up to **4.56×**. Compared with vanilla SD, SPIDE only increases the cost by **8.2%** on average, but brings an additional **67.7%** speedup on average. Compared with strictly parallel SD framework, SPIDE achieves a faster speedup of up to **19.6%** while reducing the additional cost of parallelism by **83.7%** on average. In addition, as shown in Table 3 and Table 4, SPIDE also outperforms the baseline methods and achieves significant inference speedups on 12 multilingual arithmetic tasks and 8 multi-turn dialogue tasks. Experiment results demonstrate the outstanding effectiveness of SPIDE in inference acceleration and cost control.

In addition, (1) we provide complete experimental results on MGSM and MT-bench respectively in Appendix D.1 and D.2, respectively. (2) We investigate the sensitivity of SPIDE to τ. Generally speaking, larger τ leads to better speedup but more usage cost. On the contrary, smaller τ leads to lower usage costs but sub-optimal speedup. We find that setting τ between 0.5 and 0.9 is a good balance between speedup and usage cost. Detailed results and analysis are given in Appendix D.3. The τ and s used for the optimal SPIDE experimental results are provided in Appendix D.4. (3) The actual invocation of the target model on the HumanEval and GSM8K datasets is provided in Appendix D.7.

4.3 Ablation Study

To verify the effectiveness of SPIDE's serial dynamic drafting strategy and parallel draft verification module, we conduct an ablation study. SPIDE *w/o dyna-draft* does not dynamically adjust draft length but set it to the optimal γ of vanilla SD, which is provided in Appendix D.5. SPIDE *w/o para-verify* runs drafting and verification sequentially, i.e., it removes parallel verification from SPIDE.

We present the main results in Table 5. It is evident that the efficiency of both SPIDE *w/o dyna-draft* and SPIDE *w/o para-verify* has decreased obviously compared to the complete SPIDE. Intuitively, SPIDE *w/o dyna-draft* may cause more rollbacks due to static drafting, and SPIDE *w/o para-verify* is prone to wasting time for stopping drafting too early.

Among serial SD frameworks, the acceleration effect of SPIDE *w/o para-verify* is sill superior to that of vanilla SD, SVIP and D&V. This further shows the effectiveness of the mapping mechanism.

We also investigate the sensitivity of SPIDE *w/o para-verify* to τ. The results and analysis are provided in Appendix D.6.

5 Conclusion

We introduced a SD framework named SPIDE, which alternates between serial dynamic drafting and parallel draft verification adaptively. Experimental results demonstrate that SPIDE achieves strong acceleration with limited overhead on invoking the target model, which indicates that SPIDE is a more efficient and resource-friendly inference acceleration framework.

6 Acknowledgement

The research was supported by Natural Science Foundation of China (62272403).

7 Limitations

Diverse tasks expansion Our experiments mainly focus on text generation tasks. Although SPIDE can in principle be applied to other models or generation tasks, additional factors such as domain-specific tokenization or specialized text structure may impact overall inference speed.

Cost of Draft Models We primarily focus on the invocation of the target LLM as a critical factor in cost control within the speculative decoding framework. Although the draft model is typically a lightweight language model that incurs minimal invocation costs, it still contributes to certain usage expenses during the decoding process.

References

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, and 1 others. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

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

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Ziyi Chen, Xiaocong Yang, Jiacheng Lin, Chenkai Sun, Kevin Chang, and Jie Huang. 2024. Cascade speculative drafting for even faster llm inference. *Advances in Neural Information Processing Systems*, 37:86226–86242.

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

Krishna Teja Chitty-Venkata and Arun K Somani. 2022. Neural architecture search survey: A hardware perspective. *ACM Computing Surveys*, 55(4):1–36.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Cunxiao Du, Jing Jiang, Xu Yuanchen, Jiawei Wu, Sicheng Yu, Yongqi Li, Shenggui Li, Kai Xu, Liqiang Nie, Zhaopeng Tu, and 1 others. 2024. Glide with a cape: A low-hassle method to accelerate speculative decoding. *arXiv preprint arXiv:2402.02082*.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, and 1 others. 2024. Layerskip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, and 1 others. 2024. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

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

M Kiruthika, Vibhav Krashan Chaurasiya, Balaraman Sundarambal, N Kirubakaran, and 1 others. 2025. A comprehensive review of low-rank decomposition techniques for parameter efficient fine-tuning. In *2025 International Conference on Data Science, Agents & Artificial Intelligence (ICDSAAI)*, pages 1–6. IEEE.

Jiedong Lang, Zhehao Guo, and Shuyu Huang. 2024. A comprehensive study on quantization techniques for large language models. In *2024 4th International Conference on Artificial Intelligence, Robotics, and Communication (ICAIRC)*, pages 224–231. IEEE.

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

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

Tianyu Liu, Yun Li, Qitan Lv, Kai Liu, Jianchen Zhu, Winston Hu, and Xiao Sun. 2025. Pearl: Parallel speculative decoding with adaptive draft length. In *The Thirteenth International Conference on Learning Representations*.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2018. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*.

