PC-VOT: PREFIX-COUPLED VERIFIER-INDUCED OPTIMAL TRANSPORT FOR DRAFT TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative decoding accelerates LLM inference by using a lightweight draft model to propose candidate tokens that are verified in parallel by the target model. Diffusion drafters further improve drafting efficiency by generating an entire candidate block in a single forward pass. However, a fundamental training-decoding discrepancy remains: conventional objectives train the drafter to match the target distribution pointwise, while decoding depends on the verifier-induced decision regions that determine which draft proposals are accepted. We propose *Prefix-Coupled Verifier-Induced Optimal Transport (PC-VOT)*, a verifier-aligned training framework for diffusion drafters. PC-VOT constructs a verifier-induced feasible set around the Bayes-optimal proposal under stochastic verification and minimizes the point-to-set transport required to reach it. It then couples the transport cost across positions according to prefix reachability and the remaining speculative progress at risk. Theoretical analysis validates that PC-VOT provides a lower bound on expected acceptance length. Extensive experiments across seven math, coding, and dialogue benchmarks show that PC-VOT improves the average decoding speedup by up to 11.2% over DFlash baseline and 6.0% over D-PACE baseline.

1 INTRODUCTION

Large language models (LLMs) (Grattafiori et al., 2024) perform autoregressive decoding sequentially, requiring a model forward pass for each new token. Speculative decoding (SD) alleviates this bottleneck by using a lightweight *drafter* to propose multiple draft tokens that are verified in parallel by the target model (Leviathan et al., 2023; Stern et al., 2018; Chen et al., 2023; Sun et al., 2023). DFlash (Chen et al., 2026) further reduces drafting latency with a lightweight diffusion model that predicts an entire draft block in parallel. Since diffusion drafts tend to exhibit flatter and higher-entropy token distributions than the sharper autoregressive target, DFlash adopts *top-1 draft verification*, which compares only the draft's most likely token rather than requiring full-distribution agreement. Specifically, to preserve lossless verification, a draft token is accepted if its top-1 prediction matches the token sampled from the target distribution; otherwise, we accept the sampled target token and discard the remaining draft suffix. However, fast drafting alone is insufficient, since decoding efficiency depends on both *which decisions the drafter makes* and *where the first mismatch occurs*.

Current training objectives for SD draft model are over-specified for the verifier decision, and their surrogate ordering can be misaligned away from the optimum. Let q_i and p_i denote the draft and target distributions at position i, and let $t_i \sim p_i$ be the realized verifier token. For this training instance, hard-label cross-entropy (CE), as used by DFlash to train its draft model, pushes q_i toward the one-hot distribution δ_{t_i} induced from p_i, while distillation objectives like Kullback–Leibler (KL) divergence and total variation (TV) seek to match the full target distribution p_i. Verification needs neither endpoint. Instead, conditioned on t_i, every distribution q_i in $\mathcal{A}(t_i) = \{q : \arg\max_v q(v) = t_i\}$ satisfies the same local acceptance condition. From $\mathcal{A}(t_i)$, one can observe that the verifier specifies which token should be preferred, but leaves the remaining probabilities unspecified. Point-target matching therefore imposes distributional requirements beyond those needed for that decision.

More importantly, point-matching losses can rank draft distributions in the opposite order from verification. Consider a target distribution p and two draft distributions q^{rej} and q^{acc} on three tokens:

$$p = (0.40, 0.35, 0.25), \quad q^{\mathrm{rej}} = (0.37, 0.38, 0.25), \quad q^{\mathrm{acc}} = (0.36, 0.32, 0.32). \tag{1}$$

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.

The top-1 proposals of q^{rej} and q^{acc} are tokens 2 and 1, so their expected acceptance probabilities are 0.35 and 0.40, respectively. Nevertheless, when token 1 is realized, q^{rej} has lower hard-label CE than q^{acc} (0.994 vs. 1.022). It is also closer to p under TV (0.03 vs. 0.07) and forward KL (0.002 vs. 0.012). These objectives favor q^{rej} because they measure proximity to a prescribed distributional endpoint, even though it induces both the wrong greedy decision and a lower expected acceptance probability. Thus, a lower point-target matching loss does not necessarily correspond to either a verifier-aligned prediction or a higher expected acceptance probability. This suggests that draft training should directly optimize the decision induced by stochastic verification. Because a deterministic drafter cannot match every stochastic verifier outcome, our objective is therefore to maximize acceptance in expectation. The expected local acceptance is maximized by the *Bayes-optimal proposal* $t^\star \in \arg\max_{v\in\mathcal{V}} p(v)$. It, in turn, induces the verifier feasible set $\mathcal{A}(t^\star)$. Every distribution in the set achieves the same maximum expected local acceptance, regardless of how its remaining probability mass is allocated. For a capacity-limited drafter, this set-valued optimality motivates enforcing only the verification-relevant ranking constraint rather than reproducing a prescribed distributional endpoint.

The local mismatch becomes more consequential at the block level. Even with a verifier-aligned local objective, optimizing positions independently ignores how each prediction contributes to the accepted prefix. Verification follows a first-failure process: a mismatch at position i discards that token and the entire remaining suffix. For example, two eight-token blocks with one mismatch each have the same token-level accuracy, yet a mismatch at position 2 leaves only one accepted draft token, whereas a mismatch at position 7 leaves six. Moreover, a later prediction cannot contribute to the current accepted prefix if an earlier mismatch has already occurred. Thus, beyond aligning each local prediction with its verifier-induced acceptance set, training should also account for *whether a position is reached* and *how much speculative progress is at risk* if it fails. Together, these observations motivate a verifier-aligned objective that first optimizes each draft prediction toward an acceptance set and then couples these local objectives according to prefix-dependent decoding utility.

Contributions. Motivated by these two misalignments, we propose *Prefix-Coupled Verifier-Induced Optimal Transport* (PC-VOT), a training-only objective that aligns diffusion drafting with top-1 verification locally and at the block level. Our key contributions are highlighted below.

First, we introduce *point-to-set optimal transport* to align draft training with stochastic top-1 verification. Since a deterministic draft proposal cannot simultaneously agree with all possible verifier samples, we characterize the Bayes-optimal proposal that maximizes the expected local acceptance probability under the target distribution. We then construct a margin-constrained feasible set based on the Bayes-optimal proposal, and measure the minimum probability-mass transport required for the current draft prediction to reach this set. The optimization leaves verifier-irrelevant probability assignments unconstrained. Thus, PC-VOT does not merely replace the discrepancy to the same target; it replaces a prescribed distributional endpoint with verifier-induced set-valued supervision.

Second, we develop *prefix coupling* to resolve the mismatch between position-wise local optimization and the verifier's first-failure utility. PC-VOT weights each point-to-set transport cost by the probability that verification reaches that position and the remaining speculative suffix at risk upon failure. The resulting objective accounts jointly for prefix reachability and the remaining suffix length: earlier positions carry greater potential consequences, while their contribution also depends on the acceptance probabilities of preceding proposals. These prediction-dependent weights connect local supervision to its relevance for accepted-prefix preservation based on verification behavior.

Third, we establish a direct theoretical connection between PC-VOT and expected accepted length. We show that PC-VOT provides a lower bound on expected accepted length relative to the expected accepted length of the target-dependent greedy proposal. This result directly connects verifier-aligned training to decoding-time acceptance and speculative-decoding efficiency.

Finally, we evaluate PC-VOT across seven math, coding, and dialogue tasks with Qwen3 and Gemma target models. Compared with DFlash, PC-VOT improves average acceptance length by 16.7% and wall-clock decoding speedup by 16.2% under the same drafter architecture and verification procedure. These gains arise solely from the PC-VOT objective without additional inference-time components.

2 Related Work

Speculative decoding for autoregressive models. Speculative decoding amortizes target-model computation by proposing multiple candidate tokens for parallel verification (Leviathan et al., 2023;

Stern et al., 2018; Chen et al., 2023; Sun et al., 2023; Miao et al., 2024). Candidate generation ranges from auxiliary prediction heads (Cai et al., 2024) and self-speculative early exiting (Hooper et al., 2025; Elhoushi et al., 2024; Liu et al., 2024; Xia et al., 2024) to n-gram and retrieval-based continuations (Ou et al., 2024; Liu et al., 2025; He et al., 2024; Zhao et al., 2024). Jacobi and Lookahead decoding instead cast generation as parallel iterative refinement (Santilli et al., 2023; Teng et al., 2025; Fu et al., 2024). Multi-draft verification (Yang et al., 2025b; Hu et al., 2025; Samragh et al., 2025; Zhou et al., 2026; Sun et al., 2024) and tree-based methods (Li et al., 2024a;b; Chen et al., 2024; Wang & Zhou, 2026) expand or organize the candidate space to improve accepted length. Some variants relax exact target-distribution preservation to increase throughput (Qin et al., 2024; Zhou et al., 2024; Cai et al., 2024; Narasimhan et al., 2025; Fu et al., 2025).

Diffusion-based drafting. DiffuSpec and the SpecDiff family (Li et al., 2026a; Christopher et al., 2025; Sandler et al., 2026) use pretrained diffusion language models to generate draft blocks in parallel, although large drafters can introduce substantial memory and latency overhead. DFlash (Chen et al., 2026) instead uses a lightweight diffusion drafter for block-parallel prediction. Subsequent work explores both candidate organization and dependencies within the draft block. DDTree (Ringel & Romano, 2026) and JetSpec (Hu et al., 2026a) extend parallel drafting with tree-structured candidates or verification, while Domino (Huang et al., 2026) and DSpark (Cheng et al., 2026) strengthen dependencies among tokens within the block.

Draft training and verifier alignment. Knowledge distillation and verifier-aware objectives seek to align draft proposals with target-model behavior (Zhou et al., 2024; Hu et al., 2026b; Zhang et al., 2025; Li et al., 2026b; Zou et al., 2026). Within diffusion drafting, JetSpec (Hu et al., 2026a) and Draft-OPD (Lei et al., 2026) improve training supervision, while D-PACE (Wu et al., 2026) develops position-aware cross-entropy. PC-VOT revisits the supervision objective rather than only the weighting of an existing loss. It selects the nearest margin-valid distribution from a verifier-induced acceptance set, then couples the resulting local repair costs through the first-failure structure of verification. This distinguishes *which distributions are sufficient for acceptance* from *where training emphasis should be placed*, and connects both within a single objective.

3 Methodology

3.1 Problem Setup and Overview

Given a context c, a diffusion drafter Q_ϕ predicts a block of n tokens in parallel, which a frozen autoregressive target model P verifies from left to right (Christopher et al., 2025; Chen et al., 2026). Let $p_i(\cdot \mid c), q_{\phi,i}(\cdot \mid c) \in \Delta^{|\mathcal{V}|-1}$ denote the target and draft token distributions at position i, respectively, where $\mathcal{V}$ is the vocabulary. When no ambiguity arises, we suppress dependence on c and ϕ, writing them simply as p_i and q_i. Let $t_i \sim p_i$ be the token sampled by the target model. The drafter proposes its top-1 token, and position i is accepted when $A_i = 1$, where

$$A_i = \mathbb{1}\{\arg\max_v q_i(v) = t_i\}.$$

Verification terminates at the first mismatch. Thus, A_i records whether position i satisfies the local top-1 decision, and the accepted draft length is

$$L_{\text{acc}} = \sum_{i=1}^{n} \prod_{j=1}^{i} A_j. \tag{2}$$

Here, $\prod_{j=1}^{i} A_j$ indicates whether the first i positions are all accepted. If the first mismatch occurs at position k, the product equals one for $i < k$ and zero for all $i \geq k$, yielding $L_{\text{acc}} = k - 1$.

