Calibrated Speculative Decoding: Frequency-Guided Candidate Selection for Efficient Inference

Xuwen Zhou[1], **Fangxin Liu**[1*], **Chao Wang**[2], **Xiao Zheng**[2],
Hao Zheng[2], **Min He**[2], **Li Jiang**[1*], **Haibing Guan**[1]
[1]Shanghai Jiao Tong University / [2]Alibaba Cloud Computing
{xvenzhou, liufangxin, ljiang_cs , hbguan}@sjtu.edu.cn
[*]Corresponding authors

Abstract

Speculative decoding accelerates autoregressive generation by letting draft tokens bypass full verification, but conventional frameworks suffer from frequent false rejections, particularly when draft models produce semantically correct but lexically divergent outputs. In this paper, we present **Calibrated Speculative Decoding (CSD)**, a training-free framework that recovers valid tokens discarded by standard verification. Guided by the principle of *"Frequency-Guided Candidate Selection, Probability-Guarded Acceptance,"* CSD incorporates two lightweight modules: **Online Correction Memory**, which aggregates historical rejections to propose recurring divergence patterns as rescue candidates, and **Semantic Consistency Gating**, which verifies candidate admissibility using probability ratios instead of exact token matching. Our evaluation across diverse large language models demonstrates that CSD outperforms existing methods, achieving a peak throughput speedup of 2.33×. CSD preserves model accuracy across all tasks while boosting performance on complex reasoning datasets. These results establish CSD as a highly effective, lightweight solution for LLM deployments.

1 Introduction

Large Language Models (LLMs) have demonstrated unprecedented capabilities across a wide spectrum of tasks (Achiam et al., 2023; Guo et al., 2025). However, their deployment is severely constrained by the memory wall (Gholami et al., 2024). Specifically, the autoregressive nature of decoding renders inference predominantly memory-bandwidth bound, particularly in latency-sensitive, small-batch settings, where the arithmetic intensity is low (Chen et al., 2025; Pope et al., 2023; Liu et al., 2024a, 2021).

To alleviate this bottleneck, **Speculative Decoding (SD)** (Leviathan et al., 2023; Chen et al., 2023) has emerged as a dominant acceleration paradigm. SD employs a smaller draft model to propose candidate tokens that are verified in parallel by the target model, thereby amortizing memory access costs across multiple tokens and enabling lossless acceleration.

Recent advances in Small Language Models (SLMs) alter the role of draft models in SD. Driven by advances in model distillation and architectural optimization, modern SLMs are no longer mere approximators but capable reasoners (Bachmann et al., 2025). For instance, Llama-3-8B achieves 84.5% accuracy on the GSM8K math benchmark, significantly outperforming its much larger predecessor, Llama-2-70B (56.8%) (Dubey et al., 2024; Touvron et al., 2023). This paradigm shift presents a new opportunity: draft models are increasingly generating semantically correct responses that may differ lexically from the target model's preference.

However, existing SD frameworks fail to capitalize on this intelligence. Standard verification mechanisms, such as strict token matching or rejection sampling (Leviathan et al., 2023), operate on a rigid, distribution-alignment basis. This creates a fundamental mismatch: while the draft model demonstrates strong reasoning capabilities, the verification logic remains structurally intolerant. As illustrated in Figure 1(a), this rigidity leads to "False Rejections," where valid draft tokens (e.g., synonyms like `'x'` vs `'*'`) are discarded merely due to trivial lexical or stylistic discrepancies. This forces the target model to re-generate semantically equivalent tokens, thereby negating the efficiency gains, particularly in open-ended generation tasks.

To bridge the gap between rigid verification and flexible semantics, we introduce **Calibrated Speculative Decoding (CSD)**, a training-free framework designed to recover valid tokens from false rejections. Guided by the philoso-

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 29677–29688

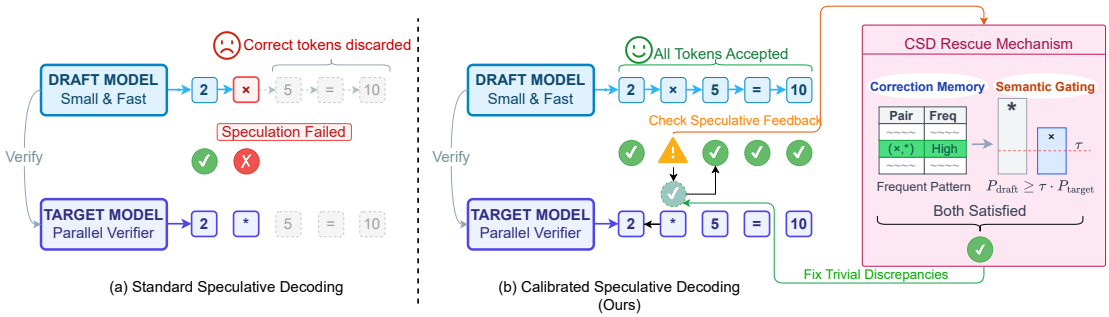


Figure 1: Overview of Calibrated Speculative Decoding (CSD). **(a)** Standard Speculative Decoding suffers from "False Rejections," where trivial lexical mismatches (e.g., 'x' vs '*') cause the discard of subsequent valid tokens (e.g., 5=10). **(b)** CSD mitigates such rigid rejections by introducing a Rescue Mechanism. It employs Online Correction Memory (OCM) to propose potential candidates based on historical priors, and uses Semantic Consistency Gating (SCG) to verify their admissibility via probability ratios (Score $\geq \tau$), effectively recovering wasted computation.

phy of *"Frequency-Guided Candidate Selection, Probability-Guarded Acceptance,"* CSD incorporates two lightweight modules (Figure 1(b)). First, Online Correction Memory continuously aggregates historical rejections to propose "rescue candidates" for recurring divergence patterns. Second, Semantic Consistency Gating evaluates the admissibility of these candidates through probability ratios rather than exact matching.

Our contributions are as follows: (1) We identify the Semantics-Alignment Mismatch in modern speculative decoding, revealing that rigid verification hinders the potential of draft models; (2) We propose Calibrated Speculative Decoding (CSD), a training-free framework that incorporates Online Correction Memory and Semantic Consistency Gating to enable adaptive, pattern-aware verification; (3) CSD delivers up to 2.33× speedup while maintaining generation quality across standard tasks. Importantly, it uniquely enhances complex reasoning accuracy by 2.5 points on HumanEval and 2.0 points on MATH500 (Hendrycks et al., 2021; Chen, 2021), proving that CSD safely recovers valid draft tokens to improve inference efficiency.