Jonathan Mamou, Oren Pereg, Daniel Korat, Moshe Berchansky, Nadav Timor, Moshe Wasserblat, and

Roy Schwartz. 2024. Dynamic speculation lookahead accelerates speculative decoding of large language models. *arXiv preprint arXiv:2405.04304*.

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. *arXiv preprint arXiv:2305.09781*, 1(2):4.

Giovanni Monea, Armand Joulin, and Edouard Grave. 2023. Pass: Parallel speculative sampling. *arXiv preprint arXiv:2311.13581*.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. 2024. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*.

Sihyeong Park, Sungryeol Jeon, Chaelyn Lee, Seokhun Jeon, Byung-Soo Kim, and Jemin Lee. 2025. A survey on inference engines for large language models: Perspectives on optimization and efficiency. *arXiv preprint arXiv:2505.01658*.

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

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, and 1 others. 2022. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*.

Benjamin Spector and Chris Re. 2023. Accelerating llm inference with staged speculative decoding. *arXiv preprint arXiv:2308.04623*.

Wenxiao Wang, Wei Chen, Yicong Luo, Yongliu Long, Zhengkai Lin, Liye Zhang, Binbin Lin, Deng Cai, and Xiaofei He. 2024. Model compression and efficient inference for large language models: A survey. *arXiv preprint arXiv:2402.09748*.

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Sen Yang, Shujian Huang, Xinyu Dai, and Jiajun Chen. 2024. Multi-candidate speculative decoding. *arXiv preprint arXiv:2401.06706*.

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

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024a. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*.

Ziyin Zhang, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Rui Wang, and Zhaopeng Tu. 2024b. Draft model knows when to stop: A self-verification length policy for speculative decoding. *arXiv preprint arXiv:2411.18462*.

Juntao Zhao, Wenhao Lu, Sheng Wang, Lingpeng Kong, and Chuan Wu. 2024a. Qspec: Speculative decoding with complementary quantization schemes. *arXiv preprint arXiv:2410.11305*.

Yao Zhao, Zhitian Xie, Chen Liang, Chenyi Zhuang, and Jinjie Gu. 2024b. Lookahead: An inference acceleration framework for large language model with lossless generation accuracy. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6344–6355.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

A Algorithm

Here, we give the whole algorithm of our SPIDE in detail in Algorithm 1.

B More Observations on Confidence and Acceptance

In the figure 5, we provide observations on the acceptance rate and proportion of draft tokens within each interval when another five pairs of models perform speculative decoding on the HumanEval dataset.

C Evaluation Details

C.1 Model Configurations

We summarize the model configurations in Table 6. In our experiments, all models are loaded with bfloat-16 precision. Our SPIDE does not introduce any additional training, but directly uses these models to evaluate our algorithm.

C.2 Dataset Configurations

In our experiments, we evaluate the effectiveness of our SPIDE on 4 text generation tasks, including code generation, arithmetic reasoning, multilingual inference, and multi-round dialogue. For the code generation task, we employ HumanEval (Chen et al., 2021), a famous code generation benchmark composed of 164 entries. For arithmetic reasoning and multilingual inference, we employ GSM8K and MGSM (Cobbe et al., 2021; Shi et al., 2022) as the evaluation benchmark. For GSM8K, we sample the first 100 entries for evaluation. For the other 10 categories in MGSM, we select 10 entries for each language. For multi-round dialogue, we employ MT-bench (Zheng et al., 2023) as the benchmark. The maximum generation lengths of these tasks are respectively set to 1024, 256, 256, and 256.

C.3 Detailed Experimental Settings

All our experiments, including ablation studies and case studies, were performed on a single-node system equipped with four NVIDIA GeForce RTX 3090 graphics cards. For inference, we use batch size 1, which is common in other speculative decoding works. For the baselines compared, including SD, SVIP, D&V and PEARL, we all reproduce their results on different text generation tasks using the optimal parameters described in their papers or found in our experiments. When evaluating these methods, the model configuration and GPU usage are exactly the same as our SPIDE.

D More Experimental Results

D.1 More Experimental Results on MGSM

We provide more experimental results on the MGSM dataset in Tables 7, 8, 9, 10, 11. SPIDE generally achieves the highest inference speed over all baseline methods on the MGSM dataset.

D.2 More Experimental Results on MT-bench

We provide more experimental results on the MT-bench dataset in Tables 12, 13, 14, 15, 16. SPIDE achieves the highest inference speed over all baseline methods on the MT-bench dataset.

D.3 Sensitive Analysis of the Threshold τ

We conduct some case studies to understand the impact of different threshold hyperparameter τ values on inference speed and LLM usage cost, as shown in the Figure 6.