Eq. (2) directly exposes the two mismatches identified in Sec. 1. *Locally*, each prediction matters through whether it satisfies the verifier decision A_i, not whether it reproduces a particular probability distribution. *At the block level*, these decisions are coupled through prefix survival: a position contributes only if all previous positions survive, while an early failure discards a larger suffix. Hence, verifier-aligned training should capture both *local decision validity* and *prefix-dependent utility*.

PC-VOT addresses these two requirements. *Point-to-set transport* measures the minimum probability-mass transport needed to reach a verifier-induced region (Sec. 3.2), while *prefix coupling* weights each local repair by its expected prefix reachability and the remaining speculative suffix at risk (Sec. 3.3). Together, they determine *what each position should learn* and *how much its correction should matter*

to the accepted prefix. Finally, we theoretically connect the training objective to decoding-time acceptance by deriving a lower bound on expected accepted length relative to the expected accepted length of the target-dependent greedy proposal (Sec. 3.4).

3.2 What Each Position Should Learn: Point-to-Set Transport

To align training supervision with the verifier, we first identify the deterministic proposal that maximizes expected local acceptance. We then construct a verifier-induced feasible set around this decision and define the minimum transport required to reach it.

From pointwise target matching to verifier-induced set matching. Given a target token $t_i \sim p_i$ and its corresponding one-hot distribution δ_{t_i}, conventional draft objectives prescribe a single distribution that q_i should reproduce (Zhou et al., 2024; Zhang et al., 2025; Wu et al., 2026):

$$\text{CE}: \; q_i \to \delta_{t_i}, \qquad \text{KL/TV}: \; q_i \to p_i. \tag{3}$$

Although using different discrepancy measures, these objectives share the same pointwise matching formulation: CE selects the one-hot endpoint δ_{t_i}, whereas KL/TV select the full target distribution p_i.

This pointwise target matching view is stronger than what top-1 draft verification actually requires. Conditioned on the realized target token t_i, the draft is accepted whenever $\arg\max_v q_i(v) = t_i$. So the verifier specifies an entire region $\mathcal{A}_i(t_i)$ of valid draft distributions rather than a unique distributional target:

$$\mathcal{A}_i(t_i) = \left\{ q_i \in \Delta^{|\mathcal{V}|-1} : \arg\max_v q_i(v) = t_i \right\}. \tag{4}$$

Pointwise matching imposes a stronger constraint than Eq. (4) and continues penalizing distributional differences even when they do not alter the verifier decision. This motivates replacing point-to-point matching with a training target based on the verifier.

Verifier-induced feasible set. The top-1 condition in Eq. (4) characterizes the verifier's acceptance region conditioned on a particular target sample $t_i \sim p_i$. However, because t_i is sampled from target the model and is stochastic, different samples from the same target distribution may induce conflicting acceptance regions. A deterministic drafter cannot simultaneously satisfy these realized decisions. Its appropriate objective is therefore to select the deterministic proposal that maximizes acceptance *in expectation* over the stochastic verifier.

Lemma 1 (Bayes-optimal verifier decision)**.** *Let q_i be the draft distribution at position i, and let $t_i \sim p_i$ be the token sampled by the stochastic target verifier. Under a fixed tie-breaking rule, the expected local acceptance probability is*

$$\alpha_i(q_i) = \mathbb{E}_{t_i \sim p_i}[\mathbb{1}\{\arg\max_{v\in\mathcal{V}} q_i(v) = t_i\}] = p_i(\arg\max_{v\in\mathcal{V}} q_i(v)). \tag{5}$$

Hence, a Bayes-optimal deterministic proposal is

$$t_i^\star \in \arg\max_{v\in\mathcal{V}} p_i(v), \qquad \alpha_i^\star = p_i(t_i^\star) = \max_{v\in\mathcal{V}} p_i(v), \tag{6}$$

and the corresponding Bayes-optimal decision region is

$$\mathcal{A}_i(t_i^\star) = \left\{ q_i \in \Delta^{|\mathcal{V}|-1} : \arg\max_{v\in\mathcal{V}} q_i(v) = t_i^\star \right\}. \tag{7}$$

Every $q_i \in \mathcal{A}_i(t_i^\star)$ attains the same maximal expected local acceptance $\alpha_i^\star$.

See its proof in Appendix C.1. Lemma 1 shows that expected acceptance depends only on the draft's top-1 decision: once q_i selects a token, its acceptance probability is simply the probability that p_i assigns to that token. Hence, the optimal deterministic proposal is the target's most probable token $t_i^\star$. More importantly, this optimum is *set-valued*: every $q_i \in \mathcal{A}_i(t_i^\star)$ induces the same Bayes-optimal proposal and attains the same maximal acceptance $\alpha_i^\star$, regardless of how its remaining probability mass is distributed. Thus, achieving maximal acceptance under stochastic verification requires q_i to lie in a Bayes-optimal *decision region*, rather than match a unique distributional target.

However, $\mathcal{A}_i(t_i^\star)$ alone does not ensure a stable top-1 proposal, since it also contains distributions arbitrarily close to the top-1 decision boundary. We therefore require the draft distribution to preserve a fraction β of the corresponding target margin and define the verifier-induced feasible set

$$\mathcal{A}_i^\beta(t_i^\star) = \left\{ \tilde{q}_i \in \Delta^{|\mathcal{V}|-1} : \tilde{q}_i(t_i^\star) - \tilde{q}_i(v) \geq \beta\,[p_i(t_i^\star) - p_i(v)], \;\; \forall v \neq t_i^\star \right\}, \qquad 0 < \beta \leq 1. \tag{8}$$

By construction, $\mathcal{A}_i^\beta(t_i^\star) \subseteq \mathcal{A}_i(t_i^\star)$. The target-adaptive margin scales with the verifier's confidence: it requires greater separation when p_i strongly favors $t_i^\star$ and contracts for nearly tied target decisions. The coefficient β specifies the fraction of these pairwise target margins that the draft distribution must preserve. Consequently, the feasible set avoids unstable decision boundaries while leaving other probability assignments unconstrained, without forcing the drafter toward either the one-hot distribution $\delta_{t_i^\star}$ or the full target distribution p_i.

Verifier-induced point-to-set optimal transport. The feasible set $\mathcal{A}_i^\beta(t_i^\star)$ specifies *which* draft distributions are verifier-aligned, but not *how much* the current prediction must change to reach one of them. We formulate this minimum decision repair through optimal transport.

For any two distributions q_i and $\tilde{q}_i$, we define their transport discrepancy as

$$W_{C_i}(q_i, \tilde{q}_i) = \min_{\pi \in \Pi(q_i, \tilde{q}_i)} C_i(\pi), \tag{9}$$

where $\Pi(q_i, \tilde{q}_i)$ denotes the set of transport plans with marginals q_i and $\tilde{q}_i$, and $C_i(\pi)$ specifies the cost associated with a transport plan π (Peyré & Cuturi, 2020). To quantify the probability mass reallocation across discrete token decisions, we adopt the standard linear Kantorovich transport cost

$$C_i(\pi) = \sum\nolimits_{a,b \in \mathcal{V}} \pi(a,b) c_i(a,b), \tag{10}$$

where $c_i(a,b)$ denotes the ground cost of transporting one unit of probability mass from token a to token b. We set $c_i(a,b) = \mathbb{1}\{a \neq b\}$ in this work, since it is well aligned with lossless speculative decoding, whose verifier operates on discrete token outcomes rather than semantic similarity: transport incurs a cost only when probability mass is reassigned across distinct token decisions.

Based on the verifier-induced feasible set defined in Eq. (8), we can define the verifier-induced point-to-set optimal-transport cost as

$$D_i^\beta(q_i; t_i^\star) = \min_{\tilde{q}_i \in \mathcal{A}_i^\beta(t_i^\star)} W_{C_i}(q_i, \tilde{q}_i). \tag{11}$$

Eq. (11) minimizes transport cost by jointly optimizing the transport endpoint and the probability-mass coupling. For each candidate endpoint $\tilde{q}_i \in \mathcal{A}_i^\beta(t_i^\star)$, the inner optimal-transport problem finds the least-cost coupling between q_i and $\tilde{q}_i$. The outer minimization then selects the feasible endpoint with the lowest such cost. Thus, D_i^β measures the minimum probability mass that must be reassigned to bring q_i into the verifier-induced feasible set, rather than the cost of matching the distribution p_i.

Lemma 2 (Feasibility and point-to-set relaxation)**.** *For any $0 < \beta \leq 1$, the point-to-set transport cost satisfies*

$$D_i^\beta(q_i; t_i^\star) = 0 \quad \Longleftrightarrow \quad q_i \in \mathcal{A}_i^\beta(t_i^\star). \tag{12}$$

Moreover,

$$D_i^\beta(q_i; t_i^\star) \leq \min\left\{W_{C_i}(q_i, p_i), W_{C_i}(q_i, \delta_{t_i^\star})\right\}. \tag{13}$$

See its proof in Appendix C.2. Lemma 2 characterizes PC-VOT as a set-valued relaxation of pointwise target supervision. Eq. (12) shows that the objective searches for the nearest verifier-induced feasible endpoint and stops penalizing q_i once the required decision margin is satisfied. Eq. (13) shows that reaching the nearest feasible distribution is no more costly than the pointwise target supervision. The objective does not force q_i to reproduce verifier-irrelevant probability details within the feasible region. This property is useful for a limited-capacity diffusion drafter: by concentrating updates on unresolved decision errors, the available model capacity can be devoted to capturing the verifier-induced decision region across a broader range of training contexts, rather than reproducing unnecessary distributional details at already-correct positions.

3.3 How Positions Should Contribute: Prefix Coupling

The point-to-set transport in Sec. 3.2 determines *what* each position should learn by measuring the minimum repair required to reach the verifier-induced feasible region. However, speculative decoding does not value all local repairs equally. Verification proceeds from left to right and terminates at the first mismatch: a local error matters only if verification reaches that position, and an earlier error can invalidate a longer speculative suffix (Zou et al., 2026; Wu et al., 2026). We therefore couple each local transport cost with its expected contribution to the accepted prefix.

Prefix-dependent utility. Recall from Eq. (5) that the deterministic proposal $\arg\max_v q_i(v)$ is accepted with probability

$$\alpha_i(q_i) = p_i\left(\arg\max_v q_i(v)\right). \tag{14}$$

Because verification proceeds sequentially and terminates at the first mismatch, position i is reached only if all preceding proposals are accepted. We therefore define its prefix reachability as

$$R_{i-1} = \prod_{j<i} \alpha_j(q_j), \qquad R_0 = 1. \tag{15}$$

The consequence of a mismatch also depends on its position. A mismatch at position i prevents the current proposal and its remaining suffix from being accepted, resulting in an accepted-length shortfall of $n - i + 1$ relative to full-block acceptance. We refer to this quantity as the *remaining-horizon factor*. Combining it with prefix-reachability at position i, we define the prefix-dependent weight as

$$w_i = \underbrace{(n-i+1)}_{\text{remaining-horizon factor}} \cdot \underbrace{R_{i-1}}_{\text{prefix-reachability factor}}. \tag{16}$$

The remaining-horizon factor emphasizes earlier positions because their failure wastes more speculative progress, while the prefix-reachability factor modulates this emphasis according to the drafter's current acceptance behavior. In particular, a later position receives less weight when its preceding prefix is unlikely to survive. As the preceding proposals improve, the position becomes more reachable, increasing the utility of repairing its local proposal during training.

