BEYOND THE TARGET: FROM IMITATION TO COLLABORATION IN SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative decoding accelerates large language model inference by letting a small draft model propose tokens for parallel verification by a larger target. For tasks evaluated by answer correctness, however, agreement with the target is an imperfect criterion: a weaker draft can occasionally offer a continuation that leads to a better final answer. Exploiting these disagreements requires learning from delayed outcomes while retaining the efficiency of parallel verification. We present Collaborative Speculative Decoding (CoSpec), a framework that learns arbitration within speculative decoding from the outcomes of complete rollouts. An arbitrator jointly observes the context, draft block, and target verification tokens through a hybrid attention mask, and decides whether to retain the draft or fall back to the target at each mismatch. Reinforcement learning combines final-answer correctness with accepted-length shaping to train these decisions. CoSpec retains parallel target verification while optimizing task utility rather than preserving the target distribution. We evaluate two model families on reasoning, code, and knowledge benchmarks. With LLaMA-3.3-70B as the target, CoSpec achieves a mean speedup of $3.96\times$ over target-only decoding and raises the mean score on GSM8K, HumanEval, and MBPP from 90.05 to 91.72. These results support learning from downstream outcomes as an effective way to exploit draft–target complementarity within speculative decoding.

1 INTRODUCTION

Speculative decoding accelerates large language model inference by letting a smaller draft model propose tokens that a larger target verifies in parallel (Leviathan et al., 2023; Chen et al., 2023). Exact methods preserve the target's greedy output or sampling distribution, making the draft useful insofar as it reduces the cost of reproducing the target. For tasks such as mathematical reasoning and code generation, however, the objective is to produce a correct answer as quickly as possible. Reproducing the target also reproduces its mistakes. This leaves room for a draft to contribute useful alternatives as well as computational savings (Figure 1).

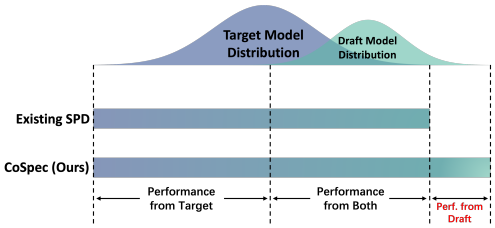


Figure 1: Draft–target complementarity: a weaker draft can offer useful alternatives at target failures. CoSpec learns which mismatches to retain from complete rollout outcomes. The target-bounded comparison illustrates methods that reproduce the target's behavior.

A weaker model need not fail on every problem that defeats a stronger one. In our LLaMA-3.3-70B/3.1-8B evaluation, the union of problems solved independently by the two models exceeds those solved by the target alone on all five benchmarks in Table 2, including MATH-500 and GPQA. Our counterfactual analysis also identifies reached mismatch states where retaining the draft token leads to a correct completion while selecting the target token does not, under the same continuation policy (Table 3). This complementarity creates an opportunity, but exploiting it requires learning which disagreements improve the final answer.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Prior work has explored this broader objective through speculative cascades (Narasimhan et al., 2025), token selection in CoSD (Wang et al., 2025d), and learned verification in Judge Decoding and AutoJudge (Bachmann et al., 2025; Garipov et al., 2025). DIVERSED also learns dynamic draft–target distribution mixtures using sequence-level task rewards and an acceptance regularizer (Wang et al., 2026). CoSpec focuses on explicit mismatch decisions informed by the entire verification round: it jointly examines the draft block and target verification tokens, then learns the consequences of retaining or rejecting mismatches through complete speculative rollouts.

This learning problem has two coupled difficulties. A locally plausible choice can lead to an incorrect answer, and each choice changes the states encountered later in the rollout. Evaluating both continuations to completion at every mismatch would also consume the computation that speculation is meant to save. A practical arbitrator therefore needs to learn from complete outcomes during training and make its decisions from the evidence available within the current verification round.

We introduce **CoSpec**, which learns such a policy while keeping the draft and target frozen. A separate arbitrator jointly processes the context, the draft block, and the target verification tokens through a hybrid attention mask. Each mismatch decision can therefore use evidence from the entire round. Matched tokens are accepted automatically; at a mismatch, the policy either retains the draft token and continues through the block, or emits the target token and starts a new round. The target still verifies the draft in one teacher-forced pass. The arbitrator uses these existing target predictions, and its own forward pass is included in the reported cost.

We initialize the arbitrator with judge-labeled mismatch preferences, then train it with reinforcement learning on complete speculative rollouts. Final-answer correctness, accepted length, and penalties on failed trajectories jointly shape the learning signal. These signals serve complementary purposes: retaining alternatives with equal downstream utility can lengthen accepted spans, while selecting better draft continuations can improve correctness. CoSpec learns these tradeoffs from complete rollouts and optimizes task utility without preserving the target distribution.

We evaluate CoSpec with LLaMA and Qwen pairs across reasoning, code, and knowledge tasks. In the LLaMA greedy setting, it achieves a mean speedup of $3.96\times$ and raises the mean score on GSM8K, HumanEval, and MBPP from 90.05 to 91.72. On the full 5,000-problem MATH test set, it solves 138 problems missed by the target while losing 68; the paired gain remains significant after Holm correction (Appendix G). Transfer evaluations, training controls, and serving measurements further characterize the method's benefits and costs.

Our contributions are threefold:

- We formulate mismatch arbitration in terms of downstream branch utility and connect instance-level model complementarity with counterfactual diagnostics at reached speculative states.
- We develop an explicit mismatch policy that jointly reads the draft block and target verification tokens, trained on complete speculative rollouts with correctness-conditioned acceptance and failure rewards.
- We demonstrate task-score gains with end-to-end speedups across two model families. Training controls and counterfactual diagnostics support the contribution of rollout optimization and characterize how the policy uses draft–target disagreements.

2 Related Work

Target-preserving speculative decoding. Speculative decoding accelerates inference through draft proposals and parallel target verification (Leviathan et al., 2023; Chen et al., 2023). Later methods improve proposals or execution through learned drafters, retrieval, early exit, dynamic trees, block verification, and systems optimizations (Cai et al., 2024; Li et al., 2024a;b; 2025b; He et al., 2024; Quan et al., 2025; Elhoushi et al., 2024; Chen et al., 2024b; Sun et al., 2025; Li et al., 2025a). CoSpec retains parallel verification but changes the decision at mismatches to optimize task utility.

Relaxed verification and utility-oriented speculative decoding. Judge Decoding trains a verification head on annotated error tokens (Bachmann et al., 2025), and AutoJudge mines such la-

bels automatically by testing whether substituting a draft token changes the target model's final answer (Garipov et al., 2025). Fuzzy speculative decoding, FLy, SelfJudge, and reflective, sequential approximate, and pivot-aware verification further relax acceptance to improve efficiency while controlling quality (Holsman et al., 2025; Li et al., 2026; Yoon et al., 2025; Wang et al., 2025c; Zhong et al., 2025; Ziashahabi et al., 2025). Reward-Guided Speculative Decoding regenerates low-reward reasoning steps (Liao et al., 2025); AASD combines retrieval-based drafts with alignment sampling and conditional verification (Wang et al., 2025a). DIVERSED trains context-dependent distribution mixing weights with sequence-level task rewards and an acceptance regularizer, using both models' prefix hidden states (Wang et al., 2026).

Cascades and model collaboration. Cascades and routers decide which model should answer a query (Chen et al., 2024a; Ong et al., 2025; Ding et al., 2024), and ensembling or fusion methods aggregate the outputs of several models (Jiang et al., 2023; Wang et al., 2025b). Closer to our setting, Big Little Decoder hands control to the large model through confidence-based fallback and rollback policies (Kim et al., 2023); speculative cascades replace the acceptance test with a deferral rule derived from both models' token distributions and can exceed the quality of the large model (Narasimhan et al., 2025); and CoSD fuses the knowledge of two models by deciding at each disagreement, via a probability rule or a decision tree trained on token-level labels, whether to replace the draft token (Wang et al., 2025d).

Positioning. CoSpec combines explicit mismatch actions with a joint view of the entire draft block and target verification tokens. This differs from CoSD's confidence features and DIVERSED's prefix-conditioned distribution mixing. Its complete-rollout training uses final correctness, accepted spans, and failure penalties to optimize these actions, extending beyond AutoJudge's answer-preservation supervision. The contribution lies in this arbitration design and its empirical analysis; the arbitrator's forward pass is included in all reported speedups.

3 METHOD

CoSpec pairs a joint view of the verification round with a learning signal from completed speculative rollouts. The former supplies evidence for each mismatch decision; the latter trains the policy on the consequences of those decisions.

3.1 PRELIMINARIES

Notation. Let x denote the input prompt and $\mathcal{V}$ the vocabulary. We consider a large target model $\mathcal{M}_{\mathrm{T}}$ and a smaller draft model $\mathcal{M}_{\mathrm{D}}$ that share $\mathcal{V}$. Both models decode autoregressively.