Intuitively speaking, τ represents the degree of conservatism in SPIDE's estimation of the reliability of the current draft sequence during serial dynamic drafting. When the τ is relatively high, the serial dynamic drafting will be more conservative, and the system will enter parallel drafting verification earlier. This effectively avoids the reduction in decoding efficiency caused by rollback. However, entering parallel drafting verification too early will also result in more frequent calls to expensive target model, causing higher LLM usage costs. In contrast, when τ is relatively low, serial dynamic drafting will be more aggressive in drafting more prediction tokens, and parallel drafting verification will start later, which saves LLM usage cost, but may increase the rollback phenomenon so that cause decrease in the acceleration effect.

Based on our case study, we observe that when τ is between 0.9 and 0.5, a better balance can be achieved between speed and cost. By controlling the value of τ, SPIDE can easily trade off the LLM usage cost and the inference speed to suit different scenarios. Moreover, when the cost of SPIDE is on a par with or even lower than that of vanilla SD, its inference speed is still significantly better than that of vanilla SD. This once again confirms the effectiveness of SPIDE in both acceleration effect and cost control.

D.4 The Optimal Hyperparameter for SPIDE

In Table 18, we provid the values of τ and s that are used for all the best results of the SPIDE experiments.

D.5 The Optimal γ for Speculative Decoding

The speedup of speculative decoding is affected by its fixed drafting length γ. Therefore, all the results of speculative decoding in this paper are based on its best γ. We show in Table 17 some search results for γ for speculative decoding.

D.6 Sensitivity Analysis of τ for SPIDE *w/o para-verify*

As shown in Figure 7, the trend of the cost change of SPIDE *w/o para-verify* is about the same as SPIDE. When τ decreases, dynamic drafting becomes relatively more aggressive, verification times decrease, and the overall LLM usage cost becomes lower. For lightweight drafting models, lower τ motivates better speedups and is less costly; The speedup of the non-lightweight drafting model is not very sensitive to τ, but the lower τ can make

the usage cost much lower than the optimal vanilla SD, while the speedup is still better than the optimal speedup of vanilla SD.

