DIRECTLY MAXIMIZING ON-POLICY EXPECTED ACCEPTED LENGTH FOR SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative decoding accelerates large language model inference by using a lightweight draft model to propose tokens for verification by a target model. Draft training commonly optimizes token-level distribution alignment, but this does not directly optimize the length of the accepted prefix: once a draft token is rejected, all subsequent draft tokens are discarded. We propose FReDraft, an on-policy training method that directly maximizes the expected accepted length of drafts sampled from the current draft model. This objective accounts for first-rejection truncation and induces prefix-aware credit assignment, weighting each token by its contribution to the acceptance of the current and subsequent tokens. To optimize the objective efficiently, we analytically marginalize token-level acceptance rate, using the target model's vocabulary distribution to reduce gradient variance. We further redistribute part of this distribution-level gradient into an explicit penalty for sampled tokens whose probabilities are overestimated by the draft model, while preserving the expected gradient. Experiments across multiple benchmarks and target models show that FReDraft improves accepted length and decoding speedup over strong baselines, achieving faster on-policy training convergence. Code is available at `https://anonymous.4open.science/r/Code-FReDraft-A78E`.

1 INTRODUCTION

Large language models (LLMs) have achieved strong performance across a wide range of generation tasks, including reasoning, coding and agentic tasks (Guo et al., 2025; Yang et al., 2025; Team et al., 2026b). But their autoregressive decoding process remains time-consuming because each new token requires one LLM forward pass (Pope et al., 2023; Chen et al., 2026b). Speculative decoding accelerates LLM inference through a draft-verification paradigm (Figure 1 (a)): a lightweight draft model proposes multiple future tokens, and the target LLM verifies them in one forward pass with a rejection-sampling rule that preserves the target distribution (Leviathan et al., 2023; Chen et al., 2023). Since acceleration comes from accepting multiple draft tokens per target-model call, the speedup depends heavily on the acceptance rate of tokens in the draft (Zhang et al., 2025; Zhao et al., 2025).

To train the draft model for higher acceptance rate, a typical approach is supervised finetuning (SFT) on target-generated sequences (Li et al., 2025; Chen et al., 2026a; Li et al., 2026a), minimizing token-level cross-entropy (CE) loss independently. However, token verification within a draft is not independent; instead, it exhibits "first-rejection truncation" property (Hu et al., 2025; Gu et al., 2026). Verification proceeds sequentially from the first draft token, and once a token is rejected, all subsequent tokens are discarded. Consequently, a later token contributes to speedup only if its entire prefix survives verification. Inspired by this, some works explore verification-aware draft model training (An et al., 2026). For example, LK-Losses (Samarin et al., 2026) maximizes exponential decay (ED) weighted token-level log acceptance rate, VSD (Zou et al., 2026) optimizes a variational objective as lower bound of draft tree utility, whereas D-PACE (Wu et al., 2026) proposes to maximize the summed cumulative draft confidence at each position in target-generated sequences.

These verification-aware methods achieve draft model enhancement over naive SFT, but they actually optimize a metric of sequences under target-distribution as a surrogate, which does not match "first-rejection truncation" in practical verification (Figure 1 (b)). First, draft model is trained on sequences following target distribution (or hybrid of draft and target distribution), whereas evaluated on drafts proposed under its own policy, leading to exposure bias. Second, the optimized metrics do not

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.

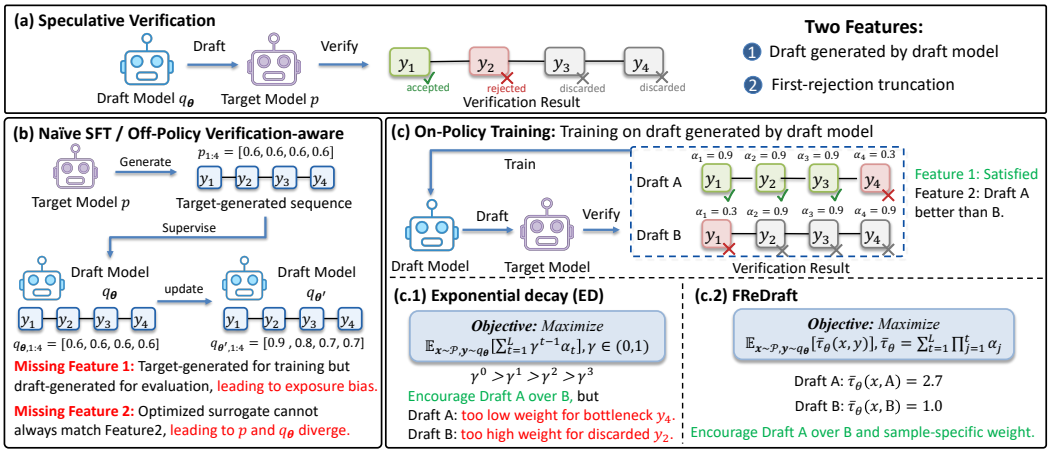

Figure 1: FReDraft trains on self-generated drafts and accounts for first-rejection truncation. Here α means token-level acceptance rate and $\bar{\tau}$ means expected accepted length.

necessarily mean later "first-rejection". For example, optimizing cumulative draft confidence makes draft model diverge from target severely when target model itself is uncertain (See Appendix C.6 for systematic analysis). Recently, Draft-OPD (Lei et al., 2026) proposes on-policy paradigm to train the draft model on its own generated drafts, eliminating exposure bias (Hu et al., 2026) (Figure 1 (c)). However, it applies heuristically designed forward and reverse KL weighted by static ED. Such static schedules cannot capture actual importance of tokens, which depends on decoding context, and target-to-draft probability ratios encountered during verification. As a result, a token constituting the verification bottleneck may receive insufficient weight, whereas tokens following prefixes with low survival probability may be overemphasized, leading to inefficient allocation of the training signal.

Draft accepted length is a metric directly related to speedup ratio of speculative decoding (Sadhukhan et al., 2025; Zhang et al., 2025; An et al., 2026). Given a draft generated by draft model, its expected accepted length can be calculated precisely by summing the prefix survival rate of token at each position. Therefore, this metric intrinsically contains influence of verification result from earlier tokens to later ones (Figure 1 (c.2)), matching first-rejection truncation property (Hu et al., 2026; Wang et al., 2025). Inspired by the advantage of expected accepted length, we propose **FReDraft**, a draft model training method that directly maximizes the expected accepted length of draft generated by draft model itself, to model first-rejection truncation. The objective naturally matches on-policy paradigm, and derived gradient captures prefix survival and influence of earlier tokens to later tokens in practical verification. But directly optimizing draft model with this gradient remains inefficient. In practice, the gradient requires Monte Carlo (MC) sampling to estimate, from which we identify two factors that make optimization inefficient: First, only target model probabilities for sampled draft tokens can be used to estimate the gradient, leading to high gradient estimation variance and the vocabulary distribution output by target model during verification discarded. Second, over-sampled tokens, for which the draft model assigns higher probability than target model, receive zero gradient. Since on-policy training data is generated by the draft model itself, such over-sampled tokens are common and become ineffective in training. To tackle these difficulties, FReDraft proposes to marginalize single token acceptance rate expectation, which reduces the variance of gradient estimation and enables leveraging of target model vocabulary distribution to supervise draft model training. FReDraft also introduces penalty for over-sampled tokens, allowing frequently-appearing over-sampled tokens to receive non-zero gradient. We summarize the contributions as follows:

- We formulate draft training as on-policy maximization of expected accepted length, aligning the objective with first-rejection truncation and deriving token-level credit assignment from each token's contribution to prefix survival.
- We develop token-level acceptance marginalization to reduce gradient variance with target vocabulary distribution and redistribute part of the resulting gradient into an explicit over-sampled token penalty, enabling efficient optimization while preserving the expected gradient.
- Experiments across multiple target models and benchmarks show that FReDraft improves accepted draft length and decoding speedup over strong baselines, including Draft-OPD. It also exhibits faster convergence during on-policy training.