Speculative decoding. Speculative decoding operates in rounds. Let $y^{(<r)}$ be the verified prefix at the beginning of round r and let K be the draft length. The draft model proposes

$$\hat{\mathbf{y}}^{(r)} = (\hat{y}_1^{(r)}, \ldots, \hat{y}_K^{(r)}) \tag{1}$$

conditioned on $(x, y^{(<r)})$. The target model then performs one teacher-forced forward pass over $[x, y^{(<r)}, \hat{y}_1^{(r)}, \ldots, \hat{y}_K^{(r)}]$, producing one next-token prediction at each of the $K+1$ rightmost positions:

$$(\tilde{y}_1^{(r)}, \ldots, \tilde{y}_{K+1}^{(r)}). \tag{2}$$

The first K predictions verify the draft positions and $\tilde{y}_{K+1}^{(r)}$ is the bonus token after the final draft token. We define

$$\Delta_i^{(r)} = \mathbb{1}[\hat{y}_i^{(r)} = \tilde{y}_i^{(r)}]. \tag{3}$$

Exact-match speculative decoding accepts draft tokens until the first mismatch and then emits the target token at that position. The number of generated tokens in round r is

$$s^{(r)} = \begin{cases} \min\{i \mid \Delta_i^{(r)} = 0\}, & \exists i : \Delta_i^{(r)} = 0, \\ K+1, & \text{otherwise.} \end{cases} \tag{4}$$

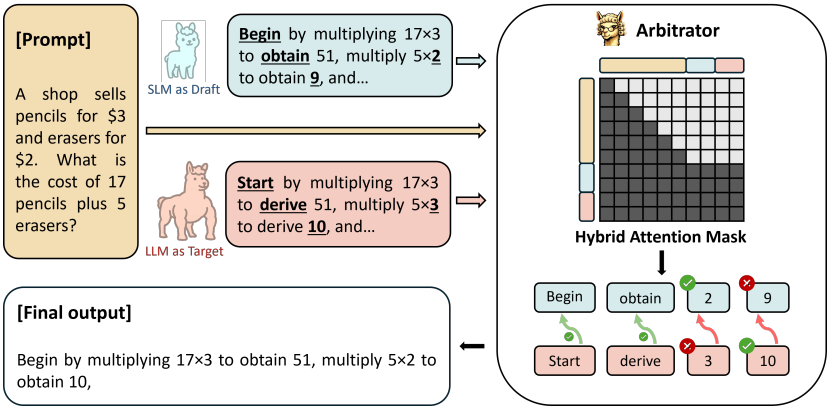


Figure 2: Overview of CoSpec. The draft proposes a block and the target verifies it in one teacher-forced pass. A learned arbitrator receives the context, draft block, and target verification tokens. It uses a hybrid attention mask and decides at each mismatch whether to keep the draft token or fall back to the target token.

For a run of R rounds, the mean accepted length is

$$\tau = \frac{1}{R}\sum_{r=1}^{R} s^{(r)}. \tag{5}$$

We report speedup as the end-to-end throughput ratio relative to autoregressive target decoding under the same hardware, batching, prompt, and output length setting.

Stochastic decoding. Let T denote the sampling temperature. For $T = 1$, distribution-preserving baselines use speculative sampling with acceptance ratio correction. CoSpec arbitrates between sampled draft and target verification candidates to optimize task utility without preserving the target distribution.

3.2 A Utility View of Collaborative SPD

Let $U(x, y) \in [0, 1]$ be the task utility of a completed response, such as exact answer correctness or pass@1. At a reached mismatch state $m = (x, y^{(<r)}, i, \hat{y}^{(r)}_{1:K}, \tilde{y}^{(r)}_{1:K+1}, \Delta^{(r)}_{1:K})$, define two counterfactual branch utilities:

$$u_D(m) = \mathbb{E}[U(x, Y) \mid m, \text{force } \hat{y}^{(r)}_i], \quad u_T(m) = \mathbb{E}[U(x, Y) \mid m, \text{force } \tilde{y}^{(r)}_i]. \tag{6}$$

Here Y is the completed response. The draft branch appends $\hat{y}^{(r)}_i$ and continues scanning the current block, whereas the target branch appends $\tilde{y}^{(r)}_i$, terminates the current round, and resumes CoSpec from the next round. The expectation is over the fixed reference continuation policy π_{ref} under these dynamics. A target-faithful verifier uses the target branch when a mismatch is rejected. An oracle arbitrator chooses the branch with larger expected utility.

Observation 3.1 (Local complementarity gap)**.** Fix π_{ref}, and let $\mathcal{S}$ be the distribution of reached mismatch states induced by the prompts, draft/target decoding, and π_{ref}. Then

$$\mathbb{E}_{m\sim\mathcal{S}}[\max\{u_D(m), u_T(m)\}] - \mathbb{E}_{m\sim\mathcal{S}}[u_T(m)] = \mathbb{E}_{m\sim\mathcal{S}}[(u_D(m) - u_T(m))_+], \tag{7}$$

where $(z)_+ := \max(z, 0)$.

This follows directly from $\max\{u_D, u_T\} = u_T + (u_D - u_T)_+$, and formalizes the opportunity available whenever the draft branch has higher downstream utility on a nonzero subset of reached mismatch states. It characterizes what an ideal arbitrator could gain over target-faithful verification; it is a motivation for CoSpec, not a guarantee for a learned policy. CoSpec therefore does not aim to make every draft token pass target verification, but to approximate this utility-maximizing branch selector while preserving the parallel verification structure that gives speculative decoding its speed advantage. Figure 2 gives an overview of this procedure, where the arbitrator compares the draft and target candidates within each speculative round and decides which branch to follow at mismatches.

3.3 COLLABORATIVE ARBITRATION

CoSpec learns an arbitration policy π_θ that decides whether a mismatch should keep the draft token or fall back to the target token. Let

$$c^{(r)} = [x, y^{(<r)}] \tag{8}$$

be the prompt and verified prefix. For round r, the arbitrator input is

$$q^{(r)} = \left[c^{(r)}, [\text{SEP}], \hat{y}^{(r)}_{1:K}, [\text{SEP}], \tilde{y}^{(r)}_{1:K}, [\text{SEP}]\right]. \tag{9}$$

The bonus token $\tilde{y}^{(r)}_{K+1}$ is omitted because arbitration only acts on the K paired draft–target candidates. Let $l_c \triangleq |c^{(r)}|$ and let $l = l_c + 2K + 3$ be the total input length. We apply a hybrid attention pattern: the context portion attends causally within itself, while the draft–target region uses fully bidirectional attention within that region and may also attend to the entire context. Information flows from context to draft/target and within draft/target in both directions, but never from draft/target back to context. The draft and target models themselves remain causal and frozen. The hybrid mask only changes the diagnostic view available to the arbitrator.

The policy backbone is a language model with LoRA (Hu et al., 2021) parameters ψ. Let $\mathcal{A}$ denote the hybrid attention mask described above. It produces hidden states

$$h^{(r)} = \text{LM}_\psi(q^{(r)}, \mathcal{A}). \tag{10}$$

For each draft position $i \in \{1, \ldots, K\}$, a lightweight head g_ϕ maps the hidden state at the position of $\hat{y}^{(r)}_i$ within $q^{(r)}$ to a scalar logit:

$$z^{(r)}_i = g_\phi\left(h^{(r)}_{l_c+1+i}\right). \tag{11}$$

The action $a^{(r)}_i \in \{0, 1\}$ denotes whether to accept the draft token:

$$a^{(r)}_i = 1 \Rightarrow \hat{y}^{(r)}_i, \qquad a^{(r)}_i = 0 \Rightarrow \tilde{y}^{(r)}_i. \tag{12}$$

With $\theta = \{\psi, \phi\}$ collecting all trainable parameters, the acceptance probability is

$$\pi_\theta(a^{(r)}_i = 1 \mid q^{(r)}) = \sigma(\tilde{z}^{(r)}_i), \tag{13}$$

where $\sigma(\cdot)$ denotes the logistic sigmoid and

$$\tilde{z}^{(r)}_i = \begin{cases} +\infty, & \Delta^{(r)}_i = 1, \\ z^{(r)}_i, & \Delta^{(r)}_i = 0. \end{cases} \tag{14}$$

Matched tokens are forced-accepted and do not create trainable decisions. At mismatches, CoSpec thresholds or samples from the learned policy.

Within a round, CoSpec stops at the first rejected mismatch:

$$s^{(r)}_{\text{CoSpec}} = \begin{cases} \min\{i \mid \Delta^{(r)}_i = 0 \wedge a^{(r)}_i = 0\}, & \exists i : \Delta^{(r)}_i = 0 \wedge a^{(r)}_i = 0, \\ K + 1, & \text{otherwise.} \end{cases} \tag{15}$$

Thus $s^{(r)}_{\text{CoSpec}} \in \{1, \ldots, K + 1\}$ is the round-terminating index, equal to the number of tokens emitted in round r. If the first rejected mismatch occurs at position i, the output appends all tokens before i and then appends $\tilde{y}^{(r)}_i$. If no mismatch is rejected, the output appends all K draft tokens plus the bonus target token. This keeps the standard round structure while allowing draft tokens to pass through selected mismatches.