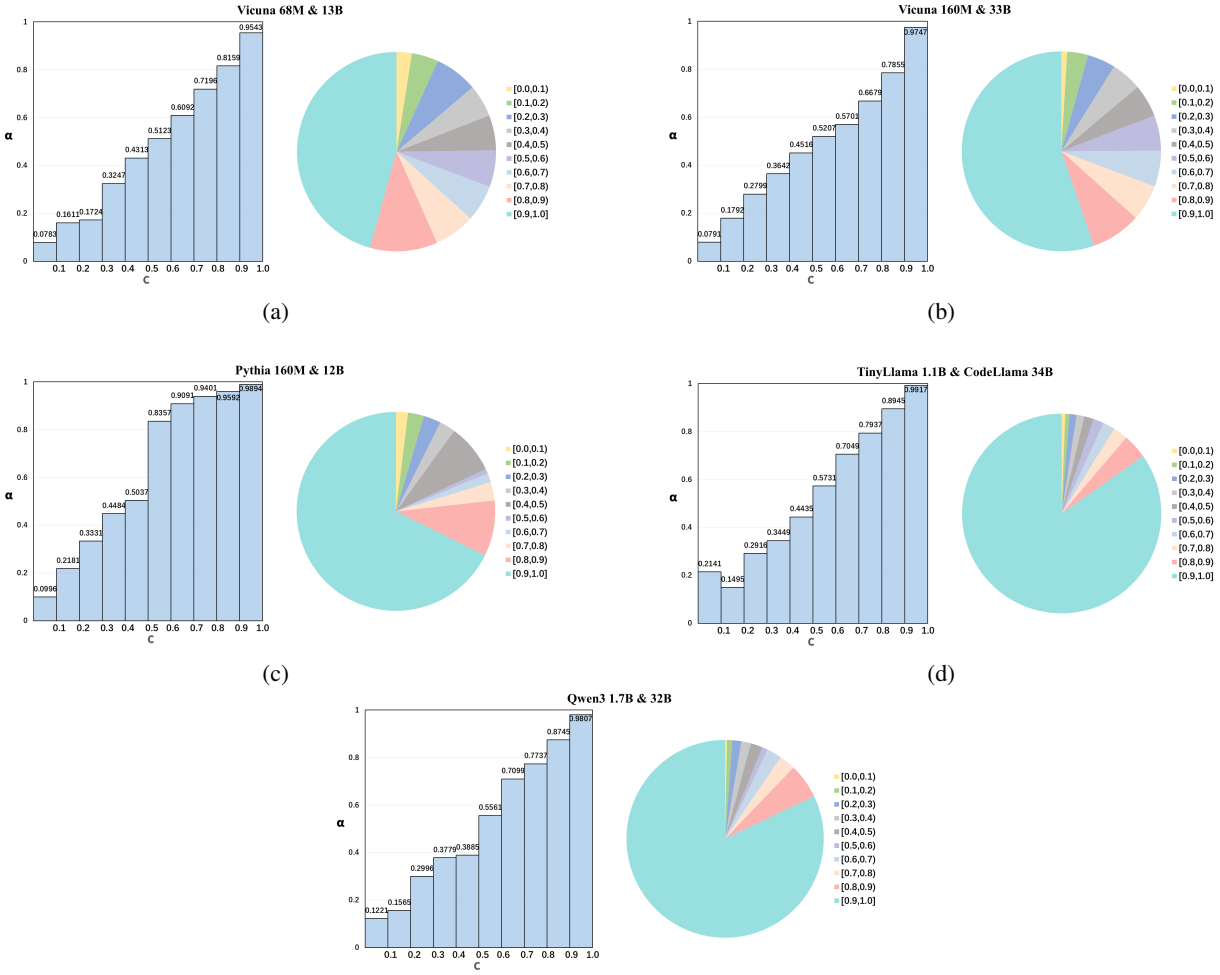
D.7 Calls to the target model

We provide the actual number of calls to the target model by our SPIDE and individual baseline methods on HumanEval and GSMK datasets in Table 19 and 20.

Algorithm 1 SPIDE

Require: the draft model M_q, the target model M_p, the input prefix x, the max generate tokens L, the window size s, the confidence threshold τ, the confidence-acceptance mapping table T

# Serial dynamic drafting is used first.
while $len(x) < L$ **do**
  if mode="serial" **then**
    # Serial dynamic drafting
    $\Gamma_0 = 1$
    for $k = 1$ to $L - len(x)$ **do**
      $q_k \leftarrow M_q(x + [x_1, ..., x_{k-1}])$
      $x_k \sim q_k$
      $b \leftarrow$ the confidence interval of x_k
      $\alpha \leftarrow T[b]$ # Check the mapping table T
      $\Gamma_k = a\Gamma_{k-1}$
      if $\Gamma_k \leq \tau$ **then**
        mode$\leftarrow$"parallel"
        break # Stop drafting
      end if
    end for
  else
    # Parallel drafting verification
    $x, [x_1, x_2, ..., x_k] \leftarrow x$ # Split the prefix to get the last k draft tokens
    for $i = k + 1$ to $k + s$ **do**
      # Run the draft model in parallel to continue drafting.
      $q_i \leftarrow M_q(x + [x_1, ..., x_{i-1}])$
      $x_i \sim q_i$
    end for
    # Run the target model in parallel to verify the tokens.
    $p_1, p_2, ..., p_k, p_{k+1} \leftarrow M_p(x), M_p(x + [x_1]), M_p(x + [x_1, x_2]), ..., M_p(x + [x_1, ..., x_k])$
    retrival $q_1, q_2, ..., q_k$ from the cache
    $r_1 \sim U(0, 1), ..., r_{k+1} \sim U(0, 1)$
    $n \leftarrow min(\{i - 1 \mid 1 \leq i \leq k + 1, r_i > \frac{p_i[x_i]}{q_i[x_i]}\} \cup \{k\})$
    update the confidence-acceptance mapping table T
    if $n = k + 1$ **then**
      # Accept all draft tokens
      $x \leftarrow x + [x_1, ..., x_{k+s}]$
      $k = s$
      mode $\leftarrow$ "parallel"
    else
      # Reject a draft token
      $y \sim norm(max(0, p_{n+1} - q_{n+1}))$
      $x \leftarrow [x_1, ..., x_n, y]$
      mode $\leftarrow$ "serial"
    end if
  end if
end while

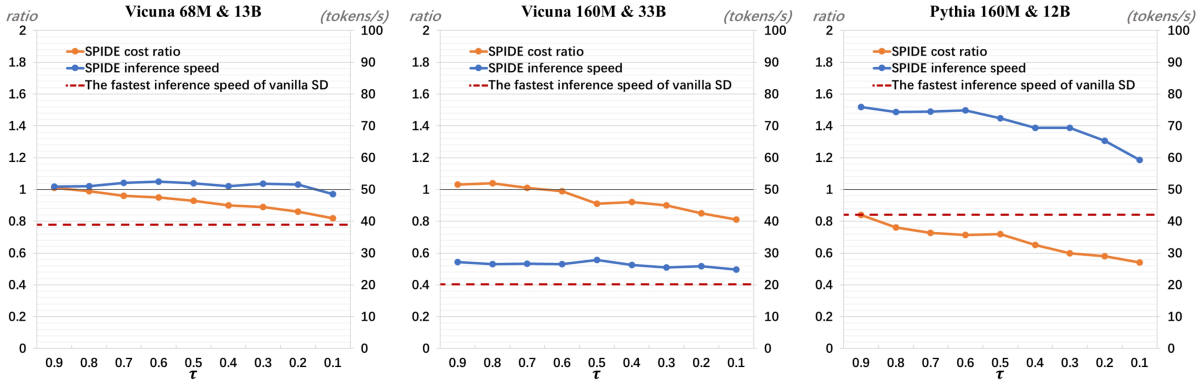


(a) (b) (c) (d) (e)