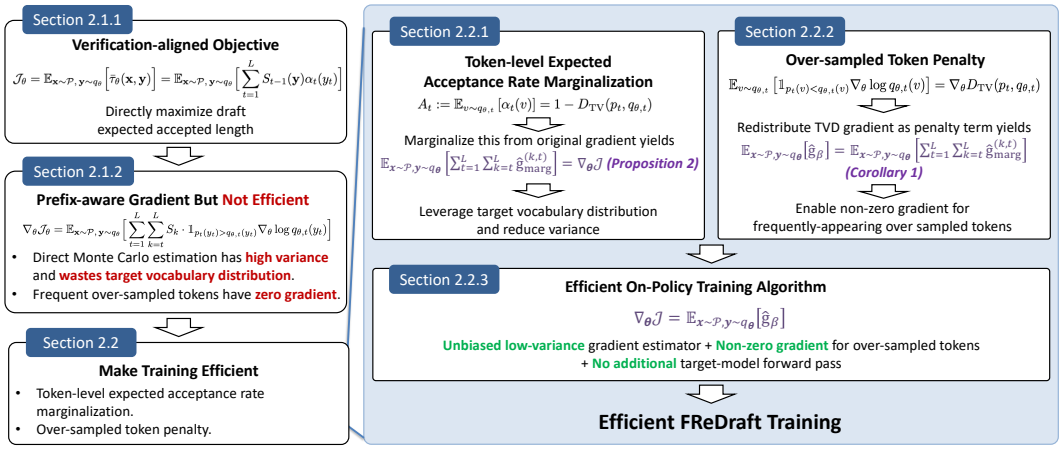

Figure 2: Overview of how FReDraft preserves the verification-aligned objective while constructing an efficient on-policy gradient estimator.

2 FReDraft

Section 2 introduces FReDraft, a draft model on-policy training method that maximizes self-generated drafts' expected accepted length, enabling first-rejection-truncation-aware training signal allocation. Section 2.1 formulates the expected accepted length objective, characterizes its gradient and identifies two factors resulting in optimization inefficiency. To deal with the two factors, Section 2.2 develops optimization techniques for efficient training. We show the overall outline of Section 2 in Figure 2.

2.1 Verification-Aligned Objective

2.1.1 Objective Formulation

Let p be the fixed target model and q_θ the trainable draft model over vocabulary $\mathcal{V}$, given a decoding context $\mathbf{x} \sim \mathcal{P}$ (referring to Draft-OPD (Lei et al., 2026), $\mathcal{P}$ is the distribution of decoding context that will be visited by p and q_θ when conducting speculative decoding), speculative decoding first employs q_θ to propose a length-L ($L > 1$) draft block $\mathbf{y} = (y_1, \ldots, y_L) \sim q_\theta$. Define $h_t = (\mathbf{x}, \mathbf{y}_{<t})$ and $p_t(v) = p(v \mid h_t)$, $q_{\theta,t}(v) = q_\theta(v \mid h_t)$ ($q_{\theta,t}(v) = q_\theta(v \mid \mathbf{x})$ for parallel draft model e.g. DFlash (Chen et al., 2026a)). Then during verification, token y_t is accepted with probability

$$\alpha_t(y_t) = \min\left\{1, p_t(y_t)/q_{\theta,t}(y_t)\right\}. \tag{1}$$

Verification stops at the first rejection, so it evaluates an accepted prefix rather than independent token predictions (Leviathan et al., 2023; Chen et al., 2023). Consequently, token y_t contributes only when every preceding token survives. To model such first-rejection truncation property of verification, we learn q_θ by directly maximizing the expected accepted length of the drafts generated by q_θ.

Definition 1 (Verification-aligned expected accepted-length objective)**.** *Given a decoding context $\mathbf{x} \sim \mathcal{P}$ and a length-L draft $\mathbf{y} = (y_1, \ldots, y_L) \sim q_\theta$, the verification-aligned objective is*

$$\mathcal{J}_\theta = \mathbb{E}_{\mathbf{x}\sim\mathcal{P},\,\mathbf{y}\sim q_\theta}\left[\bar{\tau}_\theta(\mathbf{x},\mathbf{y})\right] = \mathbb{E}_{\mathbf{x}\sim\mathcal{P},\,\mathbf{y}\sim q_\theta}\left[\sum\nolimits_{t=1}^{L} S_{t-1}(\mathbf{y})\alpha_t(y_t)\right], \tag{2}$$

where $\bar{\tau}_\theta(\mathbf{x},\mathbf{y}) = \sum_{t=1}^{L}\prod_{j=1}^{t}\alpha_j(y_j)$ is the expected accepted length of $\mathbf{y}$ (see Appendix C.1 for derivation), and $S_{t-1}(\mathbf{y}) = \prod_{j=1}^{t-1}\alpha_j(y_j)$, with $S_0(\mathbf{y}) = 1$, is the probability that verification reaches position t (i.e. prefix survival rate of prefix $\mathbf{y}_{<t}$).

Definition 1 defines the expected accepted length maximization objective and further characterizes it as token-level acceptance rate maximization with prefix-survival S_{t-1} as the coefficient, rather than treating all token positions independently. This objective aligns with verification process of speculative decoding in terms of following two aspects: (1) The verified drafts are generated by the draft model q_θ itself, matching the real-world scenario and making the objective naturally suitable for on-policy training. (2) The maximized $\bar{\tau}_\theta(\mathbf{x},\mathbf{y})$ intrinsically captures the influence of earlier token verification result to later tokens, thus matching the first-rejection truncation property of verification.

2.1.2 GRADIENT THEORETICAL CHARACTERIZATION

Definition 1 provides a verification-aligned objective based on expected accepted length. We next identify the optimization direction induced from the objective by analyzing its gradient.

Proposition 1 (Prefix-aware gradient). *Let $G_t = \mathbb{1}_{p_t(y_t)>q_{\theta,t}(y_t)} \nabla_\theta \log q_{\theta,t}(y_t)$ be the gradient for token y_t at position t and $\widehat{\mathbf{g}}_{\text{dir}}^{(k,t)} = S_k G_t$. The gradient of the objective in Equation 2 is*

$$\nabla_\theta \mathcal{J}_\theta = \mathbb{E}_{\mathbf{x}\sim\mathcal{P},\,\mathbf{y}\sim q_\theta}\Big[\sum\nolimits_{t=1}^{L}\sum\nolimits_{k=t}^{L} \widehat{\mathbf{g}}_{\text{dir}}^{(k,t)}\Big] = \mathbb{E}_{\mathbf{x}\sim\mathcal{P},\,\mathbf{y}\sim q_\theta}\Big[\sum\nolimits_{t=1}^{L}\Big(\sum\nolimits_{k=t}^{L} S_k\Big) G_t\Big]. \quad (3)$$

In practice, $\mathcal{P}$ depends on q_θ but $\nabla_\theta \mathcal{P}$ is too hard to estimate. Therefore, referring to Draft-OPD (Lei et al., 2026), Equation 3 omits $\nabla_\theta \mathcal{P}$, and what it actually optimizes is "expected accepted length of draft-model-generated draft for given decoding context $\mathbf{x}$" (See Appendix C.2 for derivation). Proposition 1 shows that the objective induces prefix-aware credit assignment. The weight $\sum_{k=t}^{L} S_k(\mathbf{y})$ for G_t aggregates the survival probabilities of all prefixes extending through position t whose survival depends on token y_t. Therefore, the optimization weight of a token reflects not only its own acceptance but also its influence on subsequent verification outcomes, matching the first-rejection mechanism.

However, directly training the draft model as Equation 3 remains inefficient, where we identify two key factors. **First**, the trajectory-dependent weight $\sum_{k=t}^{L} S_k(\mathbf{y})$ must be estimated through MC sampling and only target probabilities on sampled tokens can be used to estimate it. This leads to high variance and the target vocabulary distribution discarded during on-policy training. **Second**, over-sampled tokens satisfying $p_t(y_t) < q_{\theta,t}(y_t)$ receive zero gradient due to the indicator $\mathbb{1}_{p_t(y_t)>q_{\theta,t}(y_t)}$. Moreover, such tokens frequently appear in training because the training data is generated by q_θ.

2.2 EFFICIENT OPTIMIZATION

To deal with above two factors, as summarized in Figure 2, we develop two techniques, i.e. token-level acceptance rate marginalization (Section 2.2.1) and over-sampled-token penalty (Section 2.2.2), to make the optimization more efficient, while preserving the expected gradient of original objective.