2 RELATED WORK

Speculative Decoding (SD) breaks the serial dependency of autoregressive generation by introducing a drafting step, establishing itself as a dominant paradigm for accelerating LLM inference. The classic approach typically employs an independent lightweight model from the same family as the target model to generate draft tokens (Chen et al., 2023; Leviathan et al., 2023). To circumvent the VRAM overhead and system complexity associated with independent models, some works utilize layer skipping or early-exiting strategies to construct draft models (Elhoushi et al., 2024; Zhang et al., 2024; Xia et al., 2024; Liu et al., 2024b), reusing a subset of target weights for efficient generation. Meanwhile, approaches like Eagle (Li et al., 2024) and Medusa (Cai et al., 2024) append lightweight decoding heads to the target model to predict future tokens in parallel. Furthermore, methods such as Lookahead Decoding (Fu et al., 2024) and Jacobi decoding (Santilli et al., 2023) eliminate the draft model entirely, achieving parallelism via parallel iterative decoding.

To maximize the speedup of SD, prior works have focused on improving the acceptance rate. On one hand, Tree-based methods (Miao et al., 2024; Chen et al., 2024) extend single-chain generation to token trees built from top-k predictions and verify them in parallel with tree-attention, enlarging the covered solution space. On the other hand, efforts improve draft quality through alignment strategies, such as distilling the draft from the target model (Zhou et al., 2023b) or sharing KV caches across models (Du et al., 2024). Recently, approaches such as Judge Decoding (Bachmann et al., 2025) and R2R (Fu et al., 2025) explore learned verification by training auxiliary models to assess token validity.

However, these advancements often introduce substantial overheads, including the high computational cost of tree-based verification and the training expense of learned methods, while remaining constrained by rigid exact matching. Such rigidity leads to the false rejection of semantically equivalent tokens. In contrast, we propose a lightweight, training-free framework that calibrates the verifi-

cation logic itself. By leveraging historical divergence patterns and semantic gating, our approach recovers valid tokens without architectural modifications. Notably, our framework is generically applicable to standard rejection sampling-based methods, serving as a seamless enhancement to their verification logic.

3 Preliminaries and Motivation

3.1 Speculative Decoding (SD)

Speculative Decoding accelerates the inference of a target model $\mathcal{M}_p$ by leveraging a computationally efficient draft model $\mathcal{M}_q$. The process follows a predict-then-verify paradigm. Given an input context x, the draft model first generates a sequence of γ candidate tokens $\tilde{x}_{1:\gamma}$ autoregressively. Specifically, at each step i, the token $\tilde{x}_i$ is sampled via:

$$\tilde{x}_i \sim q(\cdot \mid x, \tilde{x}_{<i}), \quad i \in \{1, \ldots, \gamma\} \tag{1}$$

where $q(\cdot)$ denotes the probability distribution output by $\mathcal{M}_q$.

Subsequently, $\mathcal{M}_p$ evaluates the drafted sequence in a single parallel forward pass to compute the target probabilities $p(\cdot \mid x, \tilde{x}_{<i})$ for all positions. To ensure the generated sequence follows the target distribution, SD adopts a rejection sampling scheme. A candidate token $\tilde{x}_i$ is accepted with probability α_i, defined as:

$$\alpha_i = \min\left(1, \frac{p(\tilde{x}_i \mid x, \tilde{x}_{<i})}{q(\tilde{x}_i \mid x, \tilde{x}_{<i})}\right) \tag{2}$$

If a token $\tilde{x}_i$ is rejected, the process terminates at step i, and a correction token is resampled from the normalized residual distribution $p' = \text{norm}(\max(0, p - q))$. Conversely, if all γ tokens are accepted, an additional token is sampled directly from $\mathcal{M}_p$. Crucially, this mechanism guarantees that the final output distribution is mathematically identical to that of the target model $\mathcal{M}_p$.

3.2 Why Do We Reject? Analyzing Draft Rejection Patterns

Rejection sampling guarantees correctness under exact token-level verification (Eq. 2), but this strict matching rule does not account for the lexical diversity of natural language. Recent studies (Bachmann et al., 2025; Fu et al., 2025) show that draft and target models often differ by semantically equivalent lexical choices, referred to as "neutral differences." Existing approaches typically relax rigid verification by introducing learned routers or judgment heads, which increases system complexity and inference overhead. In this work, we ask whether a subset of these valid but rejected draft tokens can be recovered by refining the verification logic in a training-free manner.

To characterize the prevalence and structure of such false rejections, we conduct a large-scale empirical analysis under standard speculative decoding. We use `LLaMA-3.2-1B-Instruct` as the drafter and `LLaMA-3-70B-Instruct` as the target, and collect all rejected token pairs during standard sampling ($T = 1$). Experiments are performed on a combined dataset of 28,000 samples from CNN/DailyMail (Hermann et al., 2015) and MATH (Hendrycks et al., 2021). Figure 2 summarizes the resulting statistics. We highlight two observations that inform our design.

Observation 1: Long-tailed distribution of rejection patterns. As shown in Figure 2(a), rejected draft–target token pairs exhibit a highly skewed distribution. The top 20% most frequent patterns account for approximately 69.0% of all rejections. This concentration indicates that a limited number of recurring patterns dominate draft failures, suggesting that many rejections arise from systematic rather than isolated mismatches.

Observation 2: Strong context dependence within frequent patterns. Frequency alone, however, is insufficient to determine whether a rejected draft token should be accepted. Figure 2(b) analyzes the top-10 most frequent patterns and reports distribution of probability ratios assigned by the target model to the rejected draft token $\tilde{x}_i$ versus the optimal token t^* ($p_{\mathcal{M}_p}(\tilde{x}_i)/p_{\mathcal{M}_p}(t^*)$). For the same token pair, this ratio spans several orders of magnitude, ranging from values close to 1 to below 10^{-10}. This wide variance reflects strong context dependence. In some contexts, the draft token remains a plausible semantic alternative with high target confidence (e.g., `a` vs. `the`), while in others it corresponds to a factual error or hallucination that the target model appropriately suppresses.