3.4 REINFORCEMENT LEARNING TRAINING

We optimize the arbitrator with a two-stage pipeline. The supervised stage initializes π_θ from judge-labeled mismatch preferences over rollouts of vanilla SPD. We defer its details to Appendix C and denote the resulting policy by π_{ref}. The reinforcement learning stage optimizes decisions under full CoSpec rollouts, so training includes the states reached after the policy retains a mismatched draft token. All target and draft model weights remain frozen, only the arbitrator LoRA parameters ψ and head ϕ are trained.

Rollouts and actions. For each prompt x, we generate a group of M CoSpec rollouts $\{\xi^{(1)}, \ldots, \xi^{(M)}\}$, each recording the draft block, the target verification tokens, and the arbitrator actions of every round until EOS. During training, mismatch actions are sampled from π_θ; at evaluation time, a mismatched draft token is accepted if and only if $\pi_\theta(a_i^{(r)} = 1 \mid q^{(r)}) > \lambda$. Because a round ends at its first rejected mismatch, only the mismatches up to that point affect the output. We call them the *effective decisions* of round r, $\mathcal{I}_r^{(m)} = \{i : \Delta_i^{(r)} = 0,\ i \le \min(s_{\mathrm{CoSpec}}^{(r)}, K)\}$, which is empty for a round without mismatches.

Reward shaping. Final correctness determines which reward regime applies: successful rollouts reward longer accepted spans, while failed rollouts penalize risky acceptance and unsuccessful fallback. Each rollout receives a terminal correctness signal $\mathrm{Corr}(\xi^{(m)}) \in \{0, 1\}$, and each effective decision receives a shaped reward $g_{r,i}$. For correct rollouts, we reward accepting a useful span beyond the first mismatch. Let $i_r^\star$ be the position of the first mismatch in round r; every effective decision of the round receives $g_{r,i} = \rho_r$, where

$$\rho_r = \frac{s_{\mathrm{CoSpec}}^{(r)} - i_r^\star}{(K+1) - i_r^\star} \tag{16}$$

is the fraction of the post-mismatch span that the round accepts. For incorrect rollouts, we penalize two decision types: accepting a draft token that the target strongly disfavors, and rejecting a mismatch, which commits to a target branch that can also be wrong. Let $\mathcal{S}_{r,\mathrm{acc}}^{(m)}$ contain the mismatches accepted before the first rejection in round r, and let $\mathcal{S}_{r,\mathrm{rej}}^{(m)}$ contain the first rejected mismatch, or be empty if no mismatch is rejected. The failure reward is

$$g_{r,i} = -\alpha |\ell_i^{(r)}| \mathbb{1}[i \in \mathcal{S}_{r,\mathrm{acc}}^{(m)}] - \beta \mathbb{1}[i \in \mathcal{S}_{r,\mathrm{rej}}^{(m)}], \tag{17}$$

where $|\ell_i^{(r)}| = -\log p_{\mathcal{M}_\mathrm{T}}(\hat{y}_i^{(r)} \mid x, y^{(<r)}, \hat{y}_{<i}^{(r)})$ is the surprisal of the draft token under the target. The surprisal term is a risk regularizer, not a target-authority rule: it discourages high-variance exploration when the draft token lies far outside the target support on failed rollouts. The rejection penalty prevents the policy from treating fallback to the target as a free action when the target branch can also fail. The return $J(\xi^{(m)})$ of a rollout is the sum of $g_{r,i}$ over all its effective decisions.

Policy update. We normalize returns within each prompt group, following the value-free design of GRPO (Shao et al., 2024). If all rollouts for a prompt are incorrect, the shaped negative returns still rank arbitration traces by relative harm, and we downweight such groups by η_{fail}:

$$\mathrm{Adv}^{(m)} = \eta_{\mathrm{fail}} \frac{J(\xi^{(m)}) - \mu_x}{\sigma_x + \epsilon}. \tag{18}$$

For all-correct groups we use the same normalization with $\eta_{\mathrm{fail}} = 1$. For mixed groups, we rescale correct returns using the total magnitude of incorrect returns before standardization. Let $\mathcal{G}_+(x)$ and $\mathcal{G}_-(x)$ be correct and incorrect rollouts. We set

$$\gamma_x = \frac{\sum_{\xi \in \mathcal{G}_-(x)} |J(\xi)|}{\sum_{\xi \in \mathcal{G}_+(x)} J(\xi) + \epsilon}, \tag{19}$$

and define $\bar{J}^{(m)} = \gamma_x J(\xi^{(m)})$ for $\xi^{(m)} \in \mathcal{G}_+(x)$ and $\bar{J}^{(m)} = J(\xi^{(m)})$ otherwise. We apply (18) to these returns, recomputing μ_x, σ_x over the group. The ϵ-stabilized rescaling adjusts relative magnitudes; subtracting the recomputed μ_x centers the advantages.

We allocate rollout advantages to effective decisions: for correct rollouts we set $\mathrm{Adv}_{r,i}^{(m)} = \rho_r \cdot \mathrm{Adv}^{(m)}$, and for incorrect rollouts $\mathrm{Adv}_{r,i}^{(m)} = \frac{|g_{r,i}|}{\sum_{r',i'} |g_{r',i'}|} \cdot \mathrm{Adv}^{(m)}$. These weights emphasize larger ρ_r on correct rollouts and allocate failed-rollout advantages by shaped reward magnitude. A failed rollout can have positive advantage when its return exceeds the group mean. The policy minimizes the PPO-style clipped objective (Schulman et al., 2017)

$$\begin{aligned} \mathcal{L}_{\mathrm{RL}} = -\sum_{m,r,i} \min\Big(r_{r,i}^{(m)}(\theta) \mathrm{Adv}_{r,i}^{(m)},\ \mathrm{clip}(r_{r,i}^{(m)}(\theta), 1-\epsilon_{\mathrm{clip}}, 1+\epsilon_{\mathrm{clip}}) \mathrm{Adv}_{r,i}^{(m)} \Big) \\ - \beta_H \mathcal{H}(\pi_\theta) + \beta_{\mathrm{KL}} \mathrm{KL}(\pi_\theta \| \pi_{\mathrm{ref}}), \end{aligned} \tag{20}$$

where π_{ref} is the supervised warm-up policy and $r_{r,i}^{(m)}(\theta) = \pi_\theta(a_i^{(r)} \mid q^{(r)}) / \pi_{\theta_{\mathrm{old}}}(a_i^{(r)} \mid q^{(r)})$ is the probability ratio between the current and rollout-time policies.

Table 1: Main speed–quality comparison, averaged over three evaluation runs. Speed is end-to-end throughput relative to target-only autoregressive decoding; τ is the mean number of output tokens per target invocation. Score is accuracy (GSM8K) or pass@1 (HumanEval and MBPP), in percent. Mean is the unweighted average over the three benchmarks. Results are rounded to two decimal places after aggregation; see Appendix C.3.

Setting	Target	Method	GSM8K			HumanEval			MBPP			Mean		
			Speed	τ	Score	Speed	τ	Score	Speed	τ	Score	Speed	τ	Score
$T = 0$	LLaMA-70B	Target only	1.00×	1.00	95.60	1.00×	1.00	85.37	1.00×	1.00	89.18	1.00×	1.00	90.05
		SpS	2.11×	9.87	95.60	1.61×	9.87	85.37	1.68×	10.43	89.18	1.80×	10.06	90.05
		REST	1.74×	2.00	95.60	1.88×	2.22	85.37	2.01×	2.34	89.18	1.88×	2.19	90.05
		EAGLE-3	3.67×	5.63	95.60	3.94×	5.73	85.37	3.77×	5.45	89.18	3.79×	5.60	90.05
		FLy	2.68×	11.65	94.84	2.75×	12.20	84.54	2.45×	11.76	88.62	2.63×	11.87	89.33
		AASD	1.48×	1.72	95.92	1.62×	1.90	85.98	1.72×	2.08	90.03	1.61×	1.90	90.64
		CoSpec	**3.82×**	**20.31**	**96.59**	**4.13×**	**21.18**	**87.20**	**3.92×**	**20.96**	**91.38**	**3.96×**	**20.82**	**91.72**
	Qwen3-32B	Target only	1.00×	1.00	85.96	1.00×	1.00	89.63	1.00×	1.00	75.92	1.00×	1.00	83.84
		SpS	1.41×	8.31	85.96	1.29×	8.44	89.63	1.33×	9.52	75.92	1.34×	8.76	83.84
		EAGLE-3	1.87×	4.01	85.96	1.98×	4.13	89.63	1.91×	3.89	75.92	1.92×	4.01	83.84
		FLy	1.64×	9.82	85.40	1.69×	10.30	89.04	1.58×	10.55	75.42	1.64×	10.22	83.29
		AASD	1.50×	1.82	86.50	1.54×	1.89	89.55	1.59×	2.03	75.60	1.54×	1.91	83.88
		CoSpec	**2.03×**	**18.54**	**88.15**	**2.15×**	**18.71**	**92.07**	**1.97×**	**18.01**	**79.10**	**2.05×**	**18.42**	**86.44**
$T = 1$	LLaMA-70B	Target only	1.00×	1.00	95.10	1.00×	1.00	85.37	1.00×	1.00	88.73	1.00×	1.00	89.73
		SpS	2.08×	9.67	95.10	1.47×	6.29	85.37	1.51×	6.08	88.73	1.69×	7.35	89.73
		REST	1.71×	1.91	95.10	1.81×	2.13	85.37	1.94×	2.24	88.73	1.82×	2.09	89.73
		EAGLE-3	3.50×	5.31	95.10	3.74×	5.40	85.37	3.58×	5.15	88.73	3.61×	5.29	89.73
		FLy	2.85×	12.40	94.71	2.75×	12.55	84.69	2.69×	12.33	88.42	2.76×	12.43	89.27
		AASD	1.51×	1.78	95.66	1.65×	1.95	85.98	1.75×	2.16	89.40	1.64×	1.96	90.35
		CoSpec	**3.88×**	**20.91**	**96.80**	**4.15×**	**21.32**	**87.40**	**3.97×**	**21.13**	**90.11**	**4.00×**	**21.12**	**91.44**
	Qwen3-32B	Target only	1.00×	1.00	86.13	1.00×	1.00	89.69	1.00×	1.00	76.11	1.00×	1.00	83.98
		SpS	1.39×	8.28	86.13	1.22×	8.27	89.69	1.27×	9.41	76.11	1.29×	8.65	83.98
		EAGLE-3	1.75×	3.81	86.13	1.81×	4.05	89.69	1.83×	3.80	76.11	1.80×	3.89	83.98
		FLy	1.60×	9.54	85.48	1.66×	10.05	89.12	1.58×	10.12	75.38	1.61×	9.90	83.33
		AASD	1.48×	1.80	86.62	1.52×	1.87	89.62	1.58×	2.00	75.72	1.53×	1.89	83.99
		CoSpec	**1.92×**	**18.02**	**88.39**	**2.03×**	**18.58**	**92.28**	**1.96×**	**17.97**	**80.09**	**1.97×**	**18.19**	**86.92**