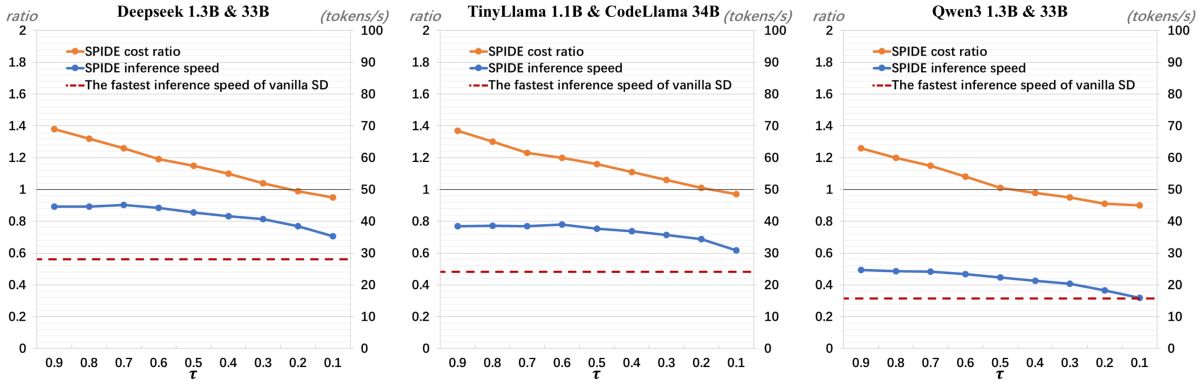
Figure 5: Additional observations about confidence.

Models	Layers	dim	FFN dim	Vocab size
Vicuna-68M	2	768	3072	32000
Vicuna-160M	12	768	3072	32000
Vicuna-13B	40	5120	13824	32000
Vicuna-33B	60	6656	17920	32000
Pythia-160M	12	768	3072	50304
Pythia-12B	36	5120	20480	50688
Deepseek-1.3B	24	2048	5504	32256
Deepseek-33B	62	7168	19200	32256
TinyLlama-1.1B	22	2048	5632	32000
CodeLlama-34B	48	8192	22016	32000
Qwen3-1.7B	28	2048	5504	151936
Qwen3-32B	64	5120	13696	151936

Table 6: Detailed model configurations.



(a)



(b)

Figure 6: Sensitivity analysis of τ for six pairs of models on the HumanEval dataset.The blue line represents the trend of SPIDE's decoding speed varying with τ. The yellow line represents the trend of SPIDE's cost ratio changing with τ. The dashed red line represents the decoding speed when vanilla SD adopts the best γ, and the solid black line in the middle represents the cost parity with the best vanilla SD.

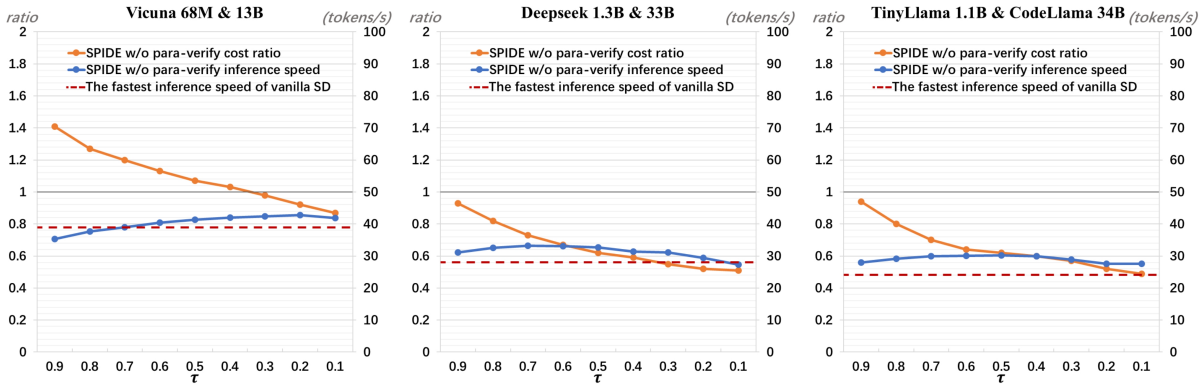


Figure 7: Sensitivity analysis of τ for SPIDE *w/o para-verify* of three pairs of models on the HumanEval dataset.The blue line represents the trend of SPIDE *w/o para-verify*'s decoding speed varying with τ. The yellow line represents the trend of SPIDE *w/o para-verify*'s cost ratio changing with τ. The dashed red line represents the decoding speed when vanilla SD adopts the best γ, and the solid black line in the middle represents the cost parity with the best vanilla SD.