Taken together, these observations show that exact matching is overly restrictive, discarding high-confidence draft tokens that are semantically valid, while purely frequency-based relaxation is unsafe due to context-sensitive failure cases. This motivates a verification strategy that leverages historical recurrence while enforcing confidence-aware

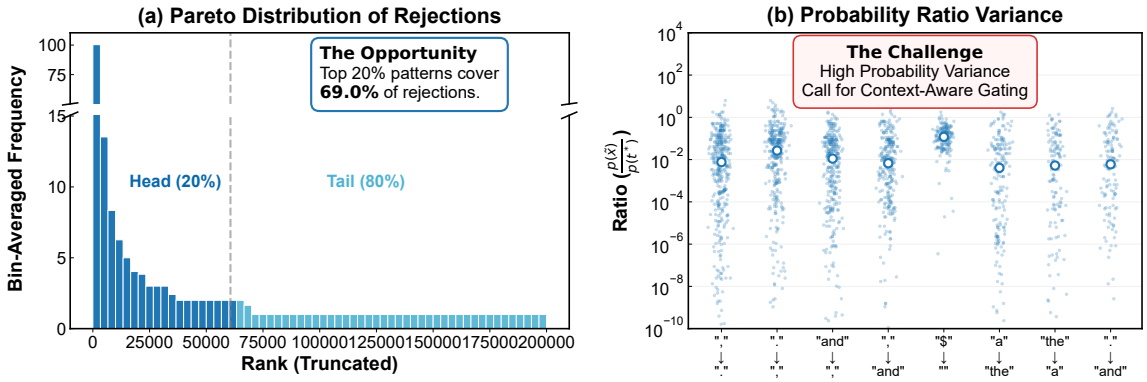


Figure 2: Statistical Analysis of False Rejections. **(a)** Long-Tailed Frequency: The distribution of rejected patterns is highly skewed; the 20% (head patterns) account for 69.0% of total rejections. This suggests a lightweight Correction Memory targeting these patterns can recover most failed speculations. **(b)** Ratio Variance: Extreme variance in probability ratios ($T = 1$) reflects a spectrum from severe hallucinations to semantically consistent alternatives. This diversity necessitates dynamic Semantic Consistency Gating to distinguish valid candidates from errors, moving beyond static relaxation.

gating, enabling selective recovery of valid draft tokens without introducing learned predictors or compromising correctness.

4 Methodology

4.1 CSD Overview

Building upon the insights from Section 3.2, we introduce **Calibrated Speculative Decoding (CSD)**, a training-free inference framework that improves the efficiency of speculative decoding by selectively relaxing rigid token-level verification.

As illustrated in Figure 1(a), conventional speculative decoding enforces exact token matching between the draft and target models. This strict verification can lead to unnecessary rejections when the two models produce lexically different yet functionally comparable tokens in context (e.g., discarding `5=10` due to a trivial `x` vs. `*` mismatch), thereby reducing the effective acceptance rate and wasting computation.

To address this limitation, CSD adopts a dual-stage inference protocol guided by the principle of *"Frequency-Guided Candidate Selection, Probability-Guarded Acceptance."* As shown in Figure 1(b), this protocol is realized through two complementary, lightweight modules. First, **Online Correction Memory (OCM)** acts as the proposal module by exploiting the heavy-tailed (Pareto) distribution of rejection patterns. It maintains a compact memory of frequent divergence pairs to propose alternative candidates during rejections. Second, **Semantic Consistency Gating (SCG)** serves as the verification module, mitigating the risk of proposal relaxation. By enforcing a confidence-based acceptance criterion derived from the target model, SCG filters out low-confidence substitutions that may lead to inconsistencies. Together, these modules form a unified inference protocol (Algorithm 1), allowing CSD to recover benign rejections on the fly without retraining.

4.2 Online Correction Memory

OCM instantiates the *Frequency-Guided Candidate Selection* stage of CSD. It records frequently observed divergence pairs $(\tilde{x}_i, t^*)$, where the draft token $\tilde{x}_i$ differs lexically from the verified target token t^*, yet such divergences recur systematically during speculative decoding. These recurring patterns serve as empirical priors for proposing alternative candidates under future rejections. The workflow of OCM is divided into two phases.

Phase 1: Prior Calibration (Initialization). Before deployment, we perform a lightweight calibration using standard speculative decoding on an unlabeled task corpus. This process collects decoding-level divergence statistics to initialize the memory table $\mathcal{T}$.

Importantly, this calibration does not involve parameter updates and only captures co-occurrence statistics between the draft and target models. It serves to establish a generic prior for the model pair, rather than task-specific tuning.

Phase 2: Online Evolution (Inference). During inference, OCM adapts to the test-time distribution via two mechanisms. First, **Dynamic Update** increments the frequency of any verified divergence $(\tilde{x}_i, t^*)$ in $\mathcal{T}$ (Algorithm 1, Line 18), en-

Algorithm 1 Calibrated Speculative Decoding with Online Adaptation

Require: Target and Draft Models M_p, M_q, Input Prefix x, Lookahead Steps γ, Temperature T, Memory $\mathcal{T}$ (OCM), Thresholds τ (SCG), λ (Freq.)

1: **for** $i = 1$ to γ **do**
2: $q_i(x) \leftarrow M_q^{(T)}(\cdot \mid x, \tilde{x}_{<i})$
3: $\tilde{x}_i \sim q_i(x)$
4: **end for**
5: $p_{1:\gamma+1}(\cdot) \leftarrow M_p^{(T)}(\cdot \mid x, \tilde{x}_{1:\gamma})$
6: $n \leftarrow 0$
7: **for** $i = 1$ to γ **do**
8: $r \sim U[0, 1]$
9: $\alpha \leftarrow \min(1, \frac{p_i(\tilde{x}_i)}{q_i(\tilde{x}_i)})$
10: **if** $r \leq \alpha$ **then**
11: $n \leftarrow n + 1$
12: **else**
13: $p'_{res}(x) \leftarrow \text{norm}(\max(0, p_i(x) - q_i(x)))$
14: $t^* \sim p'_{res}(x)$
15: {▷ CSD Logic: Attempt Rescue}
16: $is\_freq \leftarrow (\mathcal{T}[(\tilde{x}_i, t^*)] \geq \lambda)$
17: $is\_safe \leftarrow (\frac{p_i^{(T=1)}(\tilde{x}_i)}{p_i^{(T=1)}(t^*)} \geq \tau)$
18: UPDATEMEMORY$(\mathcal{T}, \tilde{x}_i, t^*)$
19: **if** $is\_freq \wedge is\_safe$ **then**
20: $n \leftarrow n + 1$ {**Rescued!**}
21: **else**
22: **return** $x \oplus [\tilde{x}_1, \ldots, \tilde{x}_n, t^*]$
23: **end if**
24: **end if**
25: **end for**
26: $t \sim p_{\gamma+1}(x)$
27: **return** $x \oplus [\tilde{x}_1, \ldots, \tilde{x}_\gamma, t]$