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We evaluate CoSpec with two target and draft pairs from the same model family: LLaMA-3.3-70B-Instruct with LLaMA-3.1-8B-Instruct, and Qwen3-32B with Qwen3-4B. Unless otherwise stated, the arbitrator uses the draft model backbone with LoRA adapters and a binary decision head. The default draft length is $K = 25$, and the default inference threshold is $\lambda = 0.6$. The arbitrator is trained on OpenR1-Math-220k (Open R1 Team, 2025) and then evaluated without task-specific retraining. The main benchmarks are GSM8K (Cobbe et al., 2021), HumanEval (Chen et al., 2021), and MBPP (Austin et al., 2021). The transfer suite includes MT-Bench (Zheng et al., 2023), TriviaQA (Joshi et al., 2017), GPQA (Rein et al., 2024), MGSM (Shi et al., 2023), and MATH-500 (Lightman et al., 2024).

We compare target-preserving methods (vanilla speculative sampling, SpS; REST; EAGLE-3), loosely verified FLy, and utility-improving AASD. Comparisons with CoSD (Wang et al., 2025d) and AutoJudge (Garipov et al., 2025) on a subset of model pairs and benchmarks appear in Appendix E. Target-only decoding is the reference for speedup and task score. Each main configuration is evaluated in three runs; Appendix C.3 details the metrics and aggregation. All speedups include arbitrator overhead, about 6% of per-round latency in the default LLaMA setting, and decrease under high-concurrency serving (Appendix J.1).

4.2 MAIN RESULTS

Table 1 shows that CoSpec improves task score while accelerating target-only decoding across both model families. All methods use the same target model, prompts, and evaluation metric; Section 4.4 examines the roles of added capacity, supervision, and rollout training.

Target-preserving baselines retain the target's greedy predictions or sampling distribution, while relaxed baselines increase acceptance but do not consistently improve task score in these evaluations. CoSpec learns which disagreements can benefit the completed output. Under greedy decoding, CoSpec improves the mean score from 90.05 to 91.72 for LLaMA-70B with a 3.96× speedup,

and from 83.84 to 86.44 for Qwen3-32B with a 2.05× speedup; under sampling, it improves the mean score from 89.73 to 91.44 and from 83.98 to 86.92, respectively.

The comparison with CoSD's learned decision tree provides a closer test of collaborative decoding (Table 10). On Qwen3-32B/4B HumanEval, CoSpec raises the score from 89.75 to 92.07 and the speedup from 1.55× to 2.15×. The implementation details and the AutoJudge comparison are given in Appendix E.

Paired exact McNemar tests (Appendix G) provide additional evidence on the full 5,000-problem MATH test set: CoSpec solves 138 problems that target-only decoding misses while losing 68 ($p \approx 1.2 \times 10^{-6}$, unadjusted). This gain remains significant after Holm correction across the four tests; the gains on GSM8K, MBPP, and HumanEval do not.

4.3 Complementarity and Arbitration Diagnostics

Table 2 reports examples solved by the target, draft, and either model (oracle union). The target outperforms the draft in every row, yet their union consistently exceeds target performance. CoSpec recovery, $(\mathrm{CoSpec} - \mathrm{Target})/(\mathrm{Union} - \mathrm{Target})$ in correct counts, measures aggregate net gain relative to the oracle gap, without identifying which draft-only successes CoSpec recovers. On MATH-500 and GPQA, the union adds 37 and 26 successes over the target, while CoSpec's net gains amount to 19% and 12% of these gaps. Complementarity thus extends to harder tasks, but is harder to exploit.

Table 2: Complementarity and recovery.

Pair	Benchmark	Total	Target correct	Draft correct	Oracle union	CoSpec recovery
LLaMA-70B/8B	GSM8K	1319	1261	1113	1281	65%
LLaMA-70B/8B	HumanEval	164	140	131	146	50%
LLaMA-70B/8B	MBPP	499	445	414	466	52%
LLaMA-70B/8B	MATH-500	500	324	231	361	19%
LLaMA-70B/8B	GPQA	448	245	178	271	12%

Table 3 evaluates arbitration at reached mismatch states on held-out prompts ($K = 25$, $\lambda = 0.6$). At each state, we force either the draft or target verification token and complete both branches with the same frozen CoSpec policy and decoding settings; correctness follows the benchmark metric. Share gives each mismatch type's frequency, Draft not worse the fraction where the draft branch has at least the target branch's final utility, CoSpec draft the draft acceptance rate, and Agreement the fraction where the selected branch has utility at least as high as the alternative.

The aggregate agreement of 94.1% includes states where both branches have equal utility. More directly informative are the unequal-utility states: CoSpec accepts the draft in 78% of draft-better cases but only 20% of target-better cases. This difference supports sensitivity to relative branch utility at the states the policy reaches.

Among accepted mismatches, 81.7% have equal measured utility (52.8% both correct and 28.9% both wrong), allowing longer acceptance without changing counterfactual correctness. Draft-better and target-better states account for 12.7% and 5.6%, respectively. These diagnostics distinguish harmless extensions, beneficial choices, and harmful accepts; the end-to-end results measure their combined effect on speed and correctness. The observed false accepts concentrate in interpretable failure modes: on GSM8K and MATH-500, 82% are locally plausible numerical substitutions or premature reasoning shortcuts (Appendix H).

4.4 Ablation Studies

Generalization. Without task-specific retraining, CoSpec improves speed and task score across dialog, factual QA, hard reasoning, multilingual reasoning, and MATH (Table 4). Stable draft–target interaction patterns may help explain this transfer; Appendix B discusses this hypothesis and the limits of the evidence.

Table 3: Counterfactual mismatch analysis.

Category	Share	Draft not worse	CoSpec draft	Agreement
All mismatches	100.0%	82.0%	64.4%	94.1%
Draft better	10.5%	100.0%	78.0%	78.0%
Target better	18.0%	0.0%	20.0%	80.0%
Both correct	41.5%	100.0%	82.0%	100.0%
Both wrong	30.0%	100.0%	62.0%	100.0%

Table 4: Transfer evaluation.

Benchmark	Target	SpS	CoSpec	Score
MT-Bench	9.38	1.33×	1.98×	9.43
TriviaQA	87.50	1.29×	2.31×	89.20
GPQA	54.69	1.31×	2.27×	55.36
MGSM	88.97	1.73×	3.35×	90.02
MATH-500	64.80	1.58×	2.96×	66.20

Table 5: Compute and training controls.

Method	GSM8K		HumanEval	
	Speed	Score	Speed	Score
Target only	1.00×	85.96	1.00×	89.63
SpS 4B draft	1.41×	85.96	1.29×	89.63
SpS 8B draft	1.22×	85.96	1.09×	89.63
Target RL math	1.00×	88.23	1.00×	86.57
SpS RL target	1.38×	88.23	1.23×	86.57
CoSpec	2.03×	88.15	2.15×	92.07

Table 6: Training and reward ablations.