2.2.1 TOKEN-LEVEL EXPECTED ACCEPTANCE RATE MARGINALIZATION

The expectation of gradient in Equation 3 requires Monte Carlo sampling to estimate, introducing high variance. It also leads to that only target probabilities for sampled tokens can be used to estimate the gradient, leaving target vocabulary distribution discarded. Actually, the token-level expected acceptance rate can be computed precisely with no need for MC sampling. Specifically, for a fixed decoding context $\mathbf{x}$ and draft prefix $\mathbf{y}_{<t}$, the expected acceptance rate of the next token is given by

$$A_t := \mathbb{E}_{v\sim q_{\theta,t}}\left[\alpha_t(v)\right] = \sum\nolimits_{v\in\mathcal{V}} \min\left\{p_t(v), q_{\theta,t}(v)\right\} = 1 - D_{\text{TV}}(p_t, q_{\theta,t}). \quad (4)$$

Here $D_{\text{TV}}(p_t, q_{\theta,t}) = \frac{1}{2}\sum_{v\in\mathcal{V}} |p_t(v) - q_{\theta,t}(v)|$ is total variation distance (TVD) (Van Handel, 2014). Unlike sampled acceptance event, A_t can be computed exactly according to vocabulary distribution of draft and target model, produced during verification. Marginalizing this expected acceptance rate from Equation 3 transforms part of the Monte-Carlo estimated gradient into precise computation, thus yielding a lower-variance estimator of the gradient as follows (See Appendix C.3 for full derivation):

Proposition 2 (Expected acceptance rate marginalization). *Define the gradient estimator*

$$\widehat{\mathbf{g}}_{\text{marg}}^{(k,t)} := \begin{cases} -S_{t-1}(\mathbf{y})\nabla_\theta D_{\text{TV}}(p_t, q_{\theta,t}), & k = t, \\ A_k S_{k-1} G_t, & k > t. \end{cases} \quad (5)$$

Then $\mathbb{E}_{\mathbf{x}\sim\mathcal{P},\mathbf{y}\sim q_\theta}[\sum_{t=1}^{L}\sum_{k=t}^{L} \widehat{\mathbf{g}}_{\text{marg}}^{(k,t)}] = \mathbb{E}_{\mathbf{x}\sim\mathcal{P},\mathbf{y}\sim q_\theta}[\sum_{t=1}^{L}\sum_{k=t}^{L} \widehat{\mathbf{g}}_{\text{dir}}^{(k,t)}] = \nabla_\theta \mathcal{J}_\theta$. *Under the same Monte Carlo sampling scheme and any fixed direction $\mathbf{a}$ of compatible dimension,* $\text{Var}(\mathbf{a}^\top \widehat{\mathbf{g}}_{\text{marg}}^{(k,t)} | \mathbf{x}, \mathbf{y}_{<t}) \le \text{Var}(\mathbf{a}^\top \widehat{\mathbf{g}}_{\text{dir}}^{(k,t)} | \mathbf{x}, \mathbf{y}_{<t}), \forall 1 \le t \le k \le L$, *where* $\text{Var}(\cdot)$ *means variance.*

Proposition 2 analytically marginalizes the terminal-token sampling in each current and future prefix-credit contribution while retaining trajectory-dependent credit assignment through the draft prefix. Equation 5 consists of two terms: The term with G_t is gradient for sampled token y_t, while the term with $\nabla_\theta D_{\text{TV}}(p_t, q_{\theta,t})$ is gradient over the vocabulary. Equation 5 leverages the target

vocabulary distribution available during verification to compute $\nabla_\theta D_{\mathrm{TV}}(p_t, q_{\theta,t})$. Compared with KL, the gradient of TVD w.r.t logits does not shrink when two distributions are close, which has been associated with better optimization efficiency in this case (Ji et al., 2023; Samarin et al., 2026) (See Appendix C.5 for detailed discussion). Notably, variance reduction is for $\widehat{\mathbf{g}}^{(k,t)}_{\mathrm{marg}}$ rather than the summed estimator of $\nabla_\theta \mathcal{J}_\theta$, which also depends on covariance among draft positions and difficult to characterize theoretically. We provide empirical evidence for total variance reduction in Section 4.5.

2.2.2 Over-Sampled Token Penalty

Token-level expected acceptance rate marginalization introduces dense distribution supervision over the vocabulary, but the gradient term for sampled tokens in Equation 5 remains gated by $\mathbb{1}_{p_t(y_t)>q_{\theta,t}(y_t)}$. Consequently, over-sampled tokens satisfying $q_{\theta,t}(y_t) > p_t(y_t)$ receive no direct sampled-token correction, although they occur frequently under on-policy sampling. One possible remedy is sampling from a mixture of the target and draft distributions to reduce the ratio of over-sampled tokens and apply importance weighting (Gu et al., 2024) for unbiased estimation. However, this requires additional auto-regressive sampling from the target model and undermines efficiency. Instead, we redistribute the dense TVD gradient $\nabla_\theta D_{\mathrm{TV}}(p_t, q_{\theta,t})$ available in Equation 5. Specifically, define $\widetilde{G}_t = \mathbb{1}_{p_t(y_t)<q_{\theta,t}(y_t)} \nabla_\theta \log q_{\theta,t}(y_t)$, then almost everywhere, it admits:

$$\mathbb{E}_{y_t \sim q_{\theta,t}}[\widetilde{G}_t] = \nabla_\theta D_{\mathrm{TV}}(p_t, q_{\theta,t}). \tag{6}$$

The two sides of Equation 6 have the same conditional expectation given the decoding context $\mathbf{x}$ and draft prefix $\mathbf{y}_{<t}$. We can therefore replace a controlled fraction of the dense TVD term by its sampled representation without changing the expected gradient as follows (see Appendix C.4 for full proof):

Corollary 1 (Expectation-preserving over-sampled-token correction)**.** *For any $\beta \in [0, 1]$, define*

$$\widehat{\mathbf{g}}_\beta := \sum\nolimits_{t=1}^{L} \sum\nolimits_{k=t}^{L} \widehat{\mathbf{g}}^{(k,t)}_{\mathrm{marg}} + \beta \sum\nolimits_{t=1}^{L} S_{t-1}(\mathbf{y}) \Big[\nabla_\theta D_{\mathrm{TV}}(p_t, q_{\theta,t}) - \widetilde{G}_t \Big]. \tag{7}$$

Then $\mathbb{E}_{\mathbf{x}\sim\mathcal{P},\mathbf{y}\sim q_\theta}[\widehat{\mathbf{g}}_\beta] = \mathbb{E}_{\mathbf{x}\sim\mathcal{P},\mathbf{y}\sim q_\theta}[\sum_{t=1}^{L}\sum_{k=t}^{L} \widehat{\mathbf{g}}^{(k,t)}_{\mathrm{marg}}] = \nabla_\theta \mathcal{J}_\theta$.

Corollary 1 redistributes existing, rather than introduces additional, gradient information. Expanding Equation 7 gives the position weight $w_t^{(\beta)}(\mathbf{y}) = [\sum_{k=t+1}^{L} A_k S_{k-1}(\mathbf{y})]\mathbb{1}_{p_t(y_t)>q_{\theta,t}(y_t)} - [\beta S_{t-1}(\mathbf{y})]\mathbb{1}_{p_t(y_t)<q_{\theta,t}(y_t)}$ for gradient of sampled token $\nabla_\theta \log q_{\theta,t}(y_t)$. Intuitively, such a weight encourages the under-sampled tokens with $p_t(y_t) > q_{\theta,t}(y_t)$ while penalizing over-sampled tokens with $p_t(y_t) < q_{\theta,t}(y_t)$. The gradient of dense TVD term $\nabla_\theta D_{\mathrm{TV}}(p_t, q_{\theta,t})$ is weighted by $-(1-\beta)S_{t-1}(\mathbf{y})$. Thus, $\beta = 0$ recovers Equation 5, whereas increasing β transfers more of the dense TVD gradient to explicit sampled-token penalty for over-sampled tokens, enabling them to receive non-zero gradient. This transformation does not introduce additional information for training. However, it weakens the stochastic optimization fluctuation of long-tailed unsampled tokens coupled with vocabulary TVD, but preserves the gradient of over-sampled tokens, which are usually bottleneck for draft accepted length. β works here as a hyperparameter to balance the gradient for over-sampled tokens and vocabulary. See Appendix C.5 for detailed discussion about effectiveness of penalty term.