abling CSD to capture domain-specific vocabulary or stylistic shifts online. Second, **Candidate Proposal** ensures a divergence is proposed as a rescue candidate only if it satisfies a frequency threshold $\mathcal{T}[(\tilde{x}_i, t^*)] \geq \lambda$ (Line 16). This filtering suppresses stochastic mismatches and focuses on systematic patterns, consistent with observations in Section 3.2.

4.3 Semantic Consistency Gating

While OCM leverages historical priors to propose alternative candidates, these priors are inherently context-agnostic. However, token validity in language generation is strongly conditioned on the local context, and a frequent historical substitution may still be inappropriate in a given instance.

To bridge the gap between static historical priors and dynamic decoding contexts, SCG instantiates the *Probability-Guarded Acceptance* principle by implementing a context-aware verification mechanism that validates each correction candidate conditioned on the target model's instantaneous confidence

Semantic Admissibility Check. SCG assesses whether a proposed token $\tilde{x}_i$ lies within a reasonable confidence margin relative to the target model's preferred token t^*. Rather than computing explicit probability ratios, we perform this check directly in the logit space for efficiency:

$$\log\left(\frac{p_i^{(T=1)}(\tilde{x}_i)}{p_i^{(T=1)}(t^*)}\right) = \underbrace{z_i(\tilde{x}_i) - z_i(t^*)}_{\text{Zero-overhead}} \geq \log\tau \tag{3}$$

where $z_i(\cdot)$ denotes the raw logits produced by the target model. This formulation avoids redundant Softmax computations, is invariant to sampling temperature, and incurs negligible computational overhead.

Gating Logic. We adopt a relatively lenient threshold τ, acknowledging that valid continuations are not restricted to the single top-ranked token. By allowing candidates with non-negligible confidence under the target model, SCG accommodates benign lexical variations while effectively rejecting low-confidence deviations that may cause contextual inconsistency or hallucination.

5 Experiments

5.1 Experimental Setup

Models & Benchmarks. To evaluate the versatility of CSD, we employ two representative model configurations: (1) Llama series (Dubey et al., 2024), pairing `Llama-3-70B-Instruct` (Target) with `Llama-3.2-1B-Instruct` (Draft); and (2) Qwen-2.5 series (Qwen et al., 2025), pairing `Qwen-2.5-72B-Instruct` (Target) with `Qwen-2.5-7B-Instruct` (Draft). Standardized evaluations are conducted using the `lm-evaluation-harness` (Gao et al., 2024) on four datasets spanning diverse capabilities: GSM8K (Cobbe et al., 2021) and Minerva-Math500 (Hendrycks et al., 2021; Lewkowycz et al., 2022) for mathematical reasoning, HumanEval (Chen, 2021) for code generation, and CNN/DailyMail (Hermann et al., 2015) for summarization.

Table 1: **Main results** across Llama-3 (70B/1B) and Qwen-2.5 (72B/7B) families. We report **Acc** (Task accuracy), **Tp** (Throughput in tokens/s), and **Spd** (Speedup relative to vanilla decoding). **Avg.** represents the mean speedup across all benchmarks. Best results are highlighted in **bold**.

	GSM8K (8-shot, Pass@1)			MATH500 (4-shot, Pass@1)			HumanEval (0-shot, Pass@1)			CNN/DM (0-shot, ROUGE-L)			Avg.
Method	**Acc**	**Tp**	**Spd**	**Acc**	**Tp**	**Spd**	**Acc**	**Tp**	**Spd**	**Acc**	**Tp**	**Spd**	**Spd**
Llama-3 Series													
Vanilla Decoding	92.6	13.4	1.00×	46.0	15.2	1.00×	76.8	15.9	1.00×	20.3	15.3	1.00×	1.00×
SpecDecode	92.3	23.3	1.74×	45.4	28.8	1.89×	76.8	30.1	1.90×	20.3	22.8	1.49×	1.75×
Lossy SD ($\tau = 0.6$)	92.3	24.2	1.80×	**49.4**	30.6	2.00×	78.0	32.3	2.04×	20.3	23.6	1.55×	1.85×
SWIFT	92.6	12.9	0.96×	46.4	16.5	1.09×	76.8	17.5	1.10×	20.2	17.5	1.14×	1.07×
Lookahead	92.6	11.1	0.83×	46.6	12.5	0.82×	76.8	14.4	0.91×	20.3	12.5	0.82×	0.84×
CSD (Ours)	**92.6**	**24.5**	**1.83**×	48.0	**35.5**	**2.33**×	**79.3**	**37.0**	**2.33**×	**20.3**	**24.4**	**1.59**×	**2.02**×
Qwen-2.5 Series													
Vanilla Decoding	91.7	13.9	1.00×	82.2	15.4	1.00×	88.4	14.9	1.00×	19.9	15.2	1.00×	1.00×
SpecDecode	91.5	22.2	1.59×	81.4	29.6	1.92×	**89.0**	26.9	1.81×	19.9	20.2	1.33×	1.66×
CSD (Ours)	**92.2**	**25.2**	**1.81**×	**83.2**	**32.8**	**2.12**×	88.4	**29.0**	**1.95**×	**19.9**	**23.7**	**1.57**×	**1.86**×

Table 2: **Comparison with Semantic Verification Methods. AR** denotes the Acceptance Rate. Other metrics (**Acc**, **Tp**, **Spd**) follow the same definitions as in Table 1. Best results are highlighted in **bold**.