Variant	Judge	Warm-up	RL signal	Speedup	Score
SFT only	30B	yes	none	1.71×	86.93
No judge + RL	none	reject all	full	2.05×	91.07
4B judge + RL	4B	yes	full	2.11×	91.80
Terminal only	30B	yes	terminal	1.62×	90.71
No length reward	30B	yes	no length	1.58×	91.52
No risk penalty	30B	yes	no α	2.23×	89.86
No rejection penalty	30B	yes	no β	2.01×	90.94
No rebalancing	30B	yes	no rebalance	2.08×	91.21
CoSpec	30B	yes	full	2.15×	92.07

Is the gain a consequence of extra supervision or compute? CoSpec adds a trained arbitrator, so we compare alternative uses of additional supervision and model capacity (Table 5; Appendix F). An 8B draft roughly matches the combined draft–arbitrator size but slows SpS without improving score. Applying the same math RL to the target raises GSM8K to 88.23 but lowers HumanEval to 86.57 without acceleration, and SpS with the RL-tuned target reaches only 1.38× and 1.23×. Reranking four complete responses scores 88.70 on GSM8K (above CoSpec) and 91.46 on HumanEval, but runs at 0.26×. CoSpec improves both accuracy and speed over target-only decoding on both evaluated benchmarks, supporting the value of applying the utility signal within mismatch arbitration.

Training and reward ablations. Table 6 evaluates warm-up, judge quality, rollout optimization, and reward design on Qwen3-32B/4B HumanEval. Judge-supervised SFT alone learns local acceptability and scores 86.93, below target-only decoding (89.63), whereas rollout-level RL reaches 92.07. No-judge RL also exceeds the target (91.07), showing that improvement does not require judge warm-up; stronger judge initialization improves it further. Removing length shaping lowers speedup, while removing the risk penalty trades accuracy for speed. Terminal-only training and the remaining ablations also weaken score, speed, or both. At fixed initialization, subset size, and budget, hard-only RL training exceeds target-only accuracy on both benchmarks at lower speedup, while mixed prompts yield the best accuracy (Appendix D).

5 Conclusion

We presented **collaborative speculative decoding (CoSpec)**, which treats draft–target disagreements as opportunities to improve the final task outcome. Even a weaker draft can offer useful alternatives when the target would make a mistake. CoSpec learns which alternatives to retain from the outcomes of complete speculative rollouts, including the states reached by its own arbitration decisions. The arbitrator jointly uses draft and target verification evidence to balance correctness and accepted length while retaining parallel verification. Across two model families, CoSpec improves task scores over target-only decoding while accelerating inference. In the LLaMA-3.3-70B greedy setting, it achieves a mean 3.96× speedup and raises the mean score on GSM8K, HumanEval, and MBPP from 90.05 to 91.72. Training controls and counterfactual diagnostics support complete-rollout optimization as a practical way to exploit draft–target complementarity. CoSpec thus turns the draft from a target approximator into a complementary proposer, contributing both computational savings and useful alternatives.

AI USE STATEMENT

Generative AI tools were used for typo revising, and were not used for method design or experimental analysis.

ETHICS STATEMENT

This work studies inference algorithms for large language models and does not involve human subjects, personally identifiable information, or newly collected data. All models and datasets are publicly available and are used in accordance with their licenses (Appendix L); in particular, no verbatim GPQA questions are reproduced. Because CoSpec optimizes task utility rather than exact target-distribution preservation, its outputs may deviate from those of the target model, including in safety-relevant behavior. In our behavioral checks, CoSpec closely preserves the target's refusal and instruction-following behavior (Appendix H), but these checks are not a formal safety guarantee, and we recommend retaining external safety filters in deployment. Limitations and broader impacts are discussed in Appendices K and M.

REPRODUCIBILITY STATEMENT

The CoSpec formulation, arbitrator architecture, and training objectives are specified in Section 3. Models, benchmarks, metrics, and default hyperparameters are listed in Section 4.1 and Appendix A; the warm-up, reinforcement learning, and inference hyperparameters are given in Appendix C; the evaluation protocol and the validation-only hyperparameter selection are described in Appendix I; the inference loop is summarized in Appendix J; and additional controls, paired significance tests, and failure analyses are reported in Appendices F, G, and H. All models and datasets used are publicly available (Appendix L), and the code will be released upon acceptance.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program Synthesis with Large Language Models. *arXiv preprint arXiv:2108.07732*, 2021. URL https://arxiv.org/abs/2108.07732.

Gregor Bachmann, Sotiris Anagnostidis, Albert Pumarola, Markos Georgopoulos, Artsiom Sanakoyeu, Yuming Du, Edgar Schönfeld, Ali Thabet, and Jonas Kohler. Judge Decoding: Faster Speculative Sampling Requires Going Beyond Model Alignment. In *International Conference on Learning Representations*, pp. 40894–40914, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/656d6174fd5ffb01c843f269649ab5cb-Abstract-Conference.html.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM Inference Acceleration Framework with Multiple Decoding Heads. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 5209–5235. PMLR, 2024. URL https://proceedings.mlr.press/v235/cai24b.html.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating Large Language Model Decoding with Speculative Sampling. *arXiv preprint arXiv:2302.01318*, 2023. URL https://arxiv.org/abs/2302.01318.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024a. URL https://openreview.net/forum?id=cSimKw5p6R.

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

Ziyi Chen, Xiaocong Yang, Jiacheng Lin, Chenkai Sun, Kevin Chen-Chuan Chang, and Jie Huang. Cascade Speculative Drafting for Even Faster LLM Inference. In *Advances in Neural Information Processing Systems*, 2024b. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/9cb5b083ba4f5ca6bd05dd307a2fb354-Abstract-Conference.html.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: Cost-Efficient and Quality-Aware Query Routing. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/b47d93c99fa22ac0b377578af0a1f63a-Abstract-Conference.html.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. 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)*, pp. 12622–12642, 2024.

Roman Garipov, Fedor Velikonivtsev, Ivan Ermakov, Ruslan Svirschevski, Vage Egiazarian, and Max Ryabinin. AutoJudge: Judge Decoding Without Manual Annotation. In *Advances in Neural Information Processing Systems*, volume 38, pp. 94605–94642, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/87e498bbb5a41b3d2e456218b4ceb939-Abstract-Conference.html.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason D. Lee, and Di He. REST: Retrieval-Based Speculative Decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1582–1595. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.naacl-long.88/.

Maximilian Holsman, Yukun Huang, and Bhuwan Dhingra. Fuzzy Speculative Decoding for a Tunable Accuracy-Runtime Tradeoff. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 26257–26273. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-acl.1346/.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv preprint arXiv:2106.09685*, 2021. URL https://arxiv.org/abs/2106.09685.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. LLM-Blender: Ensembling Large Language Models with Pairwise Ranking and Generative Fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023. URL https://aclanthology.org/2023.acl-long.792/.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017. URL https://aclanthology.org/P17-1147/.

Sehoon Kim, Karttikeya Mangalam, Suhong Moon, Jitendra Malik, Michael W. Mahoney, Amir Gholami, and Kurt Keutzer. Speculative Decoding with Big Little Decoder. In *Advances in Neural Information Processing Systems*, volume 36, pp. 39236–39256, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/7b97adeafa1c51cf65263459ca9d0d7c-Abstract-Conference.html.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast Inference from Transformers via Speculative Decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

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

Jinze Li, Yixing Xu, Guanchen Li, Shuo Yang, Jinfeng Xu, Xuanwu Yin, Dong Li, Edith C. H. Ngai, and Emad Barsoum. Training-Free Loosely Speculative Decoding: Accepting Semantically Correct Drafts Beyond Exact Match. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2511.22972.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative Sampling Requires Rethinking Feature Uncertainty. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 28935–28948. PMLR, 2024a. URL https://proceedings.mlr.press/v235/li24bt.html.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-2: Faster Inference of Language Models with Dynamic Draft Trees. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7421–7432. Association for Computational Linguistics, 2024b. URL https://aclanthology.org/2024.emnlp-main.422/.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling Up Inference Acceleration of Large Language Models via Training-Time Test. In *Advances in Neural Information Processing Systems*, volume 38, 2025b. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/c7b5a35ea98b62512a869c19ea7b03cb-Abstract-Conference.html.

Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. Reward-Guided Speculative Decoding for Efficient LLM Reasoning. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=AVeskAAETB.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's Verify Step by Step. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2305.20050.

Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Seungyeon Kim, Neha Gupta, Aditya Krishna Menon, and Sanjiv Kumar. Faster Cascades via Speculative Decoding. In *International Conference on Learning Representations*, pp. 44949–44987, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/6f43166f50f26e8d8f3edc5545b0749f-Abstract-Conference.html.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to Route LLMs with Preference Data. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=8sSqNntaMr.

Open R1 Team. OpenR1-Math-220k. Hugging Face dataset, 2025. URL https://huggingface.co/datasets/open-r1/OpenR1-Math-220k.

Guofeng Quan, Wenfeng Feng, Chuzhan Hao, Guochao Jiang, Yuewei Zhang, and Hao Henry Wang. RASD: Retrieval-augmented speculative decoding. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 6167–6177, 2025.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A Graduate-Level Google-Proof Q&A Benchmark. In *First Conference on Language Modeling*, 2024. URL https://arxiv.org/abs/2311.12022.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. Language Models are Multilingual Chain-of-Thought Reasoners. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03057.

Ziteng Sun, Uri Mendlovic, Yaniv Leviathan, Asaf Aharoni, Jae Hun Ro, Ahmad Beirami, and Ananda Theertha Suresh. Block Verification Accelerates Speculative Decoding. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=frsg32u0rO.