Training objective. Combining verifier-induced point-to-set cost with prefix-coupling utility yields

$$\min_\phi \mathcal{L}_{\text{PC-VOT}}(\phi) = \mathbb{E}\left[\sum\nolimits_{i=1}^{n} w_i D_i^\beta(q_{\phi,i}; t_i^\star)\right]. \tag{17}$$

PC-VOT therefore prioritizes a local decision repair when it is both likely to affect the surviving prefix and capable of preserving a longer speculative suffix. Thus, point-to-set transport determines *what* each position should learn, while prefix coupling determines *how much* that local correction should contribute to the block-level objective. The complete training algorithm is provided in Appendix B.

3.4 Theoretical Analysis

We establish a connection between the PC-VOT objective and the expected accepted draft length.

Connection to expected accepted length. Let $\mathbb{E}[L_{\text{acc}}]$ denote the expected accepted draft length defined in Eq. (2), where a larger value generally indicates greater decoding speedup. Given a block of size n, we define $n^\star$ as a drafter-independent reference, the expected accepted length of the target's greedy proposal under top-1 draft verification, and state the PC-VOT guarantee as follows.

Theorem 3 (Expected accepted-length guarantee)**.** *Fix a frozen target model P, an arbitrary context c, and a draft length n. Define $\gamma := \min_{1\le j\le n}\left[p_j(t_j^\star \mid c, t_{<j}^\star) - \max_{v\in\mathcal{V}\setminus\{t_j^\star\}} p_j(v \mid c, t_{<j}^\star)\right] > 0$. Then, for any $0 < \beta \le 1$ and any drafter Q_ϕ, we have*

$$n^\star := \mathbb{E}\left[\sum\nolimits_{k=1}^{n}\prod\nolimits_{j=1}^{k} \max_{v\in\mathcal{V}} p_j(v|c, t_{<j}^\star)\right], \qquad n^\star - \frac{2}{\beta\gamma}\mathcal{L}_{\text{PC-VOT}}(\phi) \le \mathbb{E}[L_{\text{acc}}]. \tag{18}$$

The proof is provided in Appendix C.3. Theorem 3 uses $n^\star$ as a reference for the expected accepted length attained by any drafter model Q_ϕ under top-1 verification. It captures the expected accepted length of the target's greedy proposal under the stochastic target verifier and depends only on the target model, making it drafter-independent. The theorem establishes, for any drafter model Q_ϕ, a relationship between the PC-VOT objective and expected accepted length: reducing $\mathcal{L}_{\text{PC-VOT}}(\phi)$ raises the theoretical lower bound on $\mathbb{E}[L_{\text{acc}}]$ toward $n^\star$. Since speculative-decoding efficiency depends on the number of consecutively accepted draft tokens, this guarantee directly connects the training objective to decoding-time acceptance.

4 Experiments

Models & Tasks. We evaluate PC-VOT on Qwen3-4B (Q3-4B), Qwen3-8B (Q3-8B) (Yang et al., 2025a), and Gemma4-12B (G4-12B) (Team et al., 2026), including both thinking and non-thinking

variants of Qwen3. We use seven benchmarks covering mathematical reasoning (GSM8K (Cobbe et al., 2021), MATH-500 (Lightman et al., 2024), AIME25 (Mathematical Association of America, 2025)), code generation (HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), LiveCodeBench (Jain et al., 2024)), and dialogue (MT-Bench (Zheng et al., 2023)). Decoding is performed on a single NVIDIA RTX PRO 6000 GPU with a maximum generation length of 2,048 tokens.

Metrics. Following prior work (Li et al., 2026b; Chen et al., 2026; Zou et al., 2026), we evaluate every method with decoding batch size 1 at temperatures $T \in \{0, 1\}$. Because speculative decoding is lossless, we focus on two efficiency metrics: (i) the **speedup ratio** (SR), defined as the wall-clock decoding speed relative to vanilla autoregressive decoding, and (ii) the **average acceptance length** (τ), i.e., the mean number of draft tokens accepted in each draft–verification cycle.

Baselines. We use vanilla autoregressive decoding ($SR = 1.00\times$) as the latency reference. For all speculative-decoding methods, we adopt the DFlash (Chen et al., 2026) drafter architecture and initialize from its released checkpoint. We then train the drafter with different objectives, including the original DFlash CE, KL, LK (Samarin et al., 2026), D-PACE (Wu et al., 2026), and our proposed PC-VOT objective. We also include the results of official checkpoints in Appendix E.1. All variants share the same initialization, architecture, training data, optimization budget, and decoding pipeline.

Implementation. All methods are trained for six epochs on Open-PerfectBlend (Xu et al., 2024) using eight NVIDIA RTX PRO 6000 GPUs, with their respective training objectives. During training, we use AdamW with $(\beta_1, \beta_2) = (0.9, 0.999)$, weight decay 0.01, and gradient clipping with a maximum norm of 1.0. The learning rate is set to 1×10^{-4} and follows a cosine decay schedule with a 5% warmup ratio. We use on-policy training with a prompt batch size of 21 and maximum prompt and response lengths of 512 and 2,048 tokens, respectively. We set $\beta = 0.5$ for PC-VOT and otherwise keep the training configuration identical across all objectives. Results are averaged over three independent runs. Further implementation details are provided in Appendix D.

4.1 Main Results

As shown in Table 1, PC-VOT outperforms the competing training objectives across all models, tasks, and decoding temperatures in both speedup ratio and acceptance length. On average, PC-VOT improves the speedup ratio by 5.2% over D-PACE and 16.2% over DFlash, while increasing the acceptance length by 5.3% and 16.7%, respectively. Specifically, at $T = 0$, PC-VOT increases the speedup ratio over D-PACE from $5.36\times$ to $5.68\times$ on Q3-4B and from $5.31\times$ to $5.59\times$ on Q3-8B. Similar gains are observed on the thinking variants, from $4.27\times$ to $4.58\times$ on Q3-4B-T and from $4.24\times$ to $4.49\times$ on Q3-8B-T, with corresponding improvements in acceptance length. The same trend extends to Gemma4-12B, where PC-VOT improves the speedup ratio from $5.24\times$ to $5.45\times$, suggesting that verifier-induced point-to-set alignment generalizes across model scales and modes.

Across datasets, PC-VOT shows strong gains on AIME25, MATH-500, and MT-Bench, suggesting that its benefit is more pronounced on tasks with long-horizon generation or high prediction uncertainty. For example, on MATH-500 with Q3-4B at $T = 0$, PC-VOT increases the speedup ratio from $6.85\times$ to $7.54\times$, indicating the benefit of prefix-aware optimization when early draft errors can propagate through long reasoning trajectories. On MT-Bench with Q3-8B, PC-VOT also improves the speedup ratio from $2.84\times$ to $3.07\times$ and the acceptance length from 4.43 to 4.79. This larger gain on open-ended dialogue can be attributed to its higher prediction uncertainty, where verifier-aligned set matching provides a more appropriate constraint than exact distribution matching.

PC-VOT remains effective under both greedy and stochastic decoding. Under greedy decoding ($T = 0$), PC-VOT achieves average speedup ratios of $5.68\times$, $5.59\times$, and $5.45\times$, corresponding to relative improvements of 6.0%, 5.3%, and 4.0% over D-PACE, respectively. Under stochastic decoding ($T = 1$), it achieves $4.76\times$, $4.59\times$, and $4.66\times$ speedups, improving over D-PACE by 4.6%, 5.0%, and 3.6%, respectively. The smaller gains at $T = 1$ arise because the flatter target distribution makes verifier sampling more stochastic, causing even a Bayes-optimal top-1 draft proposal to match the sampled target token less often and thus reducing the improvement of PC-VOT.

4.2 Ablation Study

In this section, we ablate the impact of key components in the PC-VOT objective. Due to the space limitation, we defer the extended ablation analysis regarding inference batch size to Appendix E.2.

Table 1: Comparison of decoding speedup ratio (SR) and average acceptance length (τ) on standard LLM benchmarks under $T \in \{0, 1\}$. All baselines share the same drafter and continued-training setup, differing only in objective. Comparisons with the original checkpoints are provided in Table 5.