Methods	Bengali	German	English	Spanish	French	Japanese	Russian	Swahili	Tegulu	Thai	Chinese	Avg.	Cost	MAT
AD	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	-	-
SD	1.68×	1.54×	1.55×	1.54×	1.55×	1.52×	1.51×	1.51×	1.53×	1.53×	1.53×	1.54×	1.00×	1.95
SVIP	1.68×	1.70×	1.68×	1.66×	1.70×	1.69×	1.66×	1.90×	1.63×	1.63×	1.71×	1.69×	1.09×	2.09
D&V	1.71×	1.66×	1.60×	1.65×	1.69×	1.61×	1.69×	1.87×	1.60×	1.57×	1.45×	1.74×	0.99×	2.28
PEARL	2.29×	1.81×	1.52×	1.61×	1.65×	1.70×	1.68×	2.47×	1.86×	1.94×	1.81×	1.84×	1.35×	2.67
SPIDE	**2.54×**	**2.13×**	**1.99×**	**1.86×**	**2.01×**	**1.97×**	**1.84×**	**2.51×**	**2.35×**	**2.28×**	**2.11×**	**2.13×**	**0.96×**	**2.94**

Table 7: Results of Vicuna 160M & 33B on the MGSM dataset, a multilingual arithmetic reasoning task. We bold **the best results** of each model combination.

Methods	Bengali	German	English	Spanish	French	Japanese	Russian	Swahili	Tegulu	Thai	Chinese	Avg.	Cost	MAT
AD	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	-	-
SD	1.53×	1.48×	1.42×	1.45×	1.45×	1.47×	1.42×	1.58×	1.46×	1.43×	1.44×	1.46×	1.00×	2.86
SVIP	1.48×	1.56×	1.54×	1.51×	1.57×	1.60×	1.55×	1.69×	1.50×	1.50×	1.51×	1.56×	0.96×	2.98
D&V	1.35×	1.87×	1.66×	1.67×	1.53×	1.68×	1.46×	1.63×	1.59×	1.52×	1.49×	1.59×	0.94×	3.22
PEARL	2.30×	1.98×	1.60×	1.70×	1.79×	1.80×	1.82×	2.34×	2.00×	2.19×	1.95×	1.95×	1.29×	4.69
SPIDE	**2.47×**	**2.09×**	**1.89×**	**1.80×**	**2.01×**	**1.96×**	**1.92×**	**2.35×**	**2.22×**	**2.25×**	**2.06×**	**2.08×**	**0.78×**	**6.47**

Table 8: Results of Pythia 160M & 12B on the MGSM dataset, a multilingual arithmetic reasoning task. We bold **the best results** of each model combination.

Methods	Bengali	German	English	Spanish	French	Japanese	Russian	Swahili	Tegulu	Thai	Chinese	Avg.	Cost	MAT
AD	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	-	-
SD	1.67×	1.54×	1.53×	1.52×	1.54×	1.56×	1.51×	1.71×	1.60×	1.53×	1.60×	1.59×	1.00×	5.79
SVIP	1.67×	1.66×	1.64×	1.63×	1.69×	1.67×	1.61×	1.86×	1.65×	1.62×	1.69×	1.66×	**0.94×**	6.22
D&V	1.75×	1.78×	1.60×	1.61×	1.48×	1.69×	1.56×	1.98×	1.71×	1.69×	1.76×	1.59×	0.99×	5.45
PEARL	2.34×	1.98×	1.71×	1.81×	1.89×	1.93×	1.92×	2.40×	2.20×	2.25×	2.02×	2.04×	1.64×	11.32
SPIDE	**2.60×**	**2.15×**	**2.03×**	**1.94×**	**2.19×**	**2.11×**	**1.95×**	**2.49×**	**2.46×**	**2.40×**	**2.18×**	**2.21×**	1.23×	**12.45**

Table 9: Results of Deepseek 1.3B & 33B on the MGSM dataset, a multilingual arithmetic reasoning task. We bold **the best results** of each model combination.

Methods	Bengali	German	English	Spanish	French	Japanese	Russian	Swahili	Tegulu	Thai	Chinese	Avg.	Cost	MAT
AD	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	-	-
SD	1.74×	1.63×	1.58×	1.59×	1.65×	1.66×	1.65×	1.81×	1.71×	1.60×	1.68×	1.68×	1.00×	6.78
SVIP	1.72×	1.74×	1.72×	1.70×	1.72×	1.79×	1.73×	1.96×	1.74×	1.71×	1.77×	1.77×	**0.67×**	7.38
D&V	1.89×	1.87×	1.77×	1.74×	1.87×	1.93×	1.67×	2.09×	1.98×	2.03×	1.89×	1.89×	1.00×	7.98
PEARL	2.37×	2.10×	1.86×	1.93×	2.05×	2.05×	2.07×	2.52×	2.34×	2.33×	2.09×	2.17×	1.48×	10.87
SPIDE	**2.67×**	**2.28×**	**2.19×**	**2.07×**	**2.30×**	**2.25×**	**2.12×**	**2.64×**	**2.61×**	**2.51×**	**2.32×**	**2.37×**	1.16×	**12.14**

Table 10: Results of Tinyllama 1.1B & Codellama 34B on the MGSM dataset, a multilingual arithmetic reasoning task. We bold **the best results** of each model combination.