Jikai Wang, Zhenxu Tian, Juntao Li, Qingrong Xia, Xinyu Duan, Zhefeng Wang, Baoxing Huai, and Min Zhang. Alignment-Augmented Speculative Decoding with Alignment Sampling and Conditional Verification. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025a. URL https://aclanthology.org/2025.emnlp-main.343/.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-Agents Enhances Large Language Model Capabilities. In *International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=h0ZfDIrj7T.

Yixuan Wang, Yijun Liu, Shiyu Ji, Yuzhuang Xu, Yang Xu, Qingfu Zhu, and Wanxiang Che. Think Before You Accept: Semantic Reflective Verification for Faster Speculative Decoding. *arXiv preprint arXiv:2505.18629*, 2025c. URL https://arxiv.org/abs/2505.18629.

Ziyao Wang, Muneeza Azmat, Ang Li, Raya Horesh, and Mikhail Yurochkin. Speculate, then Collaborate: Fusing Knowledge of Language Models during Decoding. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 62345–62356. PMLR, 2025d. URL https://proceedings.mlr.press/v267/wang25e.html.

Ziyi Wang, Siva Rajesh Kasa, Ankith M S, Santhosh Kumar Kasa, Jiaru Zou, Sumit Negi, Ruqi Zhang, Nan Jiang, and Qifan Song. DIVERSED: Relaxed Speculative Decoding via Dynamic Ensemble Verification. In *Proceedings of the 29th International Conference on Artificial Intelligence and Statistics*, volume 300 of *Proceedings of Machine Learning Research*, pp. 2341–2349. PMLR, 2026. URL https://proceedings.mlr.press/v300/wang26d.html.

Kanghoon Yoon, Minsub Kim, Sungjae Lee, Joonhyung Lee, Sunghyeon Woo, Yeonjun In, Se Jung Kwon, Chanyoung Park, and Dongsoo Lee. SelfJudge: Faster Speculative Decoding via Self-Supervised Judge Verification. *arXiv preprint arXiv:2510.02329*, 2025. URL https://arxiv.org/abs/2510.02329.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2306.05685.

Meiyu Zhong, Noel Teku, and Ravi Tandon. Speeding up Speculative Decoding via Sequential Approximate Verification. *arXiv preprint arXiv:2502.04557*, 2025. URL https://arxiv.org/abs/2502.04557.

Amir Ziashahabi, Yavuz Faruk Bakman, Duygu Nur Yaldiz, Mostafa El-Khamy, Sai Praneeth Karimireddy, and Salman Avestimehr. Reject Only Critical Tokens: Pivot-Aware Speculative Decoding. *arXiv preprint arXiv:2511.00351*, 2025. URL https://arxiv.org/abs/2511.00351.

A Full Experimental Setup

Models. We evaluate two same family draft and target pairs with shared tokenizers. The first pair uses Qwen3-32B as the target and Qwen3-4B as the draft. The second pair uses LLaMA-3.3-70B-Instruct as the target and LLaMA-3.1-8B-Instruct as the draft. Unless otherwise stated, the arbitrator uses the draft backbone with LoRA adapters and a lightweight binary arbitration head. The default draft length is $K = 25$ and the default deterministic threshold is $\lambda = 0.6$.

Training and validation. The arbitrator is trained in two stages. The warm-up stage uses OpenR1-Math-220k to construct judge-labeled mismatch preferences. The RL stage uses multi-rollout CoSpec decoding with final-answer correctness and accepted-length shaping. We select K, λ, and reward coefficients on held-out validation prompts and keep them fixed for all test benchmarks. Hyperparameter sweeps are reported as ablations to characterize the tradeoff, without changing the defaults used in the main evaluations.

Benchmarks and metrics. We report exact match accuracy for GSM8K, MGSM, MATH-500, TriviaQA, and GPQA when an exact answer parser is available. We report pass@1 for HumanEval and MBPP using the official unit test harness. We report the standard judge score for MT-Bench. Throughout the paper, Score denotes the native benchmark metric.

B On the Cross-Domain Transfer of the Arbitrator

Observation. Although the arbitrator is trained only on mathematical reasoning data (OpenR1-Math-220k), it transfers without retraining to code generation, open-ended dialogue, factual QA, hard scientific reasoning, and multilingual reasoning (Table 4): it improves speed on all transfer tasks and yields positive score gains on each of them.

A plausible explanation. The arbitrator does not see task labels or task-specific features; its input is the token sequence of the context, the draft block, and the target verification tokens of the current round (Equation (9)). Many of the situations it must distinguish are defined by the relation between the two candidates rather than by the task: a draft token may be a paraphrase that the target would also accept, a valid alternative continuation, a local error that the target immediately corrects, or a drift whose harm appears only later. If the arbitrator learns to recognize such draft–target interaction patterns, and if these patterns look similar across domains, the policy would transfer beyond math. This is in line with the loosely verified line of work, in which the target's reaction to a mismatched draft token is useful evidence about whether the mismatch is harmful (Li et al., 2026; Bachmann et al., 2025). The MGSM result, which tests transfer across languages rather than tasks, is consistent with this explanation.

What the current evidence does not show. These results show that a math-trained arbitrator transfers zero-shot in our evaluated settings, but they do not establish that its learned representations are task-invariant. We have not analyzed the arbitrator's internal features, and transfer is smaller on some tasks than on math: for example, the gains on GPQA are modest. The explanation above should therefore be read as a hypothesis consistent with the observed transfer rather than as an established mechanism.

C More Implementation Details

Arbitrator architecture. The arbitrator uses the same pretrained architecture and initialization as the draft model (LLaMA-3.1-8B for the LLaMA pair and Qwen3-4B for the Qwen pair), but it is loaded as a separate model instance at inference time and does not share in-memory weights with the draft. LoRA adapters are applied to the attention projection modules of every layer, and a lightweight binary head produces the mismatch decisions. The draft and target models remain frozen throughout training.

C.1 Warm-Up via Supervised Fine-Tuning

Warm-up data. For each OpenR1-Math-220k problem, we run vanilla speculative decoding and record round traces. Each trace contains the prompt, verified prefix, draft block, target verification tokens, and mismatch indicators. We supervise only mismatch positions because match positions are forced accepted by design.

Judge labels. For each mismatch, a separate Qwen3-30B-A3B-Instruct-2507 judge scores the draft and target candidates conditioned on the current context and available within round prefix. The scalar score is converted into a soft label $y_{r,i} \in [0, 1]$ for accepting the draft. The SFT loss is

$$\mathcal{L}_{\mathrm{SFT}} = - \sum_{(r,i):\Delta_i^{(r)}=0} \Big[y_{r,i} \log \pi_\theta(a_i^{(r)} = 1 \mid q^{(r)}) + (1 - y_{r,i}) \log \pi_\theta(a_i^{(r)} = 0 \mid q^{(r)}) \Big]. \quad (21)$$

Warm-up hyperparameters. The LoRA adapters (attention projections of every layer) use rank 16, scale 32, and dropout 0.05. We use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay 0.1, learning rate 10^{-4}, and batch size 256.

C.2 Reinforcement Learning Training

The rollout group size is $M = 12$ per prompt and the global prompt count per update is 128. The rollout temperature is 1 during training. We use AdamW with learning rate 5×10^{-5}. The reward coefficients are $\alpha = 1.0$, $\beta = 0.25$, and $\eta_{\mathrm{fail}} = 0.5$. The PPO clip range is 0.2, the entropy coefficient is 0.01, the KL coefficient is 0.02, and we use four update epochs per rollout batch with gradient norm clipping at 1.0.

C.3 Inference and Score Aggregation

Decoding and arbitration. Unless otherwise stated, CoSpec uses a draft length of $K = 25$ and a deterministic arbitration threshold of $\lambda = 0.6$. Matched draft and target verification tokens are accepted automatically. At a mismatch, the draft token is retained if and only if $\pi_\theta(a_i^{(r)} = 1 \mid q^{(r)}) > \lambda$; otherwise, the target verification token is emitted and the current round ends. If all K draft tokens are retained, the target bonus token is appended. For $T = 0$, both models use greedy decoding. For $T = 1$, CoSpec samples the draft and target verification candidates at temperature one and applies the same deterministic arbitration rule. Distribution-preserving baselines use their speculative-sampling acceptance and correction rules. CoSpec optimizes task utility and does not preserve the target distribution.

Evaluation repetitions and benchmark scores. Each decoding configuration in Table 1 is evaluated in three runs. The $T = 0$ runs use greedy decoding; the $T = 1$ runs use three random seeds for generation. For benchmark b and run r, let $A_{b,r}$ denote the score computed using the benchmark's standard metric: exact-match accuracy for GSM8K and pass@1 for HumanEval and MBPP. All three scores are reported as percentages. The reported benchmark score is

$$\overline{A}_b = \frac{1}{3} \sum_{r=1}^{3} A_{b,r}. \quad (22)$$

The Mean score is the unweighted arithmetic mean across the three benchmarks,

$$A_{\mathrm{Mean}} = \frac{\overline{A}_{\mathrm{GSM8K}} + \overline{A}_{\mathrm{HumanEval}} + \overline{A}_{\mathrm{MBPP}}}{3}. \quad (23)$$

We aggregate scores before rounding and display both benchmark scores and their Mean to two decimal places. Repeated greedy evaluations of the same model on the same inputs can return identical predictions; repeating the evaluation does not itself imply variability in task score.