Model	Method	MATH						CODE						CHAT		*Avg.*	
		GSM8K		MATH-500		AIME25		HumanEval		MBPP		LCB		MT-Bench			
$T=0$		SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ
Q3-4B	DFlash	5.37×	6.74	6.31×	8.04	5.82×	7.38	5.27×	6.67	4.76×	6.16	5.38×	7.03	2.86×	4.47	5.11×	6.64
	KL	5.76×	7.16	6.70×	8.45	5.98×	7.56	5.24×	6.61	4.90×	6.31	5.45×	7.13	2.93×	4.59	5.28×	6.83
	LK	5.82×	7.25	6.77×	8.54	6.06×	7.67	5.27×	6.66	4.94×	6.34	5.47×	7.17	2.92×	4.57	5.32×	6.89
	D-PACE	5.89×	7.36	6.85×	8.61	6.09×	7.76	5.35×	6.76	4.99×	6.42	5.49×	7.21	2.89×	4.51	5.36×	6.95
	PC-VOT	**6.28×**	**7.81**	**7.54×**	**9.37**	**6.34×**	**8.06**	**5.54×**	**7.02**	**5.28×**	**6.77**	**5.72×**	**7.48**	**3.09×**	**4.81**	**5.68×**	**7.33**
Q3-8B	DFlash	5.21×	6.69	6.30×	8.13	5.76×	7.34	5.17×	6.59	4.67×	6.03	5.37×	7.17	2.77×	4.31	5.04×	6.61
	KL	5.60×	7.23	6.66×	8.72	5.72×	7.28	5.12×	6.51	4.90×	6.35	5.46×	7.32	2.83×	4.41	5.18×	6.83
	LK	5.68×	7.36	6.73×	8.84	5.78×	7.36	5.22×	6.64	4.96×	6.41	5.52×	7.40	2.80×	4.36	5.24×	6.91
	D-PACE	5.78×	7.53	6.82×	9.01	5.89×	7.48	5.28×	6.73	5.01×	6.46	5.56×	7.46	2.84×	4.43	5.31×	7.01
	PC-VOT	**6.11×**	**7.96**	**7.46×**	**9.78**	**6.22×**	**7.97**	**5.37×**	**6.85**	**5.21×**	**6.69**	**5.72×**	**7.70**	**3.07×**	**4.79**	**5.59×**	**7.39**
G4-12B	DFlash	5.31×	6.16	5.77×	6.73	5.19×	6.04	5.72×	6.70	4.43×	5.05	5.09×	6.01	2.82×	3.78	4.90×	5.78
	KL	6.11×	7.11	6.43×	7.61	5.24×	6.16	5.78×	6.82	4.60×	5.29	5.19×	6.18	2.87×	3.91	5.17×	6.15
	LK	6.12×	7.14	6.48×	7.71	5.27×	6.23	5.80×	6.87	4.70×	5.46	5.26×	6.30	2.89×	3.96	5.22×	6.24
	D-PACE	6.20×	7.26	6.56×	7.84	5.30×	6.30	5.83×	6.92	4.66×	5.39	5.22×	6.23	2.91×	4.02	5.24×	6.28
	PC-VOT	**6.48×**	**7.65**	**6.94×**	**8.28**	**5.53×**	**6.63**	**5.94×**	**7.11**	**4.78×**	**5.59**	**5.46×**	**6.57**	**3.02×**	**4.28**	**5.45×**	**6.59**
Q3-4B-T	DFlash	3.80×	4.69	3.97×	4.93	3.89×	4.83	3.53×	4.54	3.10×	3.97	2.89×	3.70	2.58×	3.40	3.39×	4.29
	KL	4.79×	5.93	4.95×	6.17	4.67×	5.81	4.41×	5.47	4.22×	5.22	4.06×	4.99	2.82×	3.81	4.27×	5.34
	LK	4.82×	5.97	4.98×	6.21	4.70×	5.86	4.43×	5.50	4.25×	5.26	4.10×	5.07	2.80×	3.78	4.30×	5.38
	D-PACE	4.87×	6.04	5.04×	6.28	4.63×	5.76	4.39×	5.45	4.18×	5.17	4.02×	4.94	2.77×	3.74	4.27×	5.34
	PC-VOT	**5.19×**	**6.42**	**5.34×**	**6.64**	**5.06×**	**6.26**	**4.68×**	**5.76**	**4.51×**	**5.53**	**4.27×**	**5.24**	**2.98×**	**4.10**	**4.58×**	**5.71**
Q3-8B-T	DFlash	3.67×	4.55	4.23×	5.35	3.97×	5.06	3.53×	4.41	3.18×	3.95	2.96×	3.67	2.51×	3.42	3.44×	4.34
	KL	4.62×	5.78	5.02×	6.37	4.68×	5.97	4.25×	5.29	4.17×	5.20	3.97×	4.94	2.85×	4.07	4.22×	5.37
	LK	4.65×	5.82	5.06×	6.43	4.71×	6.01	4.28×	5.35	4.19×	5.24	3.99×	4.97	2.83×	4.02	4.24×	5.41
	D-PACE	4.73×	5.92	5.13×	6.52	4.66×	5.95	4.24×	5.28	4.13×	5.14	3.94×	4.91	2.87×	4.11	4.24×	5.40
	PC-VOT	**5.01×**	**6.24**	**5.42×**	**6.86**	**5.01×**	**6.37**	**4.43×**	**5.51**	**4.36×**	**5.42**	**4.17×**	**5.16**	**3.05×**	**4.46**	**4.49×**	**5.72**
$T=1$		SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ
Q3-4B	DFlash	4.73×	6.03	5.11×	6.72	3.77×	4.91	4.77×	6.07	4.46×	5.67	5.01×	6.77	2.67×	4.05	4.36×	5.75
	KL	5.05×	6.42	5.40×	7.06	3.85×	5.01	4.75×	6.03	4.57×	5.78	5.05×	6.84	2.71×	4.14	4.48×	5.90
	LK	5.11×	6.51	5.46×	7.14	3.91×	5.08	4.76×	6.06	4.61×	5.82	5.07×	6.88	2.71×	4.12	4.52×	5.94
	D-PACE	5.16×	6.59	5.49×	7.18	3.94×	5.13	4.85×	6.18	4.65×	5.88	5.09×	6.91	2.69×	4.09	4.55×	5.99
	PC-VOT	**5.47×**	**6.97**	**5.86×**	**7.59**	**4.11×**	**5.36**	**4.99×**	**6.37**	**4.81×**	**6.10**	**5.25×**	**7.13**	**2.84×**	**4.31**	**4.76×**	**6.26**
Q3-8B	DFlash	4.67×	6.04	4.82×	6.44	3.69×	4.98	4.37×	5.59	4.13×	5.34	5.03×	6.77	2.49×	3.76	4.17×	5.56
	KL	5.01×	6.52	5.07×	6.83	3.66×	4.93	4.33×	5.53	4.27×	5.52	5.08×	6.89	2.54×	3.83	4.28×	5.72
	LK	5.08×	6.61	5.11×	6.89	3.70×	5.00	4.38×	5.60	4.30×	5.56	5.13×	6.96	2.52×	3.80	4.32×	5.77
	D-PACE	5.16×	6.73	5.17×	6.96	3.79×	5.14	4.41×	5.65	4.33×	5.60	5.16×	7.01	2.55×	3.85	4.37×	5.85
	PC-VOT	**5.43×**	**7.07**	**5.60×**	**7.56**	**4.07×**	**5.53**	**4.55×**	**5.82**	**4.48×**	**5.79**	**5.28×**	**7.22**	**2.74×**	**4.14**	**4.59×**	**6.16**
G4-12B	DFlash	4.60×	5.40	4.94×	5.83	4.33×	5.09	4.98×	6.04	3.84×	4.43	4.40×	5.27	2.54×	3.49	4.23×	5.08
	KL	5.21×	6.16	5.50×	6.58	4.36×	5.18	5.03×	6.14	3.92×	4.55	4.47×	5.39	2.58×	3.55	4.44×	5.36
	LK	5.29×	6.26	5.54×	6.67	4.38×	5.24	5.04×	6.16	3.97×	4.63	4.52×	5.48	2.59×	3.57	4.48×	5.43
	D-PACE	5.35×	6.33	5.60×	6.76	4.40×	5.29	5.06×	6.20	3.95×	4.59	4.50×	5.44	2.61×	3.61	4.50×	5.46
	PC-VOT	**5.58×**	**6.67**	**5.92×**	**7.13**	**4.58×**	**5.58**	**5.14×**	**6.34**	**4.03×**	**4.73**	**4.69×**	**5.71**	**2.70×**	**3.76**	**4.66×**	**5.70**
Q3-4B-T	DFlash	3.42×	4.29	3.57×	4.46	3.24×	4.11	3.06×	3.77	2.66×	3.26	2.58×	3.17	2.27×	2.90	2.97×	3.71
	KL	3.95×	4.94	4.14×	5.21	3.60×	4.62	3.66×	4.52	3.49×	4.30	3.34×	4.16	2.55×	3.21	3.53×	4.42
	LK	3.97×	4.97	4.15×	5.23	3.62×	4.65	3.67×	4.53	3.51×	4.33	3.35×	4.19	2.54×	3.19	3.54×	4.44
	D-PACE	4.01×	5.03	4.20×	5.29	3.58×	4.59	3.64×	4.50	3.47×	4.27	3.30×	4.11	2.52×	3.15	3.53×	4.42
	PC-VOT	**4.19×**	**5.27**	**4.33×**	**5.46**	**3.82×**	**4.88**	**3.92×**	**4.82**	**3.71×**	**4.55**	**3.53×**	**4.39**	**2.60×**	**3.28**	**3.73×**	**4.66**
Q3-8B-T	DFlash	3.28×	4.11	3.85×	4.87	3.42×	4.39	3.06×	3.84	2.65×	3.31	2.47×	3.09	2.18×	2.85	2.99×	3.78
	KL	4.05×	5.10	4.14×	5.28	3.70×	4.77	3.46×	4.32	3.19×	3.99	3.13×	3.94	2.43×	3.28	3.44×	4.38
	LK	4.08×	5.14	4.16×	5.31	3.71×	4.78	3.48×	4.36	3.20×	4.02	3.14×	3.95	2.42×	3.26	3.46×	4.40
	D-PACE	4.11×	5.18	4.21×	5.37	3.69×	4.76	3.45×	4.30	3.17×	3.96	3.10×	3.90	2.44×	3.30	3.45×	4.40
	PC-VOT	**4.31×**	**5.41**	**4.38×**	**5.59**	**3.95×**	**5.09**	**3.62×**	**4.49**	**3.37×**	**4.19**	**3.28×**	**4.12**	**2.55×**	**3.48**	**3.64×**	**4.62**

Effect of β. Table 2 examines the effect of the verifier margin β in Eq. (8). Overall, the moderate setting $\beta = 0.5$ achieves the best average performance at both temperatures. At $T = 0$, increasing β from 0.1 to 0.5 raises the average SR from 5.61× to 5.68× and τ from 7.21 to 7.33. Increasing β further to 1.0 reduces these values to 5.63× and 7.24, respectively. A similar trend is observed at $T = 1$, where the average SR and τ increase from 4.71× and 6.20 at $\beta = 0.1$ to 4.76× and 6.26 at $\beta = 0.5$, before decreasing to 4.69× and 6.17 at $\beta = 1.0$. These results reveal a trade-off between decision robustness and set-valued flexibility: a small β provides insufficient separation from the verifier boundary, whereas a large margin overly constrains the flexibility of point-to-set alignment.

Table 2: Ablation study of the verifier margin hyperparameter (β) in PC-VOT. We report speedup ratio (SR) and acceptance length (τ) on Qwen3-4B with temperature $T \in \{0, 1\}$.

β	MATH						CODE						CHAT		*Avg.*	
	GSM8K		MATH-500		AIME25		HumanEval		MBPP		LCB		MT-Bench			
$T=0$	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ
0.1	6.16×	7.62	7.46×	9.25	6.31×	8.01	5.48×	6.90	5.21×	6.68	5.61×	7.28	3.04×	4.70	5.61×	7.21
0.3	6.22×	7.73	7.51×	9.32	**6.35×**	**8.07**	5.51×	6.98	5.25×	6.72	5.67×	7.41	3.06×	4.73	5.65×	7.28
0.5	**6.28×**	**7.81**	**7.54×**	**9.37**	6.34×	8.06	**5.54×**	**7.02**	**5.28×**	**6.77**	**5.72×**	**7.48**	3.09×	4.81	**5.68×**	**7.33**
1.0	6.18×	7.65	7.47×	9.27	6.31×	8.03	5.49×	6.93	5.22×	6.69	5.62×	7.30	**3.11×**	**4.84**	5.63×	7.24
$T=1$	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ
0.1	5.40×	6.89	5.76×	7.47	4.08×	5.34	4.93×	6.27	4.76×	6.03	5.19×	7.03	**2.87×**	**4.34**	4.71×	6.20
0.3	5.41×	6.91	5.80×	7.50	4.08×	5.34	**5.01×**	**6.40**	**4.83×**	**6.12**	5.21×	7.06	2.79×	4.22	4.73×	6.22
0.5	**5.47×**	**6.97**	**5.86×**	**7.59**	**4.11×**	**5.36**	4.99×	6.37	4.81×	6.10	**5.25×**	**7.13**	2.84×	4.31	**4.76×**	**6.26**
1.0	5.39×	6.88	5.74×	7.45	4.07×	5.33	4.92×	6.27	4.75×	6.03	5.18×	7.02	2.76×	4.21	4.69×	6.17

Table 3: Ablation study of prefix coupling in PC-VOT. We report speedup ratio (SR) and acceptance length (τ) on Qwen3-4B with temperature $T \in \{0, 1\}$.

Method	MATH						CODE						CHAT		*Avg.*	
	GSM8K		MATH-500		AIME25		HumanEval		MBPP		LCB		MT-Bench			
$T=0$	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ
PC-VOT	**6.28×**	**7.81**	**7.54×**	**9.37**	**6.34×**	**8.06**	**5.54×**	**7.02**	**5.28×**	**6.77**	**5.72×**	**7.48**	**3.09×**	**4.81**	**5.68×**	**7.33**
w/o remaining-horizon factor	6.16×	7.62	7.48×	9.30	6.32×	8.01	5.49×	6.94	5.21×	6.69	5.62×	7.32	3.04×	4.72	5.62×	7.23
w/o prefix-reachability factor	6.09×	7.49	7.41×	9.21	6.31×	7.98	5.45×	6.88	5.15×	6.61	5.62×	7.32	3.03×	4.68	5.58×	7.17
w/o prefix coupling	6.03×	7.39	7.40×	9.19	6.26×	7.93	5.41×	6.81	5.09×	6.54	5.52×	7.14	3.00×	4.64	5.53×	7.09
$T=1$	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ
PC-VOT	**5.47×**	**6.97**	**5.86×**	**7.59**	**4.11×**	**5.36**	**4.99×**	**6.37**	**4.81×**	**6.10**	**5.25×**	**7.13**	**2.84×**	**4.31**	**4.76×**	**6.26**
w/o remaining-horizon factor	5.43×	6.92	5.81×	7.54	4.08×	5.34	4.94×	6.30	4.77×	6.05	5.21×	7.09	2.80×	4.27	4.72×	6.22
w/o prefix-reachability factor	5.39×	6.84	5.76×	7.47	4.08×	5.34	4.91×	6.25	4.76×	6.03	5.20×	7.06	2.78×	4.22	4.70×	6.17
w/o prefix coupling	5.29×	6.72	5.70×	7.39	4.05×	5.31	4.88×	6.18	4.72×	5.98	5.13×	6.97	2.72×	4.15	4.64×	6.10

Effect of Prefix Coupling. Table 3 ablates the remaining-horizon $(n - i + 1)$ and prefix-reachability R_{i-1} factors in Eq. (16). At $T = 0$, removing the remaining-horizon factor, the prefix-reachability factor, or both reduces the average SR from $5.68\times$ to $5.62\times$, $5.58\times$, and $5.53\times$, respectively. The same trend holds at $T = 1$, where the corresponding variants reduce SR from $4.76\times$ to $4.72\times$, $4.70\times$, and $4.64\times$. Removing prefix reachability causes a larger degradation than removing the remaining-horizon term, indicating that reachability contributes more strongly to performance. Nevertheless, the consistent drop when either factor is removed suggests that the two provide complementary training signals. These factors provide complementary training signals that help the drafter focus on positions with greater verification relevance and downstream impact, ultimately improving decoding efficiency.