Methods	Bengali	German	English	Spanish	French	Japanese	Russian	Swahili	Tegulu	Thai	Chinese	Avg.	Cost	MAT
AD	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	-	-
SD	1.46×	1.33×	1.32×	1.31×	1.33×	1.35×	1.30×	1.50×	1.39×	1.32×	1.39×	1.38×	1.00×	2.47
SVIP	1.45×	1.44×	1.42×	1.41×	1.47×	1.45×	1.39×	1.64×	1.43×	1.40×	1.47×	1.44×	**0.92×**	2.97
D&V	1.52×	1.55×	1.37×	1.38×	1.25×	1.46×	1.33×	1.75×	1.48×	1.46×	1.53×	1.36×	1.09×	1.99
PEARL	2.09×	1.73×	1.46×	1.56×	1.64×	1.68×	1.67×	2.15×	1.95×	2.00×	1.77×	1.79×	1.77×	5.11
SPIDE	**2.40×**	**1.95×**	**1.83×**	**1.74×**	**1.99×**	**1.91×**	**1.75×**	**2.29×**	**2.26×**	**2.20×**	**1.98×**	**2.01×**	1.21×	**6.45**

Table 11: Results of Qwen3 1.7B & 32B on the MGSM dataset, a multilingual arithmetic reasoning task. We bold **the best results** of each model combination.

Methods	Writing	Roleplay	Reasoning	Math	Coding	Extraction	Stem	Humanities	Avg.	Cost	MAT
AD	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	-	-
SD	1.26×	1.32×	1.42×	1.51×	1.42×	1.31×	1.37×	1.40×	1.38×	1.00×	2.77
SVIP	1.52×	1.53×	1.63×	1.63×	1.51×	1.45×	1.58×	1.58×	1.54×	**0.89×**	3.21
D&V	1.59×	1.47×	1.69×	1.71×	1.56×	1.49×	1.48×	1.48×	1.51×	0.93×	2.95
PEARL	1.90×	1.47×	2.04×	**2.09×**	1.77×	1.88×	1.57×	1.54×	1.77×	1.46×	4.70
SPIDE	**1.99×**	**1.52×**	**2.11×**	2.06×	**1.95×**	**2.09×**	**1.84×**	**1.65×**	**1.90×**	1.05×	**5.47**

Table 12: Results of Vicuna 160M & 33B on the MT-bench dataset, a multi-round dialogue task. We bold **the best results** of each model combination.

Methods	Writing	Roleplay	Reasoning	Math	Coding	Extraction	Stem	Humanities	Avg.	Cost	MAT
AD	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	-	-
SD	1.20×	1.25×	1.31×	1.39×	1.32×	1.31×	1.26×	1.32×	1.29×	1.00×	2.14
SVIP	1.44×	1.43×	1.48×	1.50×	1.40×	1.37×	1.44×	1.47×	1.44×	1.09×	3.76
D&V	1.48×	1.56×	1.59×	1.78×	1.67×	1.68×	1.64×	1.57×	1.53×	0.97×	4.34
PEARL	2.23×	1.90×	2.26×	2.18×	2.10×	2.15×	1.86×	1.80×	2.09×	1.29×	8.29
SPIDE	**2.41×**	**2.18×**	**2.31×**	**2.20×**	**2.22×**	**2.30×**	**1.88×**	**1.82×**	**2.20×**	**0.78×**	**10.02**

Table 13: Results of Pythia 160M & 12B on the MT-bench dataset, a multi-round dialogue task. We bold **the best results** of each model combination.

Methods	Writing	Roleplay	Reasoning	Math	Coding	Extraction	Stem	Humanities	Avg.	Cost	MAT
AD	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	-	-
SD	1.37×	1.42×	1.52×	1.60×	1.55×	1.40×	1.44×	1.50×	1.47×	**1.00×**	4.92
SVIP	1.63×	1.62×	1.71×	1.72×	1.62×	1.51×	1.64×	1.63×	1.63×	1.11×	5.13
D&V	1.78×	1.76×	1.69×	1.79×	1.67×	1.47×	1.68×	1.58×	1.68×	1.02×	4.87
PEARL	2.38×	2.09×	2.44×	**2.59×**	**2.36×**	2.33×	1.98×	1.96×	2.25×	1.66×	7.89
SPIDE	**2.68×**	**2.46×**	**2.59×**	2.47×	**2.36×**	**2.37×**	**2.08×**	**2.03×**	**2.39×**	1.19×	**9.01**

Table 14: Results of Deepseek 1.3B & 33B on the MT-bench dataset, a multi-round dialogue task. We bold **the best results** of each model combination.