2.2.3 On-Policy Optimization

The objective in Equation 2 is inherently on-policy because its expectation is taken over drafts generated by the current draft model q_θ. Specifically, for a batch of prompts sampled from training dataset, we conduct the standard speculative decoding for them, during which we collect a series of draft blocks $\mathbf{y}$ for each prompt, and corresponding target model distribution p and draft model distribution q_θ. Then we estimate the gradient with the collected information according to Equation 7. FReDraft reuses the distributions already computed during verification and requires neither auto-regressive target sampling nor additional target forward pass. For $\nabla_\theta D_{\mathrm{TV}}$,

Algorithm 1 On-Policy Optimization with FReDraft

Require: Prompt set $\mathcal{P}_p$, frozen target model p, initialized draft model q_θ, coefficient β, learning rate η
Ensure: Optimized draft model q_θ
1: **while** not converged **do**
2: Sample a prompt batch from $\mathcal{P}_p$
3: Run speculative decoding with q_θ and the frozen p
4: Collect drafts and their target/draft distributions
5: Compute the batch estimator $\widehat{\mathbf{g}}_\beta$ with Equation 7
6: $\theta \leftarrow \theta + \eta \widehat{\mathbf{g}}_\beta$
7: **end while**
8: **return** q_θ

we adopt the widely used top-K target tokens in OPD to estimate it to save memory in practice (Theoretical and empirical results show that top-K introduces negligible error compared with full vocabulary. See Appendix B.2 for details) (Fu et al., 2026). Compared with existing on-policy method (e.g. Draft-OPD), FReDraft requires no extra target model forward pass and achieves comparable per-training-step efficiency (See Appendix E.5 for wall-clock time comparison). We summarize the proposed on-policy algorithm in Algorithm 1. To make the on-policy training more efficient and stable, we refer to Draft-OPD (Lei et al., 2026) and first warm up the draft model with SFT on target-generated sequences. Then we further finetune it as Algorithm 1.

3 Comparison with Representative Draft Training Methods

To position FReDraft, Table 1 compares it with representative draft-model training methods. Naive SFT (Li et al., 2026a; 2025; Chen et al., 2026a) and off-policy verification-aware method (e.g. LK-Losses (Samarin et al., 2026) and D-PACE (Wu et al., 2026)) train the draft model on target-generated sequences. The difference is that verification-aware methods introduce verification information into training objective. For example, LK-Losses optimizes token-level log acceptance rate, and D-PACE maximizes summed cumulative draft confidence on target-generated sequences (see appendix C.6 for more detailed discussion about D-PACE). However, objective of these methods does not come from practical verification and cannot necessarily enable later "first-rejection" in practical verification, due to exposure bias, and the gap between optimized surrogate and draft accepted length. Draft-OPD (Lei et al., 2026) adopts on-policy distillation (Li et al., 2026b) to mitigate exposure bias. But it applies ED to weight token-level KL, which can lead to inefficient training signal allocation. In contrast, FReDraft directly maximizes the expected accepted length of drafts generated by the draft model, achieving on-policy training. The derived credit assignment evaluates prefix survival and influence on subsequent verification outcomes in practical verification, matching the first-rejection truncation property of verification process. See Appendix A for detailed discussion about more related works.

Table 1: Comparison of draft-model training methods. V-PS and V-SO denote practical-Verification-defined Prefix-Survival and influence on Subsequent verification Outcomes in credit assignment.

Method	Objective	Data	Credit	V-PS	V-SO
SFT	Token-level cross entropy	Off-policy	Uniform / ED	×	×
D-PACE	Summed cumulative draft confidence	Off-policy	Adaptive	×	×
LK-Losses	Token-level log acceptance rate	Off-policy	ED	×	×
Draft-OPD	Forward / Reverse KL divergence	On-policy	ED	×	×
FReDraft	Expected accepted length	On-policy	Adaptive	✓	✓

4 Experiments

4.1 Experiment Setup

Baselines. We use two draft model architectures: **Eagle-3** (Li et al., 2025) and **DFlash** (Chen et al., 2026a). For both architectures, we implement SFT to train them on target-generated sequences. Based on DFlash architecture (our experimental results show that DFlash performs better than Eagle-3), we implement recent SOTA representative methods with public DFlash implementations as baselines: off-policy verification-aware **D-PACE** (Wu et al., 2026), **LK-Losses** (Samarin et al., 2026), and on-policy **Draft-OPD** (Lei et al., 2026). See Appendix D.1 for implementation details of baselines.

Benchmarks. We include the following benchmarks: (1) Mathematics: **GSM8K** (Cobbe et al., 2021), **MATH-500** (Hendrycks et al., 2021) and **AIME25** (Mathematical Association of America, 2025); (2) Coding: **MBPP** (Austin et al., 2021), **HumanEval** (Chen et al., 2021) and **SWE-Lite** (Jimenez et al., 2024); (3) Multi-turn Chat: **MT-Bench** (Zheng et al., 2023). We report accepted length (τ) and speedup ratio (SR) for benchmarks. See Appendix D.3 for details of benchmarks.

Implementations. We use **Qwen3-4B / -8B** (Yang et al., 2025) as target model (See Appendix E.4 for generalization to other targets). For SFT stage, we use the **Qwen3-4B-Instruct-100K** dataset released by DFlash (Chen et al., 2026a) for draft model on Qwen3-4B. We leverage Qwen3-8B

Table 2: Decoding speedup ratio and average acceptance length (τ) on Qwen3 models.

Model	Method	Speedup Ratio (SR)							Mean	
		Math			Code			Chat	SR	τ
		GSM8K	MATH-500	AIME25	MBPP	HumanEval	SWE-Lite	MT-Bench		
Temperature = 0										
Q3-4B	EAGLE-3	1.99×	1.83×	1.79×	1.78×	1.84×	1.62×	1.74×	1.80×	5.32
	DFlash	3.59×	4.04×	3.43×	3.23×	3.45×	2.14×	2.02×	3.13×	5.73
	D-PACE	3.74×	4.22×	3.67×	3.30×	3.33×	2.20×	1.97×	3.20×	5.83
	LK-Losses	3.92×	4.48×	3.88×	3.35×	3.52×	2.22×	2.00×	3.34×	6.01
	Draft-OPD	4.24×	4.74×	4.00×	3.42×	3.52×	2.16×	2.06×	3.45×	6.23
	FReDraft	**4.62**×	**5.13**×	**4.43**×	**3.72**×	**3.96**×	**2.43**×	**2.25**×	**3.79**×	**6.94**
Q3-8B	EAGLE-3	1.94×	1.81×	1.79×	1.69×	1.89×	1.73×	1.63×	1.78×	5.20
	DFlash	4.62×	5.50×	4.93×	4.20×	4.82×	2.69×	2.52×	4.18×	5.67
	D-PACE	4.87×	5.76×	5.12×	4.33×	4.80×	2.72×	2.43×	4.29×	5.72
	LK-Losses	5.14×	6.06×	5.38×	4.52×	4.96×	2.88×	2.69×	4.52×	6.03
	Draft-OPD	5.40×	6.36×	5.62×	4.78×	5.17×	3.06×	2.87×	4.75×	6.34
	FReDraft	**5.82**×	**6.82**×	**6.01**×	**5.14**×	**5.52**×	**3.23**×	**3.08**×	**5.09**×	**7.08**
Temperature = 1.0										
Q3-4B	EAGLE-3	1.89×	1.75×	1.64×	1.69×	1.74×	1.48×	1.70×	1.70×	5.01
	DFlash	3.31×	3.24×	2.44×	3.03×	3.25×	1.90×	1.90×	2.72×	5.34
	D-PACE	3.49×	3.37×	2.52×	2.97×	3.13×	1.93×	1.82×	2.75×	5.42
	LK-Losses	3.60×	3.52×	2.60×	3.08×	3.23×	1.95×	1.97×	2.85×	5.58
	Draft-OPD	3.81×	3.82×	2.88×	3.16×	3.27×	1.91×	1.94×	2.97×	5.70
	FReDraft	**4.14**×	**4.21**×	**3.19**×	**3.43**×	**3.65**×	**2.11**×	**2.17**×	**3.27**×	**6.01**
Q3-8B	EAGLE-3	1.87×	1.72×	1.63×	1.64×	1.75×	1.62×	1.58×	1.69×	4.72
	DFlash	4.15×	4.23×	3.09×	3.57×	4.01×	2.08×	2.16×	3.33×	4.53
	D-PACE	4.27×	4.39×	3.20×	3.69×	3.94×	2.17×	2.10×	3.39×	4.64
	LK-Losses	4.52×	4.64×	3.55×	3.84×	4.33×	2.30×	2.28×	3.64×	4.92
	Draft-OPD	4.94×	5.01×	3.76×	4.02×	4.57×	2.42×	2.49×	3.89×	5.23
	FReDraft	**5.38**×	**5.50**×	**4.11**×	**4.36**×	**4.85**×	**2.79**×	**2.74**×	**4.25**×	**5.71**

to regenerate responses for prompts in Qwen3-4B-Instruct-100K, and use the generated sequences as SFT dataset for draft model on Qwen3-8B. For on-policy training stage, we use 16K prompts released by Draft-OPD (Lei et al., 2026) as training set. While for off-policy baselines, we collect target-generated sequences on Qwen3-4B-Instruct-100K and the 16K prompts as training set. We evaluate with temperature set to 0 and 1 respectively. We set $L = 15$ for training and inference, and select $\beta = 0.3$ for FReDraft based on validation set. We refer to previous works (Chen et al., 2026a; Lei et al., 2026) and disable the thinking mode of target models (see Appendix E.3 for results with thinking-mode-enabled). See Appendix B for more implementation and hyperparameter selection.