	GSM8K				MATH500				HumanEval				CNN/DM				Avg.			
Method	**Acc**	**Tp**	**Spd**	**AR**	**Acc**	**Tp**	**Spd**	**AR**	**Acc**	**Tp**	**Spd**	**AR**	**Acc**	**Tp**	**Spd**	**AR**	**Acc**	**Tp**	**Spd**	**AR**
Vanilla Decoding	92.6	13.4	1.00×	0.0%	46.0	15.2	1.00×	0.0%	76.8	15.9	1.00×	0.0%	**20.3**	15.3	1.00×	0.0%	58.9	14.9	1.00×	0.0%
SpecDecode	92.3	22.0	1.64×	46.2%	45.4	28.2	1.85×	42.9%	76.8	25.6	1.61×	39.9%	20.3	17.2	1.13×	23.2%	58.7	23.2	1.56×	38.0%
Fly	91.8	23.7	1.77×	53.1%	46.8	32.0	2.10×	51.4%	76.8	31.3	1.97×	48.2%	20.3	17.5	1.14×	26.7%	58.9	26.1	1.75×	44.8%
Reflect Verification	92.0	22.6	1.69×	**62.0%**	**49.6**	29.6	1.94×	58.2%	**78.0**	30.6	1.93×	**51.7%**	20.2	**22.7**	**1.49**×	**38.7%**	**59.9**	26.4	1.76×	**52.6%**
CSD	**92.5**	**24.6**	**1.84**×	58.3%	49.4	**35.2**	**2.31**×	**59.3%**	76.8	**33.4**	**2.11**×	49.8%	20.2	20.2	1.32×	33.3%	59.7	**28.4**	**1.89**×	50.2%

Baselines. We evaluate CSD against a suite of training-free decoding strategies across four categories: First, as standard benchmarks for strictly lossless generation, we employ Vanilla Decoding and Standard Speculative Decoding (SpecDecode) (Leviathan et al., 2023; Chen et al., 2023). Second, to investigate the limits of memory-free probability relaxation, we evaluate Static Lossy Speculative Decoding (Static Lossy SD), which solely employs confidence gating (Eq. 3). In this setting, we tune the threshold $\tau \approx 0.6$ to identify the *"lossless boundary"*the most aggressive relaxation possible without degrading downstream performance. Third, we compare against advanced acceleration methods, including Swift (Xia et al., 2024), which implements on-the-fly self-speculation via adaptive layer-skipping and tree-based decoding, and Lookahead Decoding (Fu et al., 2024), which achieves draft-free acceleration through parallel Jacobi iteration. We further compare CSD with recent concurrent methods: *Fly* (Li et al., 2025), which employs entropy-based gating and deferred consistency validation; and *Reflective Verification* (Wang et al., 2025), which utilizes reflective prompting and dual-copy templates to fuse multi-level logits.

Implementation Details. We implement CSD using PyTorch and Hugging Face Transformers (Paszke et al., 2019; Wolf et al., 2020). All experiments are conducted on a server node equipped with 8 NVIDIA H20 GPUs. For each inference session, we allocate 2 GPUs to load the 70B/72B target models. All evaluations are performed in a single-batch setting.

Main Evaluation Protocol: For the primary results (Section 5.2), we set the CSD lookahead steps to $\gamma = 6$, frequency threshold to $\lambda = 6$, and semantic gating threshold to $\tau = 0.01$.

Semantic Baseline Protocol: For a fair comparison with concurrent methods (Section 5.3), we unify the speculative length to $\gamma = 15$, aligning with *Fly*'s default setting. Since official implementations are currently unavailable, we meticulously re-implemented these frameworks: for *Fly*, we set the deferred window to 6 and optimized its entropy threshold to 0.05; for *Reflective Verification*, we strictly adhered to the original hyperparameter configurations. This unified protocol ensures an equitable evaluation of CSD against the latest semantic-aware paradigms.

Calibration: CSD is initialized with a calibration set of 2,000 samples for most tasks, extend-

Table 3: **Ablation study** of CSD components on MATH500 and HumanEval. **AR** denotes the Acceptance Rate. Note that individual modules (*w/* OCM or *w/* SCG) incur accuracy degradation compared to the baseline, while our integrated CSD framework achieves optimal balance.

	MATH500				HumanEval			
Variant	**Acc**	**AR**	**Tp**	**Spd**	**Acc**	**AR**	**Tp**	**Spd**
SpecDecode (Baseline)	45.4	63.6%	28.8	1.00×	76.8	59.7%	30.1	1.00×
SD **w/** OCM	37.8	83.1%	36.6	1.27×	70.7	71.6%	37.2	1.24×
SD **w/** SCG	43.6	88.7%	38.2	1.33×	70.7	88.6%	44.4	1.48×
CSD (Ours)	**48.0**	**79.6%**	**35.5**	**1.23**×	**79.3**	**67.9%**	**37.0**	**1.23**×

ing to 8,000 for MATH500 to ensure comprehensive coverage. We sample from the training sets of GSM8K, CNN/DailyMail, and original MATH (for MATH500). For HumanEval, we strictly use the MBPP training set (Austin et al., 2021) as a proxy to ensure zero-shot integrity and prevent data leakage. During this phase, we set the temperature $T = 0.6$ to broaden the CSD's exposure to diverse valid token pairs. Importantly, this calibration is a one-time, offline procedure that introduces zero online overhead during inference. The process is highly efficient, taking approximately 1.5 hours per 1,000 samples on two H20 GPUs.

Evaluation: To ensure reproducibility and eliminate stochastic variance in speedup measurements, all final performance evaluations are conducted using greedy decoding ($T = 0$).

5.2 Main Results

Table 1 compares CSD against baselines across Llama-3 and Qwen-2.5 families. Overall, CSD consistently yields the highest throughput while maintaining target model quality.

Performance on Llama-3. CSD outperforms standard speculative decoding on the Llama-3-70B/1B pair. It achieves an average speedup of **2.02**×, surpassing SpecDecode[1] (1.75×) by a wide margin, with gains peaking at **2.33**× on MATH500 and HumanEval. Additional evaluations on complex instruction-following (IFEval) and long-context (RULER) benchmarks further validate CSD's generalization (see Appendix A). Crucially, CSD maintains the accuracy of vanilla decoding across most standard benchmarks, while notably surpassing the baseline on HumanEval (**+2.5** points) and MATH500 (**+2.0** points). We hypothesize that the draft model enables the system to escape the local optima of greedy decoding by proposing valid alternative trajectories.