Overall, these ablation results provide empirical support for the key design choices of PC-VOT. The margin analysis confirms the importance of balancing verifier-boundary separation with set-valued flexibility, while the prefix-coupling study demonstrates the complementary roles of reachability and remaining-horizon weighting. Together, these findings substantiate the proposed training objectives and their contribution to decoding efficiency across benchmark tasks.

5 Conclusion

We introduce PC-VOT, a verifier-aligned training objective for diffusion speculative decoding. By combining point-to-set optimal transport with prefix coupling, PC-VOT aligns local decision repair with the verifier's first-failure utility. We theoretically show that minimizing PC-VOT tightens a lower bound on expected accepted length. Across diverse tasks, PC-VOT improves accepted length and decoding speedup without introducing additional inference-time components.

Limitations. Due to computational constraints, it is hard for us to scale the draft model to a larger size. Based on the empirical training compute (Appendix E.3), training a DFlash-style drafter for a 27B target model would require approximately 2,500 GPU-hours on eight NVIDIA RTX PRO 6000 96GB GPUs. In practice, DeepSpeed ZeRO Stage 3 introduces additional communication and parameter-sharding overhead, increasing the actual training cost beyond this estimate.

AI USE STATEMENT

In this work, generative AI tools were used solely to improve the readability of the manuscript, including grammar, spelling, and word choice. They were not used to generate research ideas or synthetic data; develop theoretical models or conceptual frameworks; formulate mathematical claims or assist with proofs; propose or refine hypotheses; design or provide feedback on research methodology or experiments; implement methods; translate content; clean or reformat data; support qualitative or thematic data analysis; or interpret experimental results. All research ideas, methods, analyses, and experimental results presented in this work are original contributions of the authors. All AI-assisted edits were manually reviewed by the authors to ensure that they preserved the intended meaning and technical accuracy of the manuscript.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. 7

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. *arXiv preprint arXiv:2401.10774*, 2024. 3

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. 1, 3

Jian Chen, Yesheng Liang, and Zhijian Liu. Dflash: Block diffusion for flash speculative decoding. *arXiv preprint arXiv:2602.06036*, 2026. 1, 3, 7, 18

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. 7

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable, robust, and hardware-aware speculative decoding. *arXiv preprint arXiv:2402.12374*, 2024. 3

Xin Cheng, Xingkai Yu, Chenze Shao, Jiashi Li, Yunfan Xiong, Yi Qian, Jiaqi Zhu, Shirong Ma, Xiaokang Zhang, Jiasheng Ye, et al. Dspark: Confidence-scheduled speculative decoding with semi-autoregressive generation. *arXiv preprint arXiv:2607.05147*, 2026. 3

Jacob K Christopher, Brian R Bartoldson, Tal Ben-Nun, Michael Cardei, Bhavya Kailkhura, and Ferdinando Fioretto. Speculative diffusion decoding: Accelerating language generation through diffusion. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 12042–12059, 2025. 3

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. 7

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. 3

Jiale Fu, Yuchu Jiang, Junkai Chen, Jiaming Fan, Xin Geng, and Xu Yang. Fast large language model collaborative decoding via speculation. *arXiv preprint arXiv:2502.01662*, 2025. 3

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of llm inference using lookahead decoding. *arXiv preprint arXiv:2402.02057*, 2024. 3

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. 1

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason 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, 2024. 3

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Hasan Genc, Kurt Keutzer, Amir Gholami, and Yakun Sophia Shao. Speed: Speculative pipelined execution for efficient decoding. In *Enhancing LLM Performance: Efficacy, Fine-Tuning, and Inference Techniques*, pp. 19–32. Springer, 2025. 3

Lanxiang Hu, Zhaoxiang Feng, Yulun Wu, Haoran Yuan, Yujie Zhao, Yu-Yang Qian, Bojun Wang, Peng Zhao, Daxin Jiang, Yibo Zhu, Tajana Rosing, and Hao Zhang. Jetspec: Breaking the scaling ceiling of speculative decoding with parallel tree drafting, 2026a. URL https://arxiv.org/abs/2606.18394. 3

Shijing Hu, Jingyang Li, Xingyu Xie, Zhihui Lu, Kim-Chuan Toh, and Pan Zhou. Griffin: Effective token alignment for faster speculative decoding. *Advances in Neural Information Processing Systems*, 38:125648–125673, 2026b. 3

Zhengmian Hu, Tong Zheng, Vignesh Viswanathan, Ziyi Chen, Ryan Rossi, Yihan Wu, Dinesh Manocha, and Heng Huang. Towards optimal multi-draft speculative decoding. In *International Conference on Learning Representations*, volume 2025, pp. 3181–3203, 2025. 3

Jianuo Huang, Yaojie Zhang, Qituan Zhang, Hao Lin, Hanlin Xu, and Linfeng Zhang. Domino: Decoupling causal modeling from autoregressive drafting in speculative decoding. *arXiv preprint arXiv:2605.29707*, 2026. 3

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024. 7

Haodi Lei, Yafu Li, Haoran Zhang, Shunkai Zhang, Qianjia Cheng, Xiaoye Qu, Ganqu Cui, Bowen Zhou, Ning Ding, Yun Luo, et al. Draft-opd: On-policy distillation for speculative draft models. *arXiv preprint arXiv:2605.29343*, 2026. 3

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023. 1, 2

Guanghao Li, Zhihui Fu, Min Fang, Qibin Zhao, Ming Tang, Chun Yuan, and Jun Wang. Diffuspec: Unlocking diffusion language models for speculative decoding. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 20896–20910, 2026a. 3

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024a. 3

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, 2024b. 3

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *Advances in Neural Information Processing Systems*, 38:136737–136756, 2026b. 3, 7

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison 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*, volume 2024, pp. 39578–39601, 2024. 7

Fangcheng Liu, Yehui Tang, Zhenhua Liu, Yunsheng Ni, Kai Han, and Yunhe Wang. Kangaroo: Lossless self-speculative decoding via double early exiting. *arXiv preprint arXiv:2404.18911*, 2024. 3

Xukun Liu, Bowen Lei, Ruqi Zhang, and Dongkuan DK Xu. Adaptive draft-verification for efficient large language model decoding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 24668–24676, 2025. 3

Mathematical Association of America. American invitational mathematics examination (aime). https://maa.org/maa-invitational-competitions/, 2025. 7

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, et al. 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*, pp. 932–949, 2024. 3

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*, volume 2025, pp. 44949–44987, 2025. 3

Jie Ou, Yueming Chen, et al. Lossless acceleration of large language model via adaptive n-gram parallel decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, pp. 10–22, 2024. 3

Gabriel Peyré and Marco Cuturi. Computational optimal transport, 2020. URL https://arxiv.org/abs/1803.00567. 5

Zongyue Qin, Ziniu Hu, Zifan He, Neha Prakriya, Jason Cong, and Yizhou Sun. Optimized multi-token joint decoding with auxiliary model for llm inference. *arXiv preprint arXiv:2407.09722*, 2024. 3

Liran Ringel and Yaniv Romano. Accelerating speculative decoding with block diffusion draft trees. *arXiv preprint arXiv:2604.12989*, 2026. 3

Alexander Samarin, Sergei Krutikov, Anton Shevtsov, Sergei Skvortsov, Filipp Fisin, and Alexander Golubev. Lk losses: Direct acceptance rate optimization for speculative decoding. *arXiv preprint arXiv:2602.23881*, 2026. 7

Mohammad Samragh, Arnav Kundu, David Harrison, Kumari Nishu, Devang Naik, Minsik Cho, and Mehrdad Farajtabar. Your llm knows the future: Uncovering its multi-token prediction potential. *arXiv preprint arXiv:2507.11851*, 2025. 3

Jameson Sandler, Jacob Christopher, Tom Hartvigsen, and Ferdinando Fioretto. Specdiff-2: Scaling diffusion drafter alignment for faster speculative decoding. *Proceedings of Machine Learning and Systems*, 8:1128–1147, 2026. 3

Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodolà. Accelerating transformer inference for translation via parallel decoding. *arXiv preprint arXiv:2305.10427*, 2023. 3

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31, 2018. 1, 3

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. Spectr: Fast speculative decoding via optimal transport. *Advances in Neural Information Processing Systems*, 36:30222–30242, 2023. 1, 3

Ziteng Sun, Uri Mendlovic, Yaniv Leviathan, Asaf Aharoni, Jae Hun Ro, Ahmad Beirami, and Ananda Theertha Suresh. Block verification accelerates speculative decoding. *arXiv preprint arXiv:2403.10444*, 2024. 3

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026. 6

Yao Teng, Fuyun Wang, Xian Liu, Zhekai Chen, Han Shi, Yu Wang, Zhenguo Li, Weiyang Liu, Difan Zou, and Xihui Liu. Speculative jacobi-denoising decoding for accelerating autoregressive text-to-image generation. *arXiv preprint arXiv:2510.08994*, 2025. 3

Lifu Wang and Pan Zhou. Smart: When is it actually worth expanding a speculative tree? *arXiv preprint arXiv:2604.09731*, 2026. 3

Tianyu Wu, Yu Yao, Zhenting Qi, Han Zheng, Zhuohan Wang, Haoran Ma, Lawrence Liao, Himabindu Lakkaraju, Ju Li, and Yilun Du. D-pace: Dynamic position-aware cross-entropy for parallel speculative drafting. *arXiv preprint arXiv:2605.18810*, 2026. 3, 4, 5, 7

Heming Xia, Yongqi Li, Jun Zhang, Cunxiao Du, and Wenjie Li. Swift: On-the-fly self-speculative decoding for llm inference acceleration. *arXiv preprint arXiv:2410.06916*, 2024. 3

Tengyu Xu, Eryk Helenowski, Karthik Abinav Sankararaman, Di Jin, Kaiyan Peng, Eric Han, Shaoliang Nie, Chen Zhu, Hejia Zhang, Wenxuan Zhou, et al. The perfect blend: Redefining rlhf with mixture of judges. *arXiv preprint arXiv:2409.20370*, 2024. 7, 18

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a. 6

Sen Yang, Shujian Huang, Xinyu Dai, and Jiajun Chen. Multi-candidate speculative decoding. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pp. 335–348. Springer, 2025b. 3

Lefan Zhang, Xiaodan Wang, Yanhua Huang, and Ruiwen Xu. Learning harmonized representations for speculative sampling. In *International Conference on Learning Representations*, volume 2025, pp. 35367–35388, 2025. 3, 4

Weilin Zhao, Yuxiang Huang, Xu Han, Wang Xu, Chaojun Xiao, Xinrong Zhang, Yewei Fang, Kaihuo Zhang, Zhiyuan Liu, and Maosong Sun. Ouroboros: Generating longer drafts phrase by phrase for faster speculative decoding, 2024. URL https://arxiv.org/abs/2402.13720. 3

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023. 7

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. Distillspec: Improving speculative decoding via knowledge distillation. In *International Conference on Learning Representations*, volume 2024, pp. 32011–32050, 2024. 3, 4

Yuxuan Zhou, Fei Huang, Heng Li, Fengyi Wu, Tianyu Wang, Jianwei Zhang, Junyang Lin, and Zhi-Qi Cheng. Overcoming joint intractability with lossless hierarchical speculative decoding. *arXiv preprint arXiv:2601.05724*, 2026. 3

Xiandong Zou, Jianshu Li, Jing Huang, and Pan Zhou. Variational speculative decoding: Rethinking draft training from token likelihood to sequence acceptance. *arXiv preprint arXiv:2602.05774*, 2026. 3, 5, 7

A Notation and Definitions

The major notations used throughout the paper are summarized in Table 4. We write $[n] = \{1, \dots, n\}$ and suppress the dependence of distributions on the context c when the context is clear.