4.2 Overall Performance

Main Results. According to Table 2, FReDraft achieves the highest average SR and τ among all methods based on DFlash architecture, demonstrating the effectiveness of FReDraft. Specifically, **(1)** Compared with Eagle-3, DFlash has higher SR, because it generates multiple tokens of the draft in parallel, while Eagle-3 generates the draft tokens auto-regressively. **(2)** Compared with off-policy methods (DFlash, D-PACE and LK-Losses), on-policy methods (Draft-OPD and FReDraft) obtain draft model with higher average SR and τ, showing the effectiveness of learning from drafts generated by the draft model itself. **(3)** FReDraft outperforms Draft-OPD, suggesting effectiveness of the proposed expected accepted length optimization method beyond on-policy training paradigm. Besides Table 2, we also evaluate FReDraft in the SGLang inference framework to assess deployment efficiency. See Appendix E.1 for the results.

4.3 Ablation Study

To evaluate the effectiveness of two techniques in Section 2.2 (i.e. token-level acceptance rate expectation marginalization and over-sampled token penalty) and two properties of the gradient

in Equation 7 (i.e. prefix-aware credit assignment and TVD), we design the following variants of FReDraft: (1) **w/o marg**: removes the token-level acceptance rate expectation marginalization and uses the gradient in Equation 3; (2) **w/o penalty**: removes the over-sampled token penalty and uses the gradient in Equation 5; (3) **FReDraft-ED**: replaces the prefix-aware credit assignment with ED; (4) **FReDraft-KL**: replaces the TVD in gradient with forward / reverse KL for accepted and rejected tokens respectively, referring to Draft-OPD. Since KL does not satisfy Equation 6, we do not implement over-sampled token penalty in FReDraft-KL. So the only difference between FReDraft-KL and w/o penalty is the TVD and KL. See Appendix D.2 for implementation details. We evaluate them on AIME25, HumanEval and MT-Bench, with DFlash architecture and Qwen3-4B as target model.

Table 3: Ablation study on Qwen3-4B, temperature = 0.

Method	AIME25		HumanEval		MT-Bench	
	SR	τ	SR	τ	SR	τ
w/o marg	3.12×	5.72	3.20×	5.78	1.74×	5.09
w/o penalty	4.32×	7.70	3.80×	6.90	2.12×	5.32
FReDraft-ED	3.96×	7.20	3.47×	6.30	2.10×	5.26
FReDraft-KL	4.19×	7.56	3.66×	6.67	2.12×	5.33
FReDraft	**4.43×**	**7.96**	**3.96×**	**7.18**	**2.25×**	**5.45**

We report the results in Table 3, and summarize observations as follows: **(1)** Gradient estimator in Equation 3 underperforms naive DFlash SFT, indicating that it is not effective for on-policy training. **(2)** Token-level expected acceptance rate marginalization provides the dominant enhancement. w/o penalty substantially outperforms w/o marg, and performs similarly to FReDraft, indicating that the marginalization is the dominant factor for enhancing the draft model performance. **(3)** Penalty for over-sampled token can provide additional enhancement. The full FReDraft still outperforms w/o penalty, showing that the penalty for over-sampled token can provide additional enhancement for draft model training. **(4)** FReDraft-ED slightly underperforms Draft-OPD, which also applies ED to weight the gradient, indicating that only TVD without prefix-aware credit assignment cannot enhance performance stably. **(5)** FReDraft-KL surpasses Draft-OPD, showing that prefix-aware credit assignment can effectively enhance performance even coupled with KL. But FReDraft-KL still underperforms FReDraft and w/o penalty, indicating the effectiveness of whole FReDraft, where prefix-aware credit assignment is coupled with TVD.

4.4 Hyperparameter Sensitivity

Penalty Strength β. We fix the draft length L to 15 and set β to $\{0.0, 0.3, 0.5, 0.7, 1.0\}$ respectively. We report τ on AIME25, HumanEval, MT-Bench and the average metric across all seven benchmarks introduced in Section 4.1 in Figure 3. The results show that with β from 0 to 1, the performance exhibits a trend of first increasing and then decreasing. The optimal β is different for different benchmarks, and $\beta = 0.3$ achieves the best average performance. See Section 4.5 about training dynamics for more analysis on influence of β to training efficiency.

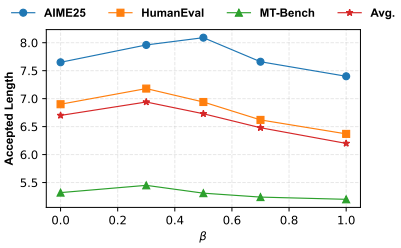

Figure 3: Influence of β.

Draft Length L. We fix β to 0.3 and set L to $\{7, 11, 15\}$ respectively during training and inference. We report the average τ of DFlash, Draft-OPD and FReDraft on all seven benchmarks in Figure 4. The results show that under larger L, FReDraft demonstrates stronger advantage over baselines. This is because larger L introduces more gradient noise from later tokens in the draft, which are usually rejected or discarded in verification. However, the prefix-aware credit assignment in FReDraft adaptively assigns low weight to these tokens, leading to better advantage.

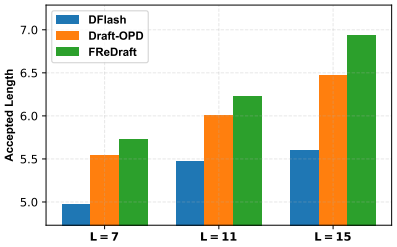

Figure 4: Influence of L.

4.5 In-Depth Analysis

Training Dynamics Analysis. We report the training dynamics during on-policy stage to show the convergence efficiency of FReDraft. Specifically, we plot the average accepted length of the training-time collected draft blocks on training set and checkpoint at each epoch on evaluation benchmarks.

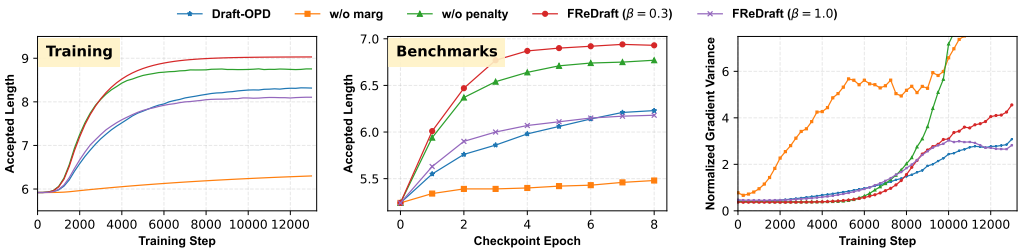

Figure 5: Average accepted length on training set, benchmarks and normalized gradient variance.

We also plot training-time normalized gradient variance (see Appendix D.4 for implementation details) during training. The results are shown in Figure 5. We summarize observations as follows: **(1)** The accepted length of w/o marg improves extremely slowly. We observe that 87.7% of the tokens in the sampled draft receive zero gradient in w/o marg (i.e. gradient in Equation 3). Its gradient variance is large, leading to low training efficiency. **(2)** Penalty for over-sampled tokens can further enhance convergence efficiency. Compared with FReDraft, though w/o penalty has similar accepted length curve during early training, its gradient variance increases much faster after middle training stage, and converges at lower accepted length, showing that penalty for over-sampled tokens can reduce the gradient noise from unsampled tokens and enhance convergence efficiency. **(3)** FReDraft and w/o penalty demonstrate higher convergence efficiency and final performance, showing the advantage of optimizing the draft expected accepted length. FReDraft, w/o penalty and Draft-OPD all leverage the target distribution to supervise draft model training. The difference is that the gradient of FReDraft and w/o penalty is derived from the expected accepted length of draft, while Draft-OPD is heuristically designed forward and reverse KL. Therefore, this comparison isolates the effectiveness of expected accepted length maximization and shows that it can enhance final performance. Moreover, compared with Draft-OPD, the gradient variance of FReDraft and w/o penalty increases more slowly in early training stage. See Appendix E.5 for wall-clock time comparison of on-policy training.