Interpretation of target preservation. Under greedy decoding, exact target-preserving methods reproduce the target's predictions and hence its benchmark scores. Under stochastic decoding, the corresponding guarantee concerns the output distribution: independently sampled finite evaluation sets need not have identical empirical scores.

Efficiency measurements. Speed is the end-to-end throughput ratio relative to target-only autoregressive decoding under the same hardware, batching, prompt, and output-length setting. It includes draft proposal, target verification, arbitration, and sampling overhead. For each benchmark, speedup and τ are averaged over the three evaluation runs. The quantity τ counts output tokens per target invocation, including the terminating target token or the bonus token, as in Equation (5). Mean speedup and Mean τ are unweighted arithmetic means across GSM8K, HumanEval, and MBPP.

D More Ablations

Hyperparameters. Table 7 shows that K and λ control the speed and accuracy tradeoff. Larger K increases the available span for acceptance, while lower λ accepts more mismatches. We use $K = 25$ and $\lambda = 0.6$ as the default because it provides a strong speedup while retaining the score improvement over target-only decoding.

Table 7: Ablation of hyperparameters.

Hyperparameter	Value	Speedup	τ	Score
K	15	3.51×	10.97	97.11
	20	3.69×	16.18	96.70
	25	3.82×	20.31	96.59
λ	0.4	3.96×	22.05	93.15
	0.6	3.82×	20.31	96.59
	0.8	3.51×	17.79	97.13

Hybrid attention mask. Table 8 ablates the hybrid attention mask used by the arbitrator. With the hybrid mask, the hidden state at each draft position, from which the decision is read, can attend to the whole draft block and to the target verification tokens. Under a standard causal mask, the target verification tokens come after the draft block in the input, so the representation at each draft position loses both bidirectional interaction and any access to the target evidence. The drop in speed, accepted length, and score supports the benefit of the hybrid design in this architecture; this ablation does not separate the effects of target-token visibility and bidirectional interaction.

Table 8: Ablation of the hybrid attention mask. Under the causal mask, draft positions cannot attend to the target verification tokens.

Variant	Speedup	τ	Score
w/ hybrid attention mask	3.82×	20.31	96.59
w/o hybrid attention mask	1.85×	12.10	88.13

Difficulty of RL training prompts. One may ask whether CoSpec learns useful arbitration only from easy prompts, on which many rollouts are correct. The default RL data are not dominated by such prompts: 31% of the RL training prompt groups contain no correct rollout. To test this directly, we define prompt difficulty by the success rate of the rollout group under the fixed warm-up policy, and train on equal-size easy-only, hard-only, or mixed subsets with the same initialization and optimization budget, using the Qwen3-32B/4B pair. Table 9 shows that hard-only training does not collapse: it remains above target-only decoding on both benchmarks, but its lower speedup indicates a more conservative arbitration policy. Easy-only training gives the highest speedup but the smallest accuracy gain, and the mixed distribution gives the best accuracy. Because each subset is smaller than the full RL training set, these numbers are not directly comparable with the full model in Table 1.

E Comparison with Closely Related Methods

We report comparisons with CoSD, which arbitrates locally between a draft and an assistant model (Wang et al., 2025d), and AutoJudge, a learned-verification method in the Judge Decoding

Table 9: RL training on prompts of different difficulty (Qwen3-32B/4B). Speedup is averaged over GSM8K and HumanEval.

Training subset	Speedup	GSM8K	HumanEval
Target only	1.00×	85.96	89.63
Easy-only	2.12×	87.34	90.85
Hard-only	1.87×	87.79	91.16
Mixed	2.06×	87.94	91.46

line (Garipov et al., 2025; Bachmann et al., 2025). Section 2 also discusses DIVERSED's outcome-trained distribution mixing and its distinction from CoSpec's explicit, round-informed mismatch policy.

CoSD. We use the decision-tree variant of CoSD with the Qwen3-32B/4B pair and our evaluation protocol, with Qwen3-4B as the draft and Qwen3-32B as the assistant model. For a controlled comparison, the decision tree is trained on the same OpenR1-Math mismatch traces used for our warm-up stage. As shown in Table 10, CoSD improves slightly over target-only decoding, whereas CoSpec obtains both substantially larger task gains and higher acceleration. The gap is not merely between a hand-designed rule and a learned verifier: the CoSD decision tree is itself learned from draft and assistant confidence features. CoSpec differs in conditioning arbitration on the full speculative round and in optimizing its decisions for downstream task outcomes.

Table 10: Comparison with CoSD under the same Qwen3-32B/4B pair and evaluation protocol. Speed is relative to target-only autoregressive decoding.

Method	GSM8K		HumanEval	
	Speed	Score	Speed	Score
Target only	1.00×	85.96	1.00×	89.63
CoSD (decision tree)	1.49×	86.35	1.55×	89.75
CoSpec	2.03×	88.15	2.15×	92.07

AutoJudge. Judge Decoding does not release its training data or implementation, so we use AutoJudge as a reproducible representative of learned verification. We run the official AutoJudge release in our evaluation harness with the LLaMA-3.1-8B/3.3-70B pair on GSM8K, and measure all speedups against target-only autoregressive decoding. AutoJudge reaches a 3.75× speedup and CoSpec a 3.82× speedup. At this comparable speed, the two methods move the task score in opposite directions relative to target-only decoding: AutoJudge falls below the target score (94.77 vs. 95.60), whereas CoSpec exceeds it (96.59; Table 1). This difference is consistent with the supervision each method uses. AutoJudge learns from rollout-derived labels whether a mismatched draft token can be retained without changing the target's answer, so its supervision remains target-referenced. CoSpec instead optimizes arbitration for final task utility and may deliberately select a draft branch that improves over the target outcome. The judge-supervised SFT-only arbitrator in Table 6, an acceptability-supervision control rather than a reimplementation of AutoJudge, shows the same pattern: it stays below target-only decoding on HumanEval (86.93 vs. 89.63), whereas full rollout-level RL raises the score to 92.07.

F Where Should the Additional Supervision Go?

CoSpec uses an extra trained model and a utility signal from final answers. To test whether its gains follow from this extra supervision or compute alone, we compare several ways of spending it on the Qwen3-32B/4B pair (Tables 5, 6 and 11):

- **In the target's weights.** RL on the target with the same math data improves GSM8K (85.96 to 88.23) but degrades HumanEval (89.63 to 86.57), and provides no acceleration on its own; with SpS it reaches 1.38× and 1.23×.

- **In a larger draft.** An 8B draft slows SpS (1.22× and 1.09× versus 1.41× and 1.29× with the 4B draft) and, as SpS preserves the target distribution, leaves the score unchanged.
- **In a judge-supervised arbitrator.** SFT on judge labels alone reaches 1.71× but stays below the target on HumanEval (86.93).
- **At the response level.** Reranking complete responses improves quality but is slower than target-only decoding (Table 11).
- **In speculative arbitration (CoSpec).** Full-rollout RL of the arbitrator improves both quality and speed.

Response-level reranking. Rerank-N samples N complete responses from the target model at temperature 1 and selects one of them with the Qwen3-30B-A3B-Instruct-2507 judge used for our warm-up labels. Table 11 shows that this selection improves quality: Rerank-4 even exceeds CoSpec on GSM8K (88.70 versus 88.15), although CoSpec remains higher on HumanEval (92.07 versus 91.46). However, reranking requires generating several full responses and is slower than target-only decoding, at 0.92× for Rerank-2 and 0.26× for Rerank-4. Response-level selection thus trades acceleration for quality, whereas CoSpec exploits draft–target complementarity inside the speculative loop and obtains both.

Table 11: Response-level reranking versus CoSpec (Qwen3-32B/4B). Speed is relative to target-only decoding and averaged over GSM8K and HumanEval.

Method	Speed	GSM8K	HumanEval
Target only	1.00×	85.96	89.63
Rerank-2	0.92×	87.41	90.24
Rerank-4	0.26×	88.70	91.46
CoSpec	2.03–2.15×	88.15	92.07

G Statistical Significance of Accuracy Gains

To test whether the above-target gains can be explained by instance-level variation, we compare CoSpec and target-only decoding on the same instances and apply two-sided exact McNemar tests to the discordant pairs. Table 12 reports, for each benchmark, the number of instances solved only by target-only decoding and only by CoSpec. The GSM8K, MBPP, and HumanEval counts correspond to the LLaMA-3.3-70B/8B greedy results in Tables 1 and 2; MATH uses the full 5,000-problem test set with the same LLaMA-3.3-70B/8B pair and greedy decoding. We report unadjusted p-values together with Holm-adjusted values across the four tests. On full MATH, CoSpec solves 138 problems missed by target-only decoding while losing 68, a net gain of 70 that remains highly significant after adjustment. GSM8K and MBPP are nominally significant but not after Holm adjustment, and the small HumanEval gain is not significant. We therefore do not claim that every per-benchmark improvement is significant; rather, the large-scale MATH evaluation shows that the above-target improvement is not explained by chance variation on small test sets.