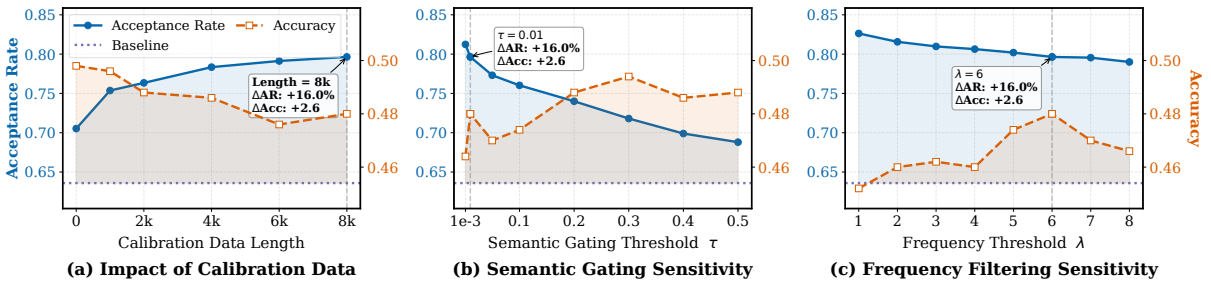
Advantage over Lossy SD. The performance of *Lossy SD* ($\tau = 0.6$) confirms the prevalence of "neutral divergence"valid tokens that differ from the greedy path. However, static gating lacks granularity. A fixed global threshold cannot perfectly distinguish acceptable deviations from errors, necessitating a conservative setting ($\tau = 0.6$) to ensure safety. This rigidity limits its ability to accept lower-probability but valid tokens, resulting in only marginal speedups over standard SpecDecode (e.g., **1.85**× vs. 1.75× on Avg). CSD leverages fine-grained filtering to unlock significantly higher throughput (Avg. **2.02**×).

Comparison with Advanced Methods. Complex acceleration schemes often yield suboptimal results on large-scale models. Lookahead can even yield negative speedups: its Jacobi decoding reduces the number of steps but incurs excessive FLOPs, creating a compute bottleneck on 70B models. Similarly, SWIFT achieves only modest gains (∼1.07×) due to a structural trade-off between minimizing speculation overhead and maintaining acceptance rate. For example, a low layer-skip rate (e.g., 45%) results in a computationally heavy draft model, while a higher skip rate causes the acceptance rate to collapse. In contrast, CSD improves the acceptance rate with negligible overhead, allowing these gains to translate directly into higher throughput.

Generalization to Qwen-2.5. On Qwen-2.5 series (72B/7B), CSD also raises the average speedup from **1.66**× to **1.86**×. Notably, CSD peaked on MATH500 with a **2.12**× speedup and **+1.0 point** accuracy gain. This confirms that our strategy generalizes well to different vocabularies and structures.

Analysis of Rescued Tokens. To characterize the specific sources of CSD's efficiency gains, we analyzed 100 randomly sampled "rescued" to-

[1]Standard speculative decoding is theoretically lossless; however, minor fluctuations (e.g., 92.6 vs. 92.3 on GSM8K) may arise from non-deterministic parallel reduction kernels during verification.



(a) Impact of Calibration Data **(b) Semantic Gating Sensitivity** **(c) Frequency Filtering Sensitivity**

Figure 3: **Sensitivity analysis on MATH500.** (a) **Calibration**: AR plateaus beyond 2k–4k samples and peaks at 8k. (b) **Semantic Gating**: Optimal trade-off at $\tau = 0.01$. (c) **Frequency**: $\lambda = 6$ maximizes accuracy. Dotted lines indicate the accuracy and AR baselines of SpecDecode.

kens from the Llama-3-70B/1B inference traces. We categorize the false rejections mitigated by CSD into four primary types of semantically neutral divergences: **(1) Math Formatting** (≈45%), such as equivalent delimiters (e.g., `$$` vs. `\[`) or trailing punctuation (e.g., `$,` vs. `$`); **(2) Punctuation & Spacing** (≈20%), including minor variations like commas versus conjunctions (e.g., ",” vs. “and”) or line breaks (e.g., `\n` vs. `\n\n`); **(3) Lexical Synonyms** (≈20%), where draft predictions are semantically identical (e.g., “get” vs. “obtain”); and **(4) Reasoning Connectives** (≈15%), such as structural transitional words (e.g., “So” vs. “Therefore”). Standard speculative decoding routinely penalizes these valid, mutually replaceable tokens. By safely salvaging these surface-level equivalents, CSD effectively breaks the acceptance rate ceiling of strict exact-match rules without disrupting the foundational reasoning process or downstream accuracy.

5.3 Comparison with Concurrent Semantic Baselines

We evaluate CSD against state-of-the-art semantic baselines using a Llama-3-70B/1B pair ($\gamma = 15$). Table 2 shows that CSD achieves higher end-to-end throughput by bypassing the structural and computational bottlenecks of prior paradigms.

(1) Efficiency over Window-based Verification (vs. *Fly*): *Fly* relies on a deferred window that requires exact-match consistency for subsequent W tokens. This strategy is inherently sensitive to minor stylistic variations and often fails at sequence boundaries where future context is unavailable. In contrast, CSD performs token-independent semantic validation via statistical gating. This granularity allows CSD to recover valid drafts even at boundaries, achieving a superior average acceptance rate (50.2% vs. 44.8%).

(2) Efficiency over Prompt-based Verification (vs. *Reflective*): While *Reflective Verification* achieves a high acceptance rate (52.6%) through logit fusion, its wall-clock speedup (1.76×) lags behind CSD (1.89×). This gap stems from its use of verification templates that duplicate tokens and append probes, which significantly increases input length and computational latency during the target model's forward pass. CSD avoids this overhead by directly utilizing standard target logits, ensuring that improved acceptance rates translate directly into physical acceleration.

In summary, CSD shifts the semantic verification bottleneck from complex structural dependencies to a simple, zero-overhead filtering problem.