Prefix Survival & Token Acceptance Rate Distribution. To reveal the internal mechanism by which FReDraft improves draft accepted length during inference, we plot the average prefix survival rate and token-level acceptance rate distribution on different positions in all drafts collected on seven benchmarks. According to the results on 15 draft-generated tokens in Figure 6, the prefix survival rate of all methods decreases as the position steps from 1 to 15. However, the decreasing rate of FReDraft is much slower than other methods, especially at positions 3–8. The slowly-decreasing prefix survival rate of FReDraft enables verification to proceed to later tokens in the draft, thus achieving larger accepted length and higher speedup ratio. In terms of token-level acceptance rate, all methods exhibit extremely high acceptance rate for first several tokens, while FReDraft still shows a slight advantage. For positions 3–8, FReDraft demonstrates significant acceptance-rate advantage over other methods, which is consistent with the observation of prefix survival rate. For later positions, some methods show higher token acceptance rate than FReDraft. But due to first-rejection truncation, verification process rarely reaches these positions, leading to that improvements in acceptance at these positions have limited impact on accepted length.

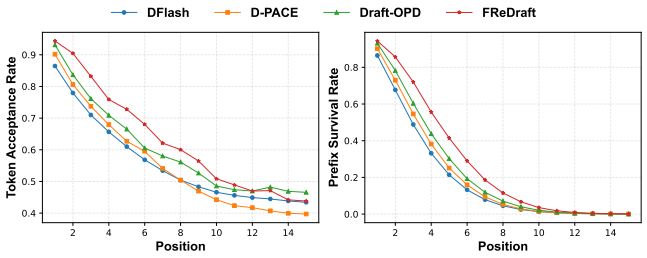

Figure 6: Token-acceptance & prefix-survival rate distribution.

5 Conclusion

We propose FReDraft, an on-policy draft-model training method that directly maximizes draft expected accepted length to model the first-rejection truncation property of verification. For efficient optimization, we introduce token-level acceptance-rate marginalization and over-sampled token penalty, enabling unbiased low-variance gradient estimation. Experiments across benchmarks and targets show that FReDraft consistently improves accepted length and speedup over strong baselines.

AI USE STATEMENT

In this work, we used generative AI tools for assist in the writing of proofs, design or provide feedback on research methodology or experiments and interpret results. We have not used generative AI tools for generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, implement methods, assist with translation, clean and reformat dataset, and support qualitative and thematic data analysis. Additionally, we used generative AI tools for create or modify scientific figures or images and draft parts of a research paper. We have reviewed all AI-assisted work. For the writing, proofs and figures drafted by LLMs, we review and revise them to ensure accuracy. For feedback provided by LLMs, we review it and reject the part that we think inappropriate. We review and revise all content generated by LLMs to ensure accuracy. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, Section 2 presents the training objective and optimization method, and Algorithm 1 summarizes the on-policy training procedure. Appendix C provides the theoretical derivations and proofs. Section 4.1 describes the target and draft models, training datasets and their construction, and evaluation settings. Appendix B details the estimator implementation, training hyperparameters, draft initialization, rollout and replay procedures, and training and evaluation systems. Appendices D.1, D.2, and D.3 document the baseline implementations, ablation variants, and benchmark sources and evaluation splits, respectively. The source code is available in an anonymous repository at `https://anonymous.4open.science/r/Code-FReDraft-A78E`.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *ICLR*, 2024.

Zihao An, Taichi Liu, Ziqiong Liu, Dong Li, Ruofeng Liu, and Emad Barsoum. PARD-2: Target-aligned parallel draft model for dual-mode speculative decoding. *arXiv preprint arXiv:2605.08632*, 2026.

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.

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

Shrenik Bhansali and Larry Heck. Draft, verify, and improve: Toward training-aware speculative decoding. *arXiv preprint arXiv:2510.05421*, 2025.

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 *ICML*, 2024.

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.

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding. In *ICML*, 2026a.

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.

Qiaoling Chen, Zijun Liu, Peng Sun, Shenggui Li, Guoteng Wang, Ziming Liu, Yonggang Wen, Siyuan Feng, and Tianwei Zhang. ReSpec: Towards optimizing speculative decoding in reinforcement learning systems. *Proceedings of Machine Learning and Systems*, 8:367–379, 2026b.

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.

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.

Yuqian Fu, Haohuan Huang, Kaiwen Jiang, Jiacai Liu, Zhuo Jiang, Yuanheng Zhu, and Dongbin Zhao. Revisiting on-Policy distillation: Empirical failure modes and simple fixes. In *Third Conference on Language Modeling*, 2026.

Roman Garipov, Fedor Velikonivtsev, Ivan Ermakov, Ruslan Svirschevski, Vage Egiazarian, and Max Ryabinin. AutoJudge: Judge decoding without manual annotation. *NeurIPS*, 2025.

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.

Geonmo Gu, Byeongho Heo, HeeJae Jun, Yoohoon Kang, Sangmin Lee, Sangdoo Yun, and Dongyoon Han. Verification-aware training for speculative decoding. *arXiv preprint arXiv:2608.30135*, 2026.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *ICLR*, 2024.

Lujun Gui, Bin Xiao, Lei Su, and Weipeng Chen. Boosting lossless speculative decoding via feature sampling and partial alignment distillation. *arXiv preprint arXiv:2408.15562*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS Datasets and Benchmarks Track (Round 2)*, 2021.

Shijing Hu, Jingyang Li, Xingyu Xie, Zhihui Lu, Kim-Chuan Toh, and Pan Zhou. GRIFFIN: Effective token alignment for faster speculative decoding. *NeurIPS*, 2025.

Shijing Hu, Jingyang Li, Zhihui Lu, and Pan Zhou. Bridging draft policy misalignment: Group tree optimization for speculative decoding. In *ICLR*, 2026.

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

Haozhe Ji, Pei Ke, Zhipeng Hu, Rongsheng Zhang, and Minlie Huang. Tailoring language generation models under total variation distance. In *ICLR*, 2023.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *ICLR*, 2024.

Haodi Lei, Yafy 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.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *ICML*, 2023.

Shenggui Li, Chao Wang, Yikai Zhu, Yubo Wang, Fan Yin, Shuai Shi, Yefei Chen, Xiaomin Dong, Qiaoling Chen, Jin Pan, et al. SpecForge: A flexible and efficient open-source training framework for speculative decoding. *arXiv preprint arXiv:2603.18567*, 2026a.

Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huanang Gao, Wenkai Yang, Zhiyuan Liu, et al. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *arXiv preprint arXiv:2604.13016*, 2026b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *ICML*, 2024a.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-2: Faster inference of language models with dynamic draft trees. In *EMNLP*, 2024b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *NeurIPS*, 2025.

Xiaoxuan Liu, Lanxiang Hu, Peter Bailis, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. Online speculative decoding. In *ICML*, 2024.

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

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.

Siru Ouyang, Shuohang Wang, Minhao Jiang, Ming Zhong, Donghan Yu, Jiawei Han, and Yelong Shen. Temperature-centric investigation of speculative decoding with knowledge distillation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 13125–13137, 2024.

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

Yu-Yang Qian, Hao-Cong Wu, Yichao Fu, Hao Zhang, and Peng Zhao. When drafts evolve: Speculative decoding meets online learning. In *ICML*, 2026.

Ranajoy Sadhukhan, Jian Chen, Zhuoming Chen, Vashisth Tiwari, Ruihang Lai, Jinyuan Shi, Ian En-Hsu Yen, Avner May, Tianqi Chen, and Beidi Chen. MagicDec: Breaking the latency-throughput tradeoff for long context generation with speculative decoding. In *ICLR*, 2025.

Alexander Samarin, Sergei Krutikov, Anton Shevtsov, Sergei Skvortsov, Filipp Fisin, and Alexander Golubev. LK losses: Direct acceptance rate optimization for speculative decoding. In *ICML*, 2026.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026a.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, et al. Kimi k2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026b.

Ramon Van Handel. Probability in high dimension. Technical report, PRINCETON UNIV NJ, 2014.

Jikai Wang, Yi Su, Juntao Li, Qingrong Xia, Zi Ye, Xinyu Duan, Zhefeng Wang, and Min Zhang. Opt-tree: Speculative decoding with adaptive draft tree structure. *Transactions of the Association for Computational Linguistics*, 13:188–199, 2025.

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.

Tianhua Xia, Mugilan Ganesan, Yifei Feng, Haiyu Wang, Maximilian Egger, and Sai Qian Zhang. Acceptance-aware draft model training for speculative decoding. *arXiv preprint arXiv:2609.24150*, 2026.