Table 12: Paired comparison of target-only decoding and CoSpec with two-sided exact McNemar tests.

Benchmark	Total	Target only	CoSpec only	Net gain	p (unadjusted)	p (Holm)
GSM8K	1319	7	20	+13	0.019	0.058
MBPP	499	5	16	+11	0.027	0.058
HumanEval	164	1	4	+3	0.375	0.375
MATH (full)	5000	68	138	+70	1.2×10^{-6}	4.9×10^{-6}

H When Does Arbitration Fail?

Failure modes. We examine arbitration errors on GSM8K and MATH-500 for the LLaMA-3.3-70B/8B pair under the default setting, using the counterfactual procedure of Section 4.3; the false

accepts were categorized manually. A *false accept* keeps a mismatched draft token whose branch leads to a worse final outcome than the target branch; a *false reject* falls back to the target when the draft branch would have been better. False rejects follow the target branch, so they forgo potential accuracy and speed gains rather than introducing errors that the target branch would avoid. We therefore categorize the false accepts (Table 13). Of the 111 false accepts, 91 (82.0%) fall into two recurring classes: locally plausible numerical substitutions (57.7%), in which the draft replaces a number or intermediate value with a plausible but incorrect one, and premature shortcuts (24.3%), in which the draft skips a reasoning step whose omission becomes harmful only later. Both classes involve decisions whose downstream consequences are difficult to infer from the current round, which suggests that longer-range lookahead is a natural direction for improving the arbitrator.

Table 13: Arbitration errors on GSM8K and MATH-500, with a breakdown of false accepts.

			False accepts by type		
Benchmark	False accept	False reject	Numeric substitution	Premature shortcut	Other
GSM8K	47	29	29 (61.7%)	10 (21.3%)	8 (17.0%)
MATH-500	64	38	35 (54.7%)	17 (26.6%)	12 (18.8%)
Total	111	67	64 (57.7%)	27 (24.3%)	20 (18.0%)

Refusal and instruction following. Because CoSpec can override the target at mismatches, we also check whether it changes the target's refusal and instruction-following behavior. We evaluate the LLaMA-3.3-70B/8B pair with greedy decoding and report the average of prompt- and instruction-level strict accuracy on IFEval. In these evaluated settings, CoSpec closely preserves both: the AdvBench refusal rate changes from 99.4% to 99.2%, and the IFEval score from 88.9 to 88.8 (Table 14). These experiments are behavioral checks rather than a formal safety guarantee.

Table 14: Refusal and instruction-following checks.

Evaluation	Target only	CoSpec
AdvBench refusal rate (%)	99.4	99.2
IFEval	88.9	88.8

I Evaluation Protocol

Prompting and stopping. For GSM8K, MGSM, and MATH-500, we use the same chain of thought prompt template for all methods. We parse the final answer with the same normalizer. For HumanEval and MBPP, we use the official unit test harness and stop at the first complete code block or the maximum generation length. For MT-Bench, the same judge model and judge prompt are used for all decoding methods.

Validation only hyperparameter selection. The default $K = 25$ and $\lambda = 0.6$ are selected on held-out validation prompts before test evaluation. We do not tune λ on test sets. The threshold ablation in Table 7 is reported only to show the controllable tradeoff.

J Inference Procedure

For completeness, the CoSpec inference loop is as follows. Starting from prompt x and verified prefix $y^{(<r)}$, the draft proposes K tokens. The target performs one teacher forced pass over the draft block and produces K verification tokens plus one bonus token. CoSpec forms the arbitrator input in Equation (9) and applies the hybrid attention mask. Matched draft tokens are forced accepted. At each mismatch, the arbitrator accepts the draft token if the policy probability exceeds λ. The round stops at the first rejected mismatch and appends the target token at that position. If no mismatch is rejected, all K draft tokens and the target bonus token are appended. The procedure repeats until EOS or the maximum length.

J.1 Serving Throughput and Overhead

Longer accepted spans translate into practical gains only when arbitrator overhead is smaller than the target forward passes avoided. Tables 16 to 18 report fixed batch throughput, continuous batching throughput, and serving cost. CoSpec improves throughput over target-only decoding across batch sizes and request rates. Table 15 breaks down the per-round latency in the default LLaMA-3.3-70B/8B setting on GSM8K at batch size 1. The arbitrator forward pass takes 19 of 295 ms, roughly 6% of per-round latency. This sequential forward introduces measurable overhead, which the longer accepted spans more than compensate for in the evaluated low-to-moderate concurrency regime. Under higher concurrency the benefit shrinks: with continuous batching, the speedup decreases from 2.18× at 2 requests per second to 1.22× at 16 (Table 17). The 2–4× speedups in Table 1 should therefore be read as applying to our benchmark setting rather than to arbitrary serving loads.

Table 15: Per-round latency breakdown (LLaMA-3.3-70B/8B, GSM8K, batch size 1).

Component	Latency (ms)
Draft proposal	212
Target verification	58
Arbitrator	19
Sampling and bookkeeping	6
Total	295

Table 16: vLLM fixed batch throughput. Absolute throughput is shown in tokens per second with speedup in parentheses.

Batch size	2	4	8	16	32	64
Target only	43.1	79.9	101.5	183.4	316.8	522.5
CoSpec	97.8 / 2.3×	162.8 / 2.0×	277.2 / 2.7×	405.9 / 2.2×	506.9 / 1.6×	627.0 / 1.2×

Table 17: vLLM continuous batching throughput under different request rates.

Request rate	Target only	CoSpec	Speedup
2 req/s	92.4	201.7	2.18×
4 req/s	168.3	302.6	1.80×
8 req/s	251.9	361.5	1.44×
12 req/s	318.5	421.3	1.32×
16 req/s	392.1	478.6	1.22×

Table 18: Training and inference overhead.

Item	Cost or description
CoSpec training pipeline	∼400 MI325X GPU hours
Target only RL baseline	∼240 MI325X GPU hours
RL rollouts	12 rollouts per prompt
Prompts per update	128 prompts
Inference overhead	one arbitrator forward per speculative round
Target only serving memory	142 GB
SpS serving memory	158 GB
EAGLE-3 serving memory	150 GB
CoSpec serving memory	174 GB
Time to first token overhead	+4.8% vs. target only

K LIMITATIONS

Objective. CoSpec optimizes final task utility rather than exact target distribution preservation at selected mismatch states. This is the intended operating point for tasks where the evaluation metric is answer correctness, code pass rate, factual accuracy, or judge score. It is not the right choice when downstream use requires samples from the exact target distribution or target-faithful log likelihoods. CoSpec also chooses between two local candidates, so it cannot repair states where both draft and target candidates lead to the same failure.

Scope of the evaluation. We evaluate two same-family pairs with shared tokenizers, LLaMA-3.3-70B/3.1-8B and Qwen3-32B/4B, which cover a limited range of draft–target size ratios; cross-family pairs, pairs with different tokenizers, and other size ratios remain untested. Long-context generation has not been evaluated systematically. RL training requires a verifiable final-outcome signal; we train only on math, and although the arbitrator transfers to other tasks (Table 4), training on tasks without automatic correctness checks would require a learned or judged reward.

Diagnostics. The counterfactual analysis (Table 3) is on-policy: it covers the mismatch states that CoSpec actually reaches and says nothing about states it never visits. The refusal and instruction-following results (Appendix H) are behavioral checks on two benchmarks, not a formal safety guarantee.

Cost. Training the arbitrator takes roughly 400 GPU hours (Table 18). At inference time, the arbitrator is a separate model instance: it increases serving memory (174 GB versus 158 GB for SpS in our setting, Table 18) and adds a sequential forward pass per round, so its relative speedup decreases under high-concurrency serving (Appendix J.1).

L LICENSE OF ASSETS USED

Table 19: Licenses of the models and datasets used in this work.

Asset	Type	License
Llama-3.3-70B-Instruct	Model	Llama 3.3 Community License
Llama-3.1-8B-Instruct	Model	Llama 3.1 Community License
Qwen3-32B	Model	Apache 2.0
Qwen3-4B	Model	Apache 2.0
Qwen3-30B-A3B-Instruct-2507	Model	Apache 2.0
OpenR1-Math-220k	Dataset	Apache 2.0
GSM8K	Dataset	MIT
HumanEval	Dataset	MIT
MBPP	Dataset	CC-BY-4.0
MATH-500 (from MATH)	Dataset	MIT
MGSM	Dataset	CC-BY-4.0
TriviaQA	Dataset	Apache 2.0
GPQA	Dataset	CC-BY-4.0 (gated)
MT-Bench	Dataset	CC-BY-4.0

All assets are used in accordance with their respective licenses for non-commercial research evaluation. GPQA is access-gated with a request to not publish examples online, we did not include verbatim GPQA questions in this paper.

M BROADER IMPACT

This work studies inference algorithms for large language models. Its main positive impact is improved serving efficiency: by increasing useful accepted spans while maintaining or improving task utility, CoSpec may reduce latency, hardware cost, and energy consumption for LLM deployment. This can make stronger models more accessible under constrained compute budgets. Faster reasoning may reduce the cost of generating harmful code on a large scale. Deployment should retain external safety filters or safety-aware arbitration, especially when draft and target safety behavior differs.