5.4 Ablation Results

To assess the contribution of each component, we evaluate two variants: (1) **SD w/ OCM**, employing only the Online Correction Memory; and (2) **SD w/ SCG**, using only Semantic Consistency Gating with $\tau = 0.01$.

Risks of Coarse-grained Filtering. As shown in Table 3, while individual modules improve throughput, they trigger significant accuracy regressions. Specifically, SD w/ SCG achieves the maximum speedup (1.48× on HumanEval) by surging acceptance rate to 88.6%, but at the cost of reducing accuracy to 70.7. Similarly, SD w/ OCM enhances MATH500 throughput (AR 83.1%) but fails to maintain precision, with accuracy dropping to 37.8. These results underscore the risk of coarse-grained filtering: relaxing verification without the joint protection of semantic gating and frequency-guided proposals introduces harmful hallucinations.

Safe Acceleration through Synergy. In contrast, the full **CSD** framework effectively synergizes both modules. By adaptively identifying “neutral divergence,” CSD boosts the acceptance rate by 16.0% (from 63.6% to 79.6%) on

Table 4: Generalization to Universal Calibration. We compare CSD calibrated on task-specific data (**Spec.**) against CSD using a **Univ.** (Universal) calibration set. **Avg.** denotes the macro-average across all tasks.

	Acc		AR		Speedup	
Task	Spec.	Univ.	Spec.	Univ.	Spec.	Univ.
HumanEval	79.2	78.0	67.9%	71.1%	2.33×	2.45×
MATH500	48.0	50.0	79.6%	74.3%	2.32×	2.23×
GSM8K	92.6	92.8	76.4%	74.5%	1.83×	1.80×
CNN/DM	20.3	20.3	55.3%	54.4%	1.59×	1.59×
Avg.	**60.0**	**60.3**	**69.8%**	**68.6%**	**2.02×**	**2.02×**

MATH500 while further boosting accuracy to **48.0**. Similarly, on HumanEval, it improves the acceptance rate to 67.9% (vs. 59.7% baseline) while achieving superior accuracy (**79.3**). These results demonstrate that CSD's fine-grained approach safely unlocks valid alternative trajectories, providing an optimal trade-off between inference speed and reasoning integrity.

5.5 Sensitivity Analysis

We evaluate the impact of CSD's core hyperparameters on the MATH500 benchmark. As shown in Fig. 3(a), the acceptance rate (AR) increases with the amount of calibration data. While the performance gain starts to plateau after 2k–4k samples, it reaches the maximum at 8k samples. Importantly, even without any initial calibration data, CSD achieves a 7% improvement in AR over standard speculative decoding, thanks to its online dynamic update mechanism. This suggests that CSD can effectively capture the divergence patterns with minimal offline overhead. Furthermore, Fig. 3(b) and (c) demonstrate the robustness of our filtering mechanism. CSD achieves an optimal balance between acceleration and reasoning integrity at $\tau = 0.01$ and $\lambda = 6$. Notably, across a wide range of thresholds, CSD consistently maintains higher accuracy and AR than the standard SpecDecode baseline (dotted lines), confirming its reliability and ease of deployment.

5.6 Generalization of Calibration

To evaluate cross-domain robustness, we calibrate CSD using a Universal Calibration Set of 6,000 sequences from six diverse RedPajama (Weber et al., 2024) sub-domains. This set is strictly disjoint from all evaluation benchmarks. As shown in Table 4, downstream accuracy is rigorously preserved across all tasks. While transitioning to this general corpus causes marginal acceptance rate fluctuations, the average speedup remains exceptionally stable at 2.02×, matching task-specific calibration. Notably, performance on HumanEval improves, likely due to the diverse syntactic patterns in RedPajama compared to the narrow MBPP domain. These results confirm that CSD avoids overfitting and effectively captures ubiquitous linguistic redundancies, enabling a single offline calibration to accelerate downstream applications without re-tuning.

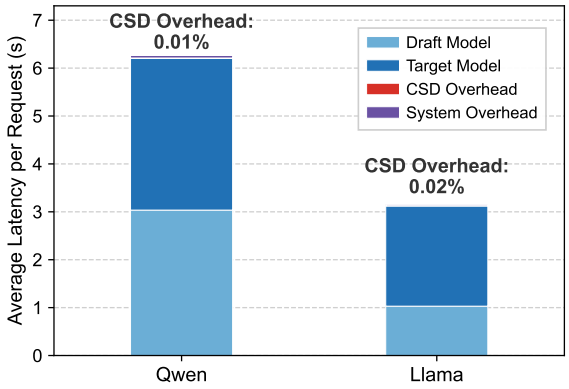


Figure 4: **Inference Latency Breakdown.** A detailed latency analysis for Qwen-72B/7B and Llama-70B/1B. The results highlight that the CSD overhead is negligible relative to the total inference time.

5.7 Analysis of Runtime Overhead

To evaluate computational efficiency, we profile end-to-end latency on 50 GSM8K requests (4-shot CoT). Using high-precision timing, we isolate the steady-state latency for each component. As shown in Figure 4, CSD's algorithmic overhead accounts for a negligible **0.01%–0.02%** of the total time. This confirms that our method introduces virtually no extra cost, ensuring that any increase in acceptance rate results in actual speedup.

6 Conclusion

We present CSD, a lightweight, training-free framework that recovers valid tokens from false rejections in SD. It combines OCM, which captures recurring divergence patterns, with SCG, which verifies candidates using model confidence rather than exact matching. Experiments show that CSD improves acceptance rates while maintaining accuracy and incurring negligible overhead.

Limitations

- **Departure from Distributional Exactness:** Unlike standard speculative decoding which uses rejection sampling to guarantee that output distributions are identical to the target model, CSD employs a heuristic acceptance criterion. This relaxation prioritizes inference speed over strict statistical parity. While empirical results are strong, the performance of this heuristic may not consistently generalize to unseen tasks or novel domains.

- **Dependency on Draft Quality:** The effectiveness of CSD is highly sensitive to the draft model's output quality. In instances where the draft model generates unreliable or hallucinatory sequences, the candidates fail to pass the semantic gating threshold. Such divergence leads to a significant drop in the acceptance rate, limiting the practical speedup to near auto-regressive levels.