Bangjun Xiao, Bingquan Xia, Bo Yang, Bofei Gao, Bowen Shen, Chen Zhang, Chenhong He, Chiheng Lou, Fuli Luo, Gang Wang, et al. Mimo-v2-flash technical report. *arXiv preprint arXiv:2601.02780*, 2026.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

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*, 2025.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. GLM-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

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

Yudi Zhang, Weilin Zhao, Xu Han, Tiejun Zhao, Wang Xu, Hailong Cao, and Conghui Zhu. Speculative decoding meets quantization: Compatibility evaluation and hierarchical framework design. *arXiv preprint arXiv:2505.22179*, 2025.

Juntao Zhao, Wenhao Lu, Sheng Wang, Lingpeng Kong, and Chuan Wu. QSpec: Speculative decoding with complementary quantization schemes. In *EMNLP*, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *NeurIPS Datasets and Benchmarks Track*, 2023.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *NeurIPS*, 2024.

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 *ICLR*, 2024.

Xiandong Zou, Jianshu Li, Jing Huang, and Pan Zhou. Variational speculative decoding: Rethinking draft training from token likelihood to sequence acceptance. In *ICML*, 2026.

A Detailed Discussion about Speculative Draft Models

A.1 Draft Model Architectures

Independent autoregressive drafters. The original speculative decoding and speculative sampling algorithms use a small, independently parameterized language model to generate a draft autoregressively, after which the target model verifies the complete draft in parallel (Leviathan et al., 2023; Chen et al., 2023). The draft and target can be developed and deployed independently, and the same verification rule also supports tree-structured proposals (Miao et al., 2024). The main limitation is that autoregressive draft generation places a serial model invocation on the critical path for every proposed token. Consequently, increasing the draft length improves the amount of work amortized by one verification call, but also increases drafting latency.

Self-drafting and feature-based drafters. Another line of work reuses computation or representations from the target model. Draft&Verify skips selected target layers to construct a faster self-draft (Zhang et al., 2024), while QSpec uses a complementary low-precision execution of the target as the drafter (Zhao et al., 2025). Medusa attaches multiple prediction heads to a frozen target model and produces a tree of candidates for several future positions (Cai et al., 2024). The EAGLE family instead predicts future target-model features and then maps those features to token distributions (Li et al., 2024a;b). Eagle-3 (Li et al., 2025) improves candidate-tree construction and training-time feature alignment. These methods reduce or share draft computation, but their draft distributions still need to be aligned with the target distribution for high acceptance.

Parallel block drafters. DFlash formulates drafting as block diffusion: conditioned on features from the target-model prefix, a lightweight draft network predicts an entire masked block in parallel (Chen et al., 2026a). DSpark likewise uses semi-autoregressive generation with a confidence-aware schedule to expose more parallelism during draft construction (Cheng et al., 2026). Parallel block drafters remove most of the serial drafting cost and therefore make accepted length especially important: once a block has been produced, accepting a longer prefix adds little drafting overhead. Our experiments use DFlash because it provides this efficient parallel drafting path. The objective proposed in this paper is nevertheless defined over the induced draft distribution and is not tied to a particular neural architecture.

A.2 Draft Model Training Methods

Supervised fine-tuning. A common recipe trains the draft model with token-level cross-entropy on sequences generated by the target model (Li et al., 2025; Chen et al., 2026a; Li et al., 2026a). This supervised finetuning (SFT) method is stable. However, it observes target prefixes during training whereas deployment conditions the draft model on its own proposals, leading to exposure bias.

Distribution distillation. Compared with SFT, white-box distillation provides the target distribution over vocabulary as supervision to train the draft, thereby conveying more information than a single hard token and typically achieves better draft model performance (Gui et al., 2024; Ouyang et al., 2024). For example, DistillSpec studies draft-model knowledge distillation with different data sources and distribution-divergence objectives (Zhou et al., 2024). OSD (Liu et al., 2024), OnlineSpec (Qian et al., 2026) and ReSpec (Chen et al., 2026b) further model an online training problem for continuous adaptation of the draft model during deployment. They leverage the vocabulary distribution output by target model during deployment to train the draft model continuously.

Verification-aware training. Recent methods begin to explore incorporating property of speculative verification process into draft model training, rather than merely letting the draft model learn to imitate target-generated tokens or distributions. For example, LK-Losses proposes to directly optimize token-level acceptance rate during verification and derives the objective as minimizing the TVD between draft model and target model distribution (Samarin et al., 2026). However, LK-Losses optimizes the draft distribution at each position with static exponential decay as weight, without modeling the influence from earlier tokens to later tokens. Draft accepted length is a metric directly related to the speedup ratio of speculative decoding given other draft model, target model and device configurations (Zhao et al., 2025; Zhang et al., 2024). Therefore, some recent methods take accepted

length into account when designing their training method (Bhansali & Heck, 2025; Gu et al., 2026; Huang et al., 2024; Bachmann et al., 2025; Garipov et al., 2025). PARD-2 (An et al., 2026) observes that cumulative confidence of target model over the preceding prefix is closely correlated to the token-level acceptance rate and it weights the gradient of tokens at different positions with the cumulative confidence. Similarly, D-PACE (Wu et al., 2026) observes that the draft-model summed cumulative confidence on target-generated sequence chunks is empirically related to the average accepted length of drafts, and D-PACE derives position weight in training data according to this surrogate. VSD (Zou et al., 2026) proposes a variational objective as lower bound of draft tree utility and implement Expectation-Maximization algorithm to optimize the objective. Besides these, two concurrent works VAT (Gu et al., 2026) and WTV (Xia et al., 2026) also take verification information into training objective. Despite effectiveness of above verification-aware training method compared with naive SFT, these methods, however, optimize a metric of sequences under target-distribution as surrogate, which does not match "first-rejection truncation" in practical verification. This results in exposure bias and gap between optimized objective and real verification process. Take VSD as an example, although it uses the draft model to propose training sequences, it further leverages the target model to filter and correct training samples for subsequent training process. Therefore, the trial-and-error samples, which are important for mitigating exposure bias, are excluded in training. Moreover, the optimized surrogate may not necessarily lead to later "first-rejection". For example, D-PACE optimizes summed cumulative draft confidence, which makes draft model diverge from target when target model itself is uncertain. In comparison, FReDraft directly optimizes the expected accepted length of draft generated by draft model itself, which naturally matches on-policy training paradigm, thus achieving better performance.

On-policy paradigm. On-policy paradigm trains the draft model on its own generated sequences, matching practical verification process and eliminating exposure bias. An early work is GTO (Hu et al., 2026) designed specifically for tree-structured draft (e.g. Eagle-3). It adopts semi-on-policy training by combining reinforcement learning (RL) with supervised finetuning. However, GTO implements its RL training recipe referring to GRPO (Shao et al., 2024), which leads to coarse-grained credit assignment. On-policy distillation (OPD) combines the advantage of on-policy training and dense supervision of distillation, thus becoming a widely-adopted training paradigm in LLM post-training stage (Xu et al., 2026; Xiao et al., 2026; Team et al., 2026a; Zeng et al., 2026). Inspired by the success of OPD in LLM post-training, Draft-OPD proposes to train the draft model on drafts collected by itself during speculative decoding (Lei et al., 2026). Its released training recipe applies a forward-KL-style loss to accepted/committed tokens and a sampled reverse-KL estimator to invalidated draft suffixes, together with a static exponential decay over draft positions. On-policy collection removes the train–inference prefix mismatch, but the token-level divergences and static position decay do not represent the realized survival probability of each prefix, leading to inefficient training signal allocation (Chen et al., 2026a; Wu et al., 2026). In contrast, FReDraft retains on-policy collection while deriving its credit assignment from the expected accepted length itself. The resulting weight of a proposal depends on both the acceptance of its prefix and the future accepted-length contributions that the proposal enables.

B Implementation Details

B.1 Block-Level Estimator Construction

Algorithm 2 expands the estimator-computation step in Algorithm 1. The implementation uses a minimization surrogate whose negative gradient equals the ascent estimator in Equation 7. Here $\mathrm{sg}(\cdot)$ denotes stop-gradient. The reverse cumulative sum used to compute w_t excludes the current position, so $\sum_{k=t+1}^{L} A_k S_{k-1} = 0$ for $t = L$; this is the implementation of the "beyond position t" term in Proposition 2.