Methods	Writing	Roleplay	Reasoning	Math	Coding	Extraction	Stem	Humanities	Avg.	Cost	MAT
AD	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	-	-
SD	1.39×	1.43×	1.52×	1.61×	1.52×	1.38×	1.45×	1.51×	1.47×	1.00×	3.45
SVIP	1.65×	1.62×	1.73×	1.74×	1.61×	1.52×	1.66×	1.67×	1.65×	**0.69×**	5.40
D&V	1.78×	1.95×	1.84×	1.94×	1.79×	1.60×	1.71×	1.88×	1.90×	0.87×	7.21
PEARL	2.40×	2.15×	2.46×	2.49×	2.27×	2.29×	2.08×	2.06×	2.25×	1.61×	9.84
SPIDE	**2.66×**	**2.46×**	**2.58×**	**2.61×**	**2.31×**	**2.33×**	**2.12×**	**2.11×**	**2.43×**	1.29×	**11.22**

Table 15: Results of Tinyllama 1.1B & Codellama 34B on the MT-bench dataset, a multi-round dialogue task. We bold **the best results** of each model combination.

Methods	Writing	Roleplay	Reasoning	Math	Coding	Extraction	Stem	Humanities	Avg.	Cost	MAT
AD	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	-	-
SD	1.14×	1.19×	1.29×	1.37×	1.32×	1.17×	1.21×	1.27×	1.24×	1.00×	2.56
SVIP	1.40×	1.39×	1.48×	1.49×	1.39×	1.28×	1.41×	1.40×	1.40×	1.08×	3.46
D&V	1.53×	1.51×	1.44×	1.54×	1.42×	1.22×	1.43×	1.33×	1.43×	**0.97×**	3.71
PEARL	2.13×	1.84×	2.19×	**2.34×**	2.11×	2.08×	1.73×	1.71×	2.00×	1.76×	8.56
SPIDE	**2.47×**	**2.25×**	**2.38×**	2.26×	**2.15×**	**2.16×**	**1.87×**	**1.82×**	**2.18×**	1.11×	**10.18**

Table 16: Results of Qwen 1.7B & 32B on the MT-bench dataset, a multi-round dialogue task. We bold **the best results** of each model combination.

Vicuna 68M & 13B	Vicuna 160M & 33B	Pythia 160M & 12B	DeepSeek 1.3B & 33B	TinyLlama 1.1B & CodeLlama 34B	Qwen3 1.7B & 32B
39.03 (γ=4)	18.67 (γ=3)	40.09 (γ=3)	27.66 (γ=9)	23.70 (γ=6)	14.01 (γ=3)
39.18 (γ=5)	19.18 (γ=4)	42.10 (γ=4)	27.84 (γ=10)	23.96 (γ=7)	14.76 (γ=4)
39.30 (γ=6)	**20.14 (γ=5)**	**42.34 (γ=5)**	**28.14 (γ=11)**	**24.24 (γ=8)**	**15.91 (γ=5)**
38.68 (γ=7)	20.00 (γ=6)	41.93 (γ=6)	27.85 (γ=12)	23.95 (γ=9)	15.03(γ=6)
37.45 (γ=8)	18.88 (γ=7)	41.66 (γ=7)	27.82 (γ=13)	22.45 (γ=10)	14.88(γ=7)

Table 17: Optimal γ values of speculative decoding for each model pair. (Unit: tokens / second)

		Vicuna 68M & 13B	Vicuna 160M & 33B	Pythia 160M & 12B	DeepSeek 1.3B & 33B	TinyLlama 1.1B & CodeLlama 34B	Qwen3 1.7B & 32B
Optimal τ	HumanEval	0.6	0.5	0.9	0.7	0.6	0.9
	GSM8K	0.7	0.8	0.7	0.7	0.7	0.8
	MGSM	0.6	0.7	0.9	0.8	0.7	0.9
	MT-bench	0.7	0.7	0.8	0.8	0.7	0.9
Optimal s		20	11	6	7	6	4

Table 18: The optimal values of τ and s used to produce the best experimental results on the four datasets.

Methods	Vicuna 68M & 13B	Vicuna 160M & 33B	Pythia 160M & 12B	DeepSeek 1.3B & 33B	TinyLlama 1.1B & CodeLlama 34B	Qwen3 1.7B & 32B
SD	57633	51081	52708	18251	25340	18154
SVIP	60517	50570	45856	18799	13430	19062
D&V	62820	54657	47964	19894	21792	17246
PEARL	76652	75090	64830	30662	39784	32677
SPIDE	58786	55678	39531	22814	35476	22874

Table 19: The number of times each method was called on each model pair on the HumanEval dataset.

Methods	Vicuna 68M & 13B	Vicuna 160M & 33B	Pythia 160M & 12B	DeepSeek 1.3B & 33B	TinyLlama 1.1B & CodeLlama 34B	Qwen3 1.7B & 32B
SD	33286	19515	27062	13560	12736	14669
SVIP	34950	19320	23544	13967	6750	14376
D&V	36282	20881	24626	14780	10953	15109
PEARL	44270	28687	33286	22781	19996	24204
SPIDE	33952	21271	20297	16950	17830	15402

Table 20: The number of times each method was called on each model pair on the GSM8K dataset.