- **High-Concurrency Scalability:** Our evaluation primarily focuses on latency reduction in small-batch settings. The mechanism for coordinating Online Correction Memory (OCM)specifically the synchronization of real-time frequency updates across multiple concurrent requests—remains to be fully explored in high-throughput environments. The impact of potential contention in these critical sections on system-level performance warrants further investigation.Furthermore, CSD currently lacks integration with industrial-grade inference engines (e.g., vLLM), which is necessary to realize its full potential in large-scale, production-ready deployments.

Acknowledgements

This work was partially supported by the National Key Research and Development Program of China (2024YFE0204300), National Natural Science Foundation of China (Grant No.62402311), Natural Science Foundation of Shanghai (Grant No.24ZR1433700), Key Research and Development Program of Shanghai (25LN3201200), and Alibaba Innovative Research Program. Fangxin Liu and Li Jiang are corresponding authors.

References

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

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Gregor Bachmann, Sotiris Anagnostidis, Albert Pumarola, Markos Georgopoulos, Artsiom Sanakoyeu, Yuming Du, Edgar Schönfeld, Ali Thabet, and Jonas Kohler. 2025. Judge decoding: Faster speculative sampling requires going beyond model alignment. *arXiv preprint arXiv:2501.19309*.

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. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Yukang Chen, Weihao Cui, Han Zhao, Ziyi Xu, Xiaoze Fan, Xusheng Chen, Yangjie Zhou, Shixuan Sun, Bingsheng He, and Quan Chen. 2025. Towards high-goodput llm serving with prefill-decode multiplexing. *arXiv preprint arXiv:2504.14489*.

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. 2024. Sequoia: Scalable, robust, and hardware-aware speculative decoding. *arXiv preprint arXiv:2402.12374*.

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*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, and 1 others. 2024. Layerskip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12622–12642.

Tianyu Fu, Yi Ge, Yichen You, Enshu Liu, Zhihang Yuan, Guohao Dai, Shengen Yan, Huazhong Yang, and Yu Wang. 2025. R2r: Efficiently navigating divergent reasoning paths with small-large model token routing. *arXiv preprint arXiv:2505.21600*.

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

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.

Amir Gholami, Zhewei Yao, Sehoon Kim, Coleman Hooper, Michael W Mahoney, and Kurt Keutzer. 2024. Ai and memory wall. *IEEE Micro*, 44(3):33–39.

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*.

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.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857.

Jinze Li, Yixing Xu, Guanchen Li, Shuo Yang, Jinfeng Xu, Xuanwu Yin, Dong Li, Edith CH Ngai, and Emad Barsoum. 2025. Training-free loosely speculative decoding: Accepting semantically correct drafts beyond exact match. *arXiv preprint arXiv:2511.22972*.

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

Fangxin Liu, Ning Yang, Haomin Li, Zongwu Wang, Zhuoran Song, Songwen Pei, and Li Jiang. 2024a. Spark: Scalable and precision-aware acceleration of neural networks via efficient encoding. In *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 1029–1042. IEEE.

Fangxin Liu, Wenbo Zhao, Zhezhi He, Yanzhi Wang, Zongwu Wang, Changzhi Dai, Xiaoyao Liang, and Li Jiang. 2021. Improving neural network efficiency via post-training quantization with adaptive floating-point. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5281–5290.

Jiahao Liu, Qifan Wang, Jingang Wang, and Xunliang Cai. 2024b. Speculative decoding via early-exiting for faster llm inference with thompson sampling control mechanism. *arXiv preprint arXiv:2406.03853*.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, and 1 others. 2024. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pages 932–949.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, and 1 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

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

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin

Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodolà. 2023. Accelerating transformer inference for translation via parallel decoding. *arXiv preprint arXiv:2305.10427*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Yixuan Wang, Yijun Liu, Yuzhuang Xu, Yang Xu, Qingfu Zhu, Wanxiang Che, and 1 others. 2025. Think before you accept: Semantic reflective verification for faster speculative decoding. *arXiv preprint arXiv:2505.18629*.

Maurice Weber, Daniel Y. Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, Ben Athiwaratkun, Rahul Chalamala, Kezhen Chen, Max Ryabinin, Tri Dao, Percy Liang, Christopher Ré, Irina Rish, and Ce Zhang. 2024. Redpajama: an open dataset for training large language models. *NeurIPS Datasets and Benchmarks Track*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, and 1 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Heming Xia, Yongqi Li, Jun Zhang, Cunxiao Du, and Wenjie Li. 2024. Swift: On-the-fly self-speculative decoding for llm inference acceleration. *arXiv preprint arXiv:2410.06916*.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. 2024. Draft& verify: Lossless large language model acceleration via self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11263–11282.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023a. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. 2023b. Distillspec: Improving speculative decoding via knowledge distillation. *arXiv preprint arXiv:2310.08461*.

A Extended Evaluation on Instruction Following and Long-Context Tasks

To further demonstrate the versatility of CSD across highly complex scenarios, we conducted additional evaluations on IFEval (Zhou et al., 2023a) (measuring strict instruction-following capabilities) and RULER (Hsieh et al., 2024) (evaluating long-context understanding). These experiments were conducted on the Llama-3-70B/1B model pair, with all other experimental settings kept strictly consistent with our main evaluation protocol.

As shown in Table 5, CSD excels on both benchmarks. On IFEval, CSD achieves a 1.77× speedup while perfectly preserving vanilla decoding accuracy (76.8). On the RULER benchmark, massive prefill overheads typically bottleneck end-to-end latency for all decoding methods. However, CSD's superior acceptance rate effectively mitigates this bottleneck, yielding a 1.37× speedup (compared to a marginal 1.02× for Standard SD) while notably boosting accuracy to 95.9.

Table 5: Performance comparison on IFEval (Instruction Following) and RULER (Long-Context). **AR** denotes Acceptance Rate, and **Spd** denotes the end-to-end speedup relative to Vanilla.

	IFEval			RULER		
Method	**Acc**	**AR**	**Spd**	**Acc**	**AR**	**Spd**
Vanilla	76.8	-	1.00×	88.1	-	1.00×
Standard SD	**77.4**	39.7%	1.65×	88.0	73.6%	1.02×
CSD (Ours)	76.8	**42.3%**	**1.77**×	**95.9**	**83.2%**	**1.37**×