Table 4: Core notations used in PC-VOT.

Notation	Definition
P	Frozen autoregressive target model.
Q_ϕ	Diffusion drafter parameterized by ϕ.
p_i	Target token distribution at draft position i.
$q_{\phi,i}$	Draft token distribution at position i; we write q_i when ϕ is unambiguous.
$t_i^\star$	Bayes-optimal verifier decision, defined by $t_i^\star = \arg\max_v p_i(v)$.
$\alpha_i(q_i)$	Expected local acceptance probability $p_i(\arg\max_v q_i(v))$.
R_i	Expected prefix reachability $R_i = \prod_{j=1}^{i} \alpha_j(q_j)$, with $R_0 = 1$.
$\mathcal{A}_i^\beta(t_i^\star)$	Verifier-feasible set that preserves a fraction β of the target decision margins.
β	Margin fraction controlling the strength of the verifier-feasible constraints.
$W_{C_i}(q_i, \tilde{q}_i)$	Optimal-transport discrepancy between two token distributions.
$D_i^\beta(q_i; t_i^\star)$	Minimum transport from q_i to $\mathcal{A}_i^\beta(t_i^\star)$.
w_i	Prefix-dependent utility weight $(n - i + 1)R_{i-1}$.
$\mathcal{L}_{\text{PC-VOT}}$	Prefix-weighted point-to-set transport objective.
L_{acc}	Number of consecutively accepted draft tokens.

B Algorithm

B.1 Algorithm Detail

Given a frozen target model P, an initialized drafter Q_ϕ, and training prompts $\mathcal{D}_{\text{prompt}}$, PC-VOT trains the drafter on on-policy speculative trajectories. At each visited context, it computes verifier-induced point-to-set transport costs and couples them across positions using prefix reachability and remaining speculative progress. Only the drafter parameters are updated.

Algorithm 1 On-Policy PC-VOT Training

Require: Frozen target model P, initialized drafter Q_ϕ, training prompts $\mathcal{D}_{\text{prompt}}$, feasibility margin β
Ensure: Trained drafter $Q_{\phi^\star}$

for each training iteration **do**
  $\mathcal{B} \sim \mathcal{D}_{\text{prompt}}$, $\mathcal{L}_\mathcal{B} \leftarrow 0$
  for all $c_0 \in \mathcal{B}$ **do**
    $c \leftarrow c_0$
    while c is non-terminal **do**
      $(\hat{x}_{1:n}, \{q_i\}_{i=1}^n) \leftarrow Q_\phi(c)$
      $(\{p_i\}_{i=1}^n, y) \leftarrow \text{SpecVerify}(P, c, \hat{x}_{1:n})$
      $R_0 \leftarrow 1$
      for $i = 1, \dots, n$ **do**
        $t_i^\star \leftarrow \arg\max_{a \in \mathcal{V}} p_i(a)$
        $\mathcal{A}_i^\beta(t_i^\star) \leftarrow \{\tilde{q}_i \in \Delta(\mathcal{V}) : \tilde{q}_i(t_i^\star) \geq \tilde{q}_i(v) + \beta[p_i(t_i^\star) - p_i(v)],\ \forall v \neq t_i^\star\}$
        $D_i^\beta \leftarrow \min_{\tilde{q}_i \in \mathcal{A}_i^\beta(t_i^\star)} W_{C_i}(q_i, \tilde{q}_i)$
        $\alpha_i \leftarrow p_i(\arg\max_{a \in \mathcal{V}} q_i(a))$
        $w_i \leftarrow (n - i + 1)\,\text{sg}[R_{i-1}]$
        $R_i \leftarrow R_{i-1}\alpha_i$
      end for
      $\ell \leftarrow \sum_{i=1}^n w_i D_i^\beta$
      $\mathcal{L}_\mathcal{B} \leftarrow \mathcal{L}_\mathcal{B} + \ell$
      $c \leftarrow c \oplus y$
    end while
  end for
  $\phi \leftarrow \text{Opt}(\phi, \nabla_\phi \mathcal{L}_\mathcal{B})$
end for
return $Q_{\phi^\star}$

Here, $\mathrm{SpecVerify}$ denotes the standard lossless speculative-verification procedure. Given the current context c and draft block $\hat{x}_{1:n}$, it returns the target distributions $\{p_i\}_{i=1}^{n}$ used for verification and the sequence y of tokens committed by the verifier in the current speculative step. The update $c \leftarrow c \oplus y$ appends these committed tokens to the current context, yielding the next on-policy rollout state. After accumulating the PC-VOT loss over the minibatch, $\phi \leftarrow \mathrm{Opt}(\phi, \nabla_\phi \mathcal{L}_\mathcal{B})$ updates only the drafter parameters using the chosen optimizer, while the target model P remains frozen throughout training.

B.2 Algorithm Rationale

On-policy training. Training trajectories are generated by the current drafter under the same speculative-decoding procedure used at inference. This reduces the state-distribution mismatch introduced by training on fixed target-generated responses and exposes the model to the contexts and draft errors it actually encounters. As the drafter improves, newly collected rollouts continuously adapt the training distribution to its evolving policy. The discrete rollout decisions are treated as sampled trajectory data and are not differentiated through.

Verifier-aligned optimization. At each visited context, the frozen target model provides the verifier distributions used to define the Bayes-optimal proposal and its verifier-feasible region. The drafter is optimized toward the closest feasible distribution rather than a prescribed target distribution, while prefix coupling emphasizes positions according to their probability of being reached and the remaining speculative progress at risk. Prefix reachability is detached so that it acts only as an importance weight.

Optimization scope. The target model remains frozen and is used only for verification and supervision. Gradients update only the drafter, leaving the standard lossless speculative-decoding procedure unchanged at inference.

C Proofs

C.1 Proof of Lemma 1

Proof. Fix a position i and a draft distribution q_i. Under the fixed tie-breaking rule, denote its deterministic top-1 proposal by

$$\hat{t}_i = \arg\max_{v \in \mathcal{V}} q_i(v). \tag{19}$$

Taking the expectation over the target token $t_i \sim p_i$ gives

$$\begin{aligned} \alpha_i(q_i) &= \mathbb{E}_{t_i \sim p_i}\left[\mathbb{1}\{\hat{t}_i = t_i\}\right] \\ &= \sum_{v \in \mathcal{V}} p_i(v)\mathbb{1}\{\hat{t}_i = v\} \\ &= p_i(\hat{t}_i) = p_i\left(\arg\max_{v \in \mathcal{V}} q_i(v)\right). \end{aligned} \tag{20}$$

Therefore, for any draft distribution q_i,

$$\alpha_i(q_i) = p_i(\hat{t}_i) \le \max_{v \in \mathcal{V}} p_i(v). \tag{21}$$

The upper bound is attained by choosing any

$$t_i^\star \in \arg\max_{v \in \mathcal{V}} p_i(v), \tag{22}$$

because a draft distribution whose top-1 proposal is $t_i^\star$ achieves

$$\alpha_i(q_i) = p_i(t_i^\star) = \max_{v \in \mathcal{V}} p_i(v) = \alpha_i^\star. \tag{23}$$

In particular, the decision region $\mathcal{A}_i(t_i^\star)$ is nonempty because it contains the point mass $\delta_{t_i^\star}$. Moreover, by the definition of $\mathcal{A}_i(t_i^\star)$, every $q_i \in \mathcal{A}_i(t_i^\star)$ satisfies

$$\arg\max_{v \in \mathcal{V}} q_i(v) = t_i^\star. \tag{24}$$

Consequently,

$$\alpha_i(q_i) = p_i(t_i^\star) = \alpha_i^\star, \qquad \forall q_i \in \mathcal{A}_i(t_i^\star). \tag{25}$$

□

C.2 Proof of Lemma 2

Proof. Fix a position i. We have

$$W_{C_i}(q_i, \tilde{q}_i) = 0 \quad \Longleftrightarrow \quad q_i = \tilde{q}_i. \tag{26}$$

If $q_i \in \mathcal{A}_i^\beta(t_i^\star)$, then q_i itself is a feasible transport endpoint. Hence,

$$D_i^\beta(q_i; t_i^\star) \le W_{C_i}(q_i, q_i) = 0. \tag{27}$$

Since optimal-transport costs are nonnegative, this implies $D_i^\beta(q_i; t_i^\star) = 0$.

Conversely, suppose that $D_i^\beta(q_i; t_i^\star) = 0$. The feasible set $\mathcal{A}_i^\beta(t_i^\star)$ is a closed subset of the probability simplex and is therefore compact. Since W_{C_i} is continuous, the minimum in Eq. (11) is attained by some $\tilde{q}_i^\star \in \mathcal{A}_i^\beta(t_i^\star)$. Consequently,

$$W_{C_i}(q_i, \tilde{q}_i^\star) = 0. \tag{28}$$

By Eq. (26), $q_i = \tilde{q}_i^\star$, and hence $q_i \in \mathcal{A}_i^\beta(t_i^\star)$. This proves Eq. (12).

Because $t_i^\star$ is a maximizer of p_i, for every $v \neq t_i^\star$,

$$p_i(t_i^\star) - p_i(v) \ge 0. \tag{29}$$

For any $0 < \beta \le 1$, it follows that

$$p_i(t_i^\star) - p_i(v) \ge \beta \left[p_i(t_i^\star) - p_i(v) \right]. \tag{30}$$

Therefore, $p_i \in \mathcal{A}_i^\beta(t_i^\star)$.

Moreover, for the one-hot distribution $\delta_{t_i^\star}$,

$$\delta_{t_i^\star}(t_i^\star) - \delta_{t_i^\star}(v) = 1 \ge \beta \left[p_i(t_i^\star) - p_i(v) \right], \qquad \forall v \neq t_i^\star, \tag{31}$$

where the inequality follows from $0 < \beta \le 1$ and $0 \le p_i(t_i^\star) - p_i(v) \le 1$. Thus, $\delta_{t_i^\star} \in \mathcal{A}_i^\beta(t_i^\star)$.

Since both p_i and $\delta_{t_i^\star}$ are feasible transport endpoints in Eq. (11), choosing each of them separately gives

$$D_i^\beta(q_i; t_i^\star) \le W_{C_i}(q_i, p_i) \tag{32}$$

and

$$D_i^\beta(q_i; t_i^\star) \le W_{C_i}(q_i, \delta_{t_i^\star}). \tag{33}$$

Combining the two inequalities yields

$$D_i^\beta(q_i; t_i^\star) \le \min \left\{ W_{C_i}(q_i, p_i), W_{C_i}(q_i, \delta_{t_i^\star}) \right\}. \tag{34}$$

□

C.3 Proof of Theorem 3

Proof. For an arbitrary context c and a realized draft block of length n, let $\hat{t}_i = \arg\max_{v \in \mathcal{V}} q_i(v)$, and let $t_{1:n}^\star$ denote the target's greedy trajectory. Define the local acceptance probabilities evaluated along the respective prefixes by the frozen target model P as

$$\alpha_i = p_i(\hat{t}_i \mid c, \hat{t}_{<i}), \qquad \alpha_i^\star = p_i(t_i^\star \mid c, t_{<i}^\star). \tag{35}$$

The corresponding prefix reachabilities are

$$R_i = \prod_{j=1}^{i} \alpha_j, \qquad R_i^\star = \prod_{j=1}^{i} \alpha_j^\star, \qquad R_0 = R_0^\star = 1. \tag{36}$$

By sequential verification and the law of total expectation,

$$\mathbb{E}[L_{\text{acc}}] = \mathbb{E}\left[\sum_{k=1}^{n} R_k \right], \qquad n^\star = \mathbb{E}\left[\sum_{k=1}^{n} R_k^\star \right]. \tag{37}$$

If the draft block coincides with the target's greedy trajectory, then $R_k = R_k^\star$ for every k, and the desired bound follows from the nonnegativity of the loss. Otherwise, let ℓ denote the first position at which they differ:

$$\ell = \min\{i : \hat{t}_i \neq t_i^\star\}. \tag{38}$$

The two trajectories have identical prefixes before position ℓ, and hence $R_k = R_k^\star$ for $k < \ell$. Since all acceptance probabilities lie in $[0, 1]$, it follows that

$$\begin{aligned}\sum_{k=1}^{n}(R_k^\star - R_k) &= \sum_{k=\ell}^{n}(R_k^\star - R_k) \\ &\leq \sum_{k=\ell}^{n} R_k^\star \\ &= R_{\ell-1}\sum_{k=\ell}^{n}\prod_{j=\ell}^{k}\alpha_j^\star \\ &\leq (n-\ell+1)R_{\ell-1}.\end{aligned} \tag{39}$$

We next relate this potential shortfall to the point-to-set transport cost at the first disagreement. Because the prefixes coincide before position ℓ, the target distribution used by the draft objective at that position coincides with the distribution on the greedy trajectory. The assumed minimum greedy-path margin therefore implies

$$\alpha_\ell^\star - \alpha_\ell = p_\ell(t_\ell^\star) - p_\ell(\hat{t}_\ell) \geq \gamma. \tag{40}$$

Let $b = \hat{t}_\ell$. Since b is the draft's top-1 token,

$$q_\ell(b) \geq q_\ell(t_\ell^\star). \tag{41}$$

For any feasible endpoint $r \in \mathcal{A}_\ell^\beta(t_\ell^\star)$, the target-adaptive margin constraint gives

$$r(t_\ell^\star) - r(b) \geq \beta\left[p_\ell(t_\ell^\star) - p_\ell(b)\right] = \beta(\alpha_\ell^\star - \alpha_\ell). \tag{42}$$

Consider any coupling $\pi \in \Pi(q_\ell, r)$, and denote its off-diagonal transported mass by

$$M(\pi) = \sum_{u \neq v}\pi(u, v). \tag{43}$$

Under the unit ground cost, $M(\pi) = C_\ell(\pi)$. Using Eq. (41), we obtain

$$\begin{aligned}\beta(\alpha_\ell^\star - \alpha_\ell) &\leq \left[r(t_\ell^\star) - r(b)\right] - \left[q_\ell(t_\ell^\star) - q_\ell(b)\right] \\ &= \left[r(t_\ell^\star) - q_\ell(t_\ell^\star)\right] + \left[q_\ell(b) - r(b)\right] \\ &\leq 2M(\pi).\end{aligned} \tag{44}$$

The final inequality holds because the change in either coordinate is bounded in absolute value by the total off-diagonal mass of π. Taking the infimum over feasible endpoints and their couplings yields

$$D_\ell^\beta(q_\ell; t_\ell^\star) \geq \frac{\beta}{2}(\alpha_\ell^\star - \alpha_\ell) \geq \frac{\beta\gamma}{2}. \tag{45}$$

Let D_i^β be the local point-to-set cost in the PC-VOT objective, evaluated along the realized draft prefix. At the first disagreement, this cost equals $D_\ell^\beta(q_\ell; t_\ell^\star)$. Combining Eqs. (39) and (45) gives

$$\begin{aligned}\sum_{k=1}^{n}(R_k^\star - R_k) &\leq \frac{2}{\beta\gamma}(n-\ell+1)R_{\ell-1}D_\ell^\beta \\ &\leq \frac{2}{\beta\gamma}\sum_{i=1}^{n}(n-i+1)R_{i-1}D_i^\beta,\end{aligned} \tag{46}$$

where the second inequality follows because every term in the sum is nonnegative. The same bound holds when the two trajectories coincide.

Taking expectations over the draft-block randomness and using Eq. (37), we yield that

$$n^\star - \mathbb{E}[L_{\text{acc}}] \le \frac{2}{\beta\gamma}\mathbb{E}\left[\sum_{i=1}^{n}(n-i+1)R_{i-1}D_i^\beta\right] = \frac{2}{\beta\gamma}\mathcal{L}_{\text{PC-VOT}}(\phi). \tag{47}$$

Rearranging yields

$$\mathbb{E}[L_{\text{acc}}] \ge n^\star - \frac{2}{\beta\gamma}\mathcal{L}_{\text{PC-VOT}}(\phi). \tag{48}$$

□

D Implementation Details

For all trainable variants, we initialize the drafter from the publicly released DFlash checkpoint (Chen et al., 2026) and train it on Open-PerfectBlend (Xu et al., 2024). Training is performed for 6 epochs on 8 NVIDIA RTX PRO 6000 GPUs, with 7 GPUs used for drafter optimization and 1 GPU reserved for target-model inference. All objective variants use the same initialization, training data, optimization schedule, and compute budget, so that their only difference is the training objective. We optimize the drafter using AdamW with $(\beta_1, \beta_2) = (0.9, 0.999)$, weight decay 0.01, and gradient clipping with a maximum norm of 1.0. The learning rate is set to 1×10^{-4} and follows a cosine decay schedule with a 5% warmup ratio. Training is conducted on policy with a prompt batch size of 21. We use a micro-batch size of 1 per drafter GPU and accumulate 10 rollout batches between actor updates. The maximum prompt and response lengths are set to 512 and 2,048 tokens, respectively, and dynamic batching is enabled during drafter optimization. For on-policy data generation, responses are sampled from the current drafter and evaluated by the target model. The drafter rollout temperature is set to 0.0, while the target model uses a temperature of 1.0 during training.

Monte Carlo approximation of target-induced feasible set. Directly constructing and optimizing over the verifier-feasible set on the full vocabulary simplex is computationally intractable because of the large vocabulary size and the continuous family of feasible distributions. We therefore restrict the feasible distributions to a compact candidate support $\mathcal{S}_i$ consisting of the Bayes-optimal proposal $t_i^\star$ and the highest-probability competing tokens under the current drafter:

$$\mathcal{S}_i = \{t_i^\star\} \cup \text{Top}_{K-1}\left(q_i; \mathcal{V} \setminus \{t_i^\star\}\right).$$

We use $K = 16$, motivated by our empirical observation that the highest-probability draft tokens capture most of the probability mass relevant to speculative verification. This restriction substantially reduces the dimensionality of the endpoint search while retaining the dominant competing draft decisions. Distributions on $\mathcal{S}_i$ are extended by zeros outside this support. The original target and draft distributions, p_i and q_i, remain unchanged.

On this restricted support, we define the target-adaptive margin-constrained feasible set as

$$\mathcal{A}_{i,K}^{\beta}(t_i^\star) = \left\{\tilde{q}_i \in \Delta(\mathcal{S}_i) : \tilde{q}_i(t_i^\star) - \tilde{q}_i(v) \ge \beta\left[p_i(t_i^\star) - p_i(v)\right], \quad \forall v \in \mathcal{V} \setminus \{t_i^\star\}\right\}.$$

The coefficient β specifies the fraction of each target decision margin that the candidate distribution must preserve. The constraints also apply to tokens outside $\mathcal{S}_i$, ensuring that every restricted endpoint remains feasible under the full-vocabulary definition in Eq. (8).

Sampling directly from this constrained simplex would require enforcing the margin inequalities for every sample. To avoid rejection sampling, we first construct a sparse reference distribution that preserves the target margins. Let

$$\epsilon_i = \sum_{u \notin \mathcal{S}_i} p_i(u)$$

denote the target probability mass outside the candidate support. We reallocate the probability mass to the Bayes-optimal proposal and define

$$\bar{p}_i(v) = \begin{cases} p_i(t_i^\star) + \epsilon_i, & v = t_i^\star, \\ p_i(v), & v \in \mathcal{S}_i \setminus \{t_i^\star\}, \\ 0, & v \notin \mathcal{S}_i. \end{cases}$$

This construction yields a valid probability distribution supported on $\mathcal{S}_i$ and satisfies $\bar{p}_i(t_i^\star) - \bar{p}_i(v) \geq p_i(t_i^\star) - p_i(v), \forall v \neq t_i^\star$. We then use the reparameterization $\tilde{q}_i = \beta\bar{p}_i + (1-\beta)z_i$, where $z_i \in \Delta(\mathcal{S}_i)$ satisfies $z_i(t_i^\star) \geq z_i(v), \forall v \in \mathcal{S}_i \setminus \{t_i^\star\}$. Under this parameterization, for every $v \neq t_i^\star$,

$$\begin{aligned}\tilde{q}_i(t_i^\star) - \tilde{q}_i(v) &= \beta\big[\bar{p}_i(t_i^\star) - \bar{p}_i(v)\big] + (1-\beta)\big[z_i(t_i^\star) - z_i(v)\big] \\ &\geq \beta\big[p_i(t_i^\star) - p_i(v)\big].\end{aligned}$$

Thus, every generated distribution lies in $\mathcal{A}_{i,K}^{\beta}(t_i^\star)$ and assigns zero probability mass outside $\mathcal{S}_i$.

To generate z_i efficiently, we first sample $z_i' \sim \mathrm{Dirichlet}(\mathbb{1}_K)$, which provides a symmetric distribution over the probability simplex on K tokens. We then swap its largest component into the coordinate corresponding to $t_i^\star$. By symmetry, this yields a sample satisfying the required top-1 constraint without introducing a preference among the remaining competing tokens. The subsequent affine transformation produces a feasible endpoint directly, without rejection sampling.

The continuous feasible set still contains infinitely many such endpoints, making exact point-to-set minimization expensive during training. We therefore draw N independent feasible distributions

$$\left\{\tilde{q}_i^{(m)}\right\}_{m=1}^{N} \subseteq \mathcal{A}_{i,K}^{\beta}(t_i^\star)$$

and use them as a Monte Carlo approximation of the continuous feasible set. The empirical Monte Carlo point-to-set transport cost is

$$\widehat{D}_i^{\beta}(q_i; t_i^\star) = \min_{1 \leq m \leq N} W_{C_i}\left(q_i, \tilde{q}_i^{(m)}\right).$$

Hence, the Monte Carlo objective retains the point-to-set structure of the original formulation by minimizing over sampled feasible endpoints. Since every sampled endpoint satisfies the original feasibility constraints,

$$D_i^{\beta}(q_i; t_i^\star) \leq \widehat{D}_i^{\beta}(q_i; t_i^\star).$$

The sampled objective therefore provides an upper bound on the exact point-to-set distance. Increasing N can tighten this bound by improving coverage of the parameterized sampling family, at the cost of additional training overhead. We use $N = 16$ and set $\beta = 0.5$.

Prefix-coupled transport weighting. To account for the first-mismatch structure of speculative verification, we weight the local transport cost at position i by both the speculative progress at risk and the probability that verification reaches that position:

$$w_i = \underbrace{(n-i+1)}_{\text{remaining-horizon factor}} \cdot \underbrace{\mathrm{sg}[R_{i-1}]}_{\text{prefix-reachability factor}}, \tag{49}$$

where $n-i+1$ measures the number of current and subsequent draft positions forfeited by a mismatch at position i, R_{i-1} is the probability that the preceding draft prefix is accepted, and $\mathrm{sg}[\cdot]$ denotes the stop-gradient operator. We detach R_{i-1} so that prefix reachability acts solely as an importance weight rather than an additional optimization pathway. This prevents the model from reducing the objective by decreasing the reachability weight itself and instead directs optimization toward reducing the local transport cost at positions likely to be encountered during verification.

All PC-VOT-specific hyperparameters are selected by grid search on the validation set and then fixed across all experiments. PC-VOT is generally robust to small variations around the selected settings.

For evaluation, we report results averaged over three independent runs on a single NVIDIA RTX PRO 6000 GPU. All methods use the same speculative-decoding implementation and decoding configuration. We evaluate with decoding batch size 1 at temperatures $T \in \{0, 1\}$, with a maximum generation length of 2,048 tokens.

E Extended Experimental Results

E.1 Additional Experimental Results on Baselines

We compare our implementations of DFlash and D-PACE with their available official checkpoints on Qwen3-4B, Qwen3-4B-Thinking, and Qwen3-8B-Thinking. DFlash* and D-PACE* denote the

Table 5: Comparison of official checkpoints and their continued-trained counterparts. We report decoding speedup ratio (SR) and average acceptance length (τ) on Qwen3-4B, Qwen3-4B-Thinking, and Qwen3-8B-Thinking under $T \in \{0, 1\}$. DFlash* and D-PACE* denote official checkpoints, while DFlash and D-PACE denote their counterparts after continued training under our framework.

Model	Method	MATH						CODE						CHAT		*Avg.*	
		GSM8K		MATH-500		AIME25		HumanEval		MBPP		LCB		MT-Bench			
$T=0$		SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ
Q3-4B	DFlash*	5.19×	6.51	6.13×	7.83	5.70×	7.22	5.20×	6.58	4.68×	6.07	5.30×	6.92	2.80×	4.33	5.00×	6.49
	DFlash	5.37×	6.74	6.31×	8.04	5.82×	7.38	5.27×	6.67	4.76×	6.16	5.38×	7.03	2.86×	4.47	5.11×	6.64
	D-PACE*	5.61×	7.01	6.52×	8.26	5.88×	7.45	5.28×	6.68	4.82×	6.24	5.36×	7.01	2.85×	4.45	5.19×	6.73
	D-PACE	5.89×	7.36	6.85×	8.61	6.09×	7.76	5.35×	6.76	4.99×	6.42	5.49×	7.21	2.89×	4.51	5.36×	6.95
	PC-VOT	**6.28×**	**7.81**	**7.54×**	**9.37**	**6.34×**	**8.06**	**5.54×**	**7.02**	**5.28×**	**6.77**	**5.72×**	**7.48**	**3.09×**	**4.81**	**5.68×**	**7.33**
Q3-4B-T	DFlash*	2.96×	3.63	3.04×	3.76	3.02×	3.73	2.90×	3.87	2.76×	3.59	2.70×	3.49	2.41×	3.11	2.83×	3.60
	DFlash	3.80×	4.69	3.97×	4.93	3.89×	4.83	3.53×	4.54	3.10×	3.97	2.89×	3.70	2.58×	3.40	3.39×	4.29
	PC-VOT	**5.19×**	**6.42**	**5.34×**	**6.64**	**5.06×**	**6.26**	**4.68×**	**5.76**	**4.51×**	**5.53**	**4.27×**	**5.24**	**2.98×**	**4.10**	**4.58×**	**5.71**
Q3-8B-T	DFlash*	2.82×	3.50	2.98×	3.73	3.02×	3.81	2.70×	3.35	2.59×	3.22	2.58×	3.18	2.29×	2.97	2.71×	3.39
	DFlash	3.67×	4.55	4.23×	5.35	3.97×	5.06	3.53×	4.41	3.18×	3.95	2.96×	3.67	2.51×	3.42	3.44×	4.34
	PC-VOT	**5.01×**	**6.24**	**5.42×**	**6.86**	**5.01×**	**6.37**	**4.43×**	**5.51**	**4.36×**	**5.42**	**4.17×**	**5.16**	**3.05×**	**4.46**	**4.49×**	**5.72**
$T=1$		SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ
Q3-4B	DFlash*	4.66×	5.94	5.05×	6.65	3.71×	4.84	4.72×	6.01	4.41×	5.61	4.90×	6.64	2.64×	4.00	4.30×	5.67
	DFlash	4.73×	6.03	5.11×	6.72	3.77×	4.91	4.77×	6.07	4.46×	5.67	5.01×	6.77	2.67×	4.05	4.36×	5.75
	D-PACE*	4.91×	6.28	5.25×	6.89	3.78×	4.93	4.80×	6.11	4.49×	5.71	4.97×	6.72	2.66×	4.03	4.41×	5.81
	D-PACE	5.16×	6.59	5.49×	7.18	3.94×	5.13	4.85×	6.18	4.65×	5.88	5.09×	6.91	2.69×	4.09	4.55×	5.99
	PC-VOT	**5.47×**	**6.97**	**5.86×**	**7.59**	**4.11×**	**5.36**	**4.99×**	**6.37**	**4.81×**	**6.10**	**5.25×**	**7.13**	**2.84×**	**4.31**	**4.76×**	**6.26**
Q3-4B-T	DFlash*	2.69×	3.30	2.85×	3.51	2.76×	3.43	2.62×	3.21	2.52×	3.09	2.46×	3.01	2.20×	2.80	2.59×	3.19
	DFlash	3.42×	4.29	3.57×	4.46	3.24×	4.11	3.06×	3.77	2.66×	3.26	2.58×	3.17	2.27×	2.90	2.97×	3.71
	PC-VOT	**4.19×**	**5.27**	**4.33×**	**5.46**	**3.82×**	**4.88**	**3.92×**	**4.82**	**3.71×**	**4.55**	**3.53×**	**4.39**	**2.60×**	**3.28**	**3.73×**	**4.66**
Q3-8B-T	DFlash*	2.70×	3.36	2.80×	3.50	2.73×	3.45	2.47×	3.08	2.34×	2.90	2.32×	2.87	2.08×	2.66	2.49×	3.12
	DFlash	3.28×	4.11	3.85×	4.87	3.42×	4.39	3.06×	3.84	2.65×	3.31	2.47×	3.09	2.18×	2.85	2.99×	3.78
	PC-VOT	**4.31×**	**5.41**	**4.38×**	**5.59**	**3.95×**	**5.09**	**3.62×**	**4.49**	**3.37×**	**4.19**	**3.28×**	**4.12**	**2.55×**	**3.48**	**3.64×**	**4.62**

Table 6: Ablation study of inference batch size ($B \in \{4, 8\}$) in PC-VOT. We report speedup ratio (SR) and acceptance length (τ) on Qwen3-8B with temperature $T \in \{0, 1\}$.

Batch B	Method	MATH						CODE						CHAT		*Avg.*	
		GSM8K		MATH-500		AIME25		HumanEval		MBPP		LCB		MT-Bench			
$T=0$		SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ
4	D-PACE	5.48×	7.51	6.14×	8.98	5.48×	7.44	5.00×	6.70	4.65×	6.44	5.31×	7.46	2.65×	4.41	4.96×	6.99
	PC-VOT	**5.87×**	**7.94**	**6.83×**	**9.78**	**5.93×**	**7.97**	**5.09×**	**6.84**	**4.87×**	**6.66**	**5.49×**	**7.69**	**2.86×**	**4.78**	**5.28×**	**7.38**
8	D-PACE	4.54×	7.52	5.17×	9.00	4.25×	7.47	4.11×	6.71	4.02×	6.43	4.17×	7.45	2.10×	4.42	4.05×	7.00
	PC-VOT	**4.85×**	**7.96**	**5.92×**	**9.75**	**4.91×**	**7.97**	**4.39×**	**6.83**	**4.28×**	**6.69**	**4.53×**	**7.69**	**2.36×**	**4.76**	**4.46×**	**7.38**
$T=1$		SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ	SR	τ
4	D-PACE	3.73×	6.71	3.63×	6.95	2.42×	5.14	3.16×	5.64	3.07×	5.59	3.52×	7.00	1.88×	3.83	3.06×	5.84
	PC-VOT	**4.22×**	**7.07**	**4.27×**	**7.54**	**2.86×**	**5.52**	**3.51×**	**5.82**	**3.40×**	**5.77**	**3.93×**	**7.22**	**2.17×**	**4.13**	**3.48×**	**6.15**
8	D-PACE	3.08×	6.71	3.01×	6.94	1.65×	5.12	2.48×	5.64	2.65×	5.59	2.72×	6.99	1.52×	3.84	2.44×	5.83
	PC-VOT	**3.37×**	**7.06**	**3.59×**	**7.55**	**2.33×**	**5.53**	**2.91×**	**5.80**	**2.88×**	**5.77**	**3.16×**	**7.21**	**1.79×**	**4.14**	**2.86×**	**6.15**

officially released checkpoints, whereas DFlash and D-PACE denote models finetuned under our training framework. As shown in Table 5, continued training improves DFlash by 11.7% in speedup ratio and 12.0% in acceptance length on average; it improves D-PACE by 3.2% in both metrics. More importantly, PC-VOT further outperforms the retrained DFlash by 20.8% in speedup ratio and 20.3% in acceptance length, and the retrained D-PACE by 5.3% and 5.0%, respectively. These results demonstrate that the improvements of PC-VOT persist against strengthened baselines. Under our training framework, all objective variants share the same initialization, training data, optimization schedule, and compute budget, differing only in the training objective.

E.2 Additional Ablation Studies

Ablation on Inference Batch Size. Table 6 evaluates the effect of inference batch size on D-PACE

and PC-VOT on Qwen3-8B. At $T = 0$, PC-VOT improves the average speedup ratio over D-PACE by 6.45% and 10.12% for batch sizes 4 and 8, respectively. At $T = 1$, the corresponding improvements are 13.73% and 17.06%. Although both methods exhibit lower average speedup ratios as the batch size increases, PC-VOT consistently outperforms D-PACE, with larger relative gains at batch size 8 under both temperatures. These results demonstrate that PC-VOT remains effective as the inference batch size increases, exhibiting robust scalability in batched inference settings.

E.3 Analysis on Training Overhead

PC-VOT introduces additional training cost mainly from the Monte Carlo approximation of the verifier-induced feasible set and the corresponding transport computation. The remaining optimization follows the standard drafter-training pipeline and introduces little additional overhead. PC-VOT requires approximately 518 and 943 GPU-hours to train the 4B and 8B models, respectively, on 8 NVIDIA RTX PRO 6000 GPUs. This additional cost is incurred only once during draft-model training and can be amortized over subsequent deployment. Importantly, PC-VOT does not introduce extra computation into the speculative-decoding pipeline at inference.

The training overhead should therefore be viewed as a one-time investment for improving recurring inference efficiency. In high-throughput deployments, even a modest reduction in decoding cost can accumulate into substantial GPU savings as the total number of served tokens increases.

For large-scale services, an efficiency improvement of around 6% can translate into tens of thousands of GPU-hours saved per day. The cumulative savings can substantially exceed the one-time training cost. Moreover, PC-VOT preserves the target model's output distribution through lossless speculative decoding, improving efficiency without sacrificing generation quality.