B.2 Memory-efficient distribution representation

On-policy training requires target-distribution information for collected draft proposals, including rejected tokens and their invalidated suffixes. For N collected proposal positions and vocabulary $\mathcal{V}$, retaining the full target distributions requires $\mathcal{O}(N|\mathcal{V}|)$ memory. This storage cost can become prohibitive and cause out-of-memory errors when collecting and replaying many draft blocks. Fol-

Algorithm 2 Block-Level Construction of $\widehat{\mathbf{g}}_\beta$

Require: Complete collected draft blocks $\mathcal{B}$, frozen target model p, draft model q_θ, penalty coefficient β
Ensure: Batch estimator $\widehat{\mathbf{g}}_\beta$
Initialize $\mathcal{L} \leftarrow 0$ and $M \leftarrow 0$
for each $\mathbf{y} = (y_1, \ldots, y_L) \in \mathcal{B}$ **do**
  for $t = 1, \ldots, L$ **do**
    Obtain p_t, $q_{\theta,t}$, and the sampled token y_t from verification / replay
    $\alpha_t \leftarrow \exp(\min\{0, \log p_t(y_t) - \log q_{\theta,t}(y_t)\})$
    $A_t \leftarrow 1 - D_{\mathrm{TV}}^{(K)}(p_t, q_{\theta,t})$
  end for
  $S_0 \leftarrow 1$ and $S_{t-1} \leftarrow \prod_{j=1}^{t-1} \alpha_j$ for $t = 2, \ldots, L$
  for $t = 1, \ldots, L$ **do**
    $w_t \leftarrow (\sum_{k=t+1}^{L} A_k S_{k-1}) \mathbb{1}_{p_t(y_t) > q_{\theta,t}(y_t)} - (\beta S_{t-1}) \mathbb{1}_{p_t(y_t) < q_{\theta,t}(y_t)}$
    $\ell_t \leftarrow -\operatorname{sg}(w_t) \log q_{\theta,t}(y_t) + \operatorname{sg}((1-\beta) S_{t-1}) D_{\mathrm{TV}}^{(K)}(p_t, q_{\theta,t})$
  end for
  $\mathcal{L} \leftarrow \mathcal{L} + \sum_{t=1}^{L} \ell_t$; $M \leftarrow M + 1$
end for
return $\widehat{\mathbf{g}}_\beta = -\nabla_\theta(\mathcal{L}/M)$

lowing the use of target Top-K entries in on-policy distillation (Agarwal et al., 2024; Fu et al., 2026), we compress the stored target distributions to reduce this cost to $\mathcal{O}(NK)$. Specifically, the implementation is as follows:

Implementation. At position t, we retain the target top-K tokens and the sampled draft token:

$$\mathcal{K}_t := \mathrm{TopK}(p_t) \cup \{y_t\}, \qquad \mathcal{T}_t := \mathcal{V} \setminus \mathcal{K}_t.$$

During replay, we gather the draft probabilities at the same tokens, using the original vocabulary-wide normalization rather than renormalizing over $\mathcal{K}_t$. All omitted tokens are aggregated into one residual category with target and draft masses

$$\begin{aligned} \varepsilon_{p,t} &:= 1 - \sum_{v \in \mathcal{K}_t} p_t(v), \\ \varepsilon_{q,t} &:= 1 - \sum_{v \in \mathcal{K}_t} q_{\theta,t}(v), \\ \varepsilon_t &:= \min\{\varepsilon_{p,t}, \varepsilon_{q,t}\}. \end{aligned} \tag{8}$$

The resulting TVD on the common retained-token-plus-tail partition is

$$\widetilde{D}_{\mathrm{TV}}^{(K)}(p_t, q_{\theta,t}) := \frac{1}{2} \left[\sum_{v \in \mathcal{K}_t} |p_t(v) - q_{\theta,t}(v)| + |\varepsilon_{p,t} - \varepsilon_{q,t}| \right].$$

This representation preserves normalization and all retained probability mass. The draft tail mass $\varepsilon_{q,t}$ remains differentiable; neither it nor the retained draft probabilities are detached. Because $y_t \in \mathcal{K}_t$, the sampled-token acceptance ratio and under-/over-sampling indicators are computed from exact probabilities. The compression introduces no additional target-model forward pass.

For brevity, write

$$D_t := D_{\mathrm{TV}}(p_t, q_{\theta,t}), \qquad \widetilde{D}_t := \widetilde{D}_{\mathrm{TV}}^{(K)}(p_t, q_{\theta,t}),$$

and

$$A_t := 1 - D_t, \qquad \widetilde{A}_t := 1 - \widetilde{D}_t.$$

Here $\widetilde{A}_t$ is the approximation used for the marginalized acceptance coefficient in Algorithm 2.

Next, we provide theoretical analysis and empirical analysis to show that the approximation error is negligible in practice.

Theoretical bounds. The approximation error is controlled by the omitted probability mass, independently of vocabulary size. Throughout the analysis, the target distribution, sampled prefix, and realized retained set are held fixed during differentiation. The gradient statements apply away from the nondifferentiable equality boundaries of the absolute-value terms and concern tail aggregation in exact arithmetic.

Proposition 3 (TVD and logit-gradient error under tail aggregation)**.** *Let $q_{\theta,t} = \mathrm{softmax}(z_t)$ and define ε_t as in Equation 8. Then*

$$0 \le D_t - \widetilde{D}_t = \widetilde{A}_t - A_t \le \varepsilon_t, \tag{9}$$

$$\left\| \nabla_{z_t} D_t - \nabla_{z_t} \widetilde{D}_t \right\|_1 \le 2\varepsilon_t, \tag{10}$$

$$\left\| \nabla_{z_t} D_t - \nabla_{z_t} \widetilde{D}_t \right\|_2 \le \sqrt{2}\,\varepsilon_t. \tag{11}$$

Furthermore, with $\mathbf{J}_t := \partial z_t / \partial \theta$,

$$\left\| \nabla_\theta D_t - \nabla_\theta \widetilde{D}_t \right\|_2 \le \sqrt{2}\,\|\mathbf{J}_t\|_{\mathrm{op}}\,\varepsilon_t, \tag{12}$$

where $\|\cdot\|_{\mathrm{op}}$ denotes the Euclidean operator norm.

Proposition 3 bounds both quantities affected by compression: the marginalized acceptance coefficient and the dense TVD gradient. Importantly, the bounds depend on the *smaller* omitted mass, rather than the sum of the target and draft tail masses. They hold pointwise for every realized retained set, including when that set contains the sampled token.

We can also bound the perturbation of the complete block-level gradient estimator. Let $\widehat{\mathbf{g}}_\beta$ denote the full-vocabulary estimator in Equation 7, and let $\widehat{\mathbf{g}}_\beta^{(K)}$ denote its compressed implementation in Algorithm 2.

Corollary 2 (Compression error of the block-level estimator)**.** *Suppose that, for all relevant positions, prefixes, and tokens,*

$$\|\nabla_\theta \log q_{\theta,t}(v)\|_2 \le C. \tag{13}$$

For any $\beta \in [0,1]$, evaluating the full and compressed estimators on the same sampled block gives

$$\left\| \widehat{\mathbf{g}}_\beta^{(K)} - \widehat{\mathbf{g}}_\beta \right\|_2 \le C \sum_{k=1}^{L} (k-\beta) S_{k-1} \varepsilon_k. \tag{14}$$

Consequently, under on-policy draft sampling with the decoding-context distribution held fixed during differentiation,

$$\left\| \mathbb{E}\left[\widehat{\mathbf{g}}_\beta^{(K)}\right] - \nabla_\theta J(\theta;\bar{\theta}) \right\|_2 \le C\,\mathbb{E}\left[\sum_{k=1}^{L} (k-\beta) S_{k-1} \varepsilon_k\right], \tag{15}$$

where $J(\theta;\bar{\theta})$ is the fixed-context-distribution objective defined in Appendix C.2.

Thus, the full-vocabulary estimator retains its exact expectation-preservation guarantee, while the compressed implementation has a tail-mass-dependent approximation error. The proofs of both results are provided after the following empirical discussion.

Empirical evidence. We use $K = 16$ for FReDraft and its corresponding loss variants. Empirically, the target top-16 entries cover 99.7% of the target probability mass and 97.6% of the draft probability mass on average. Additionally retaining the sampled token can only decrease the omitted masses. Let $\widehat{\mathbb{E}}$ denote averaging over the distributions used for these coverage measurements. Applying Proposition 3 pointwise and averaging gives

$$\widehat{\mathbb{E}}\left[D_t - \widetilde{D}_t\right] \le \widehat{\mathbb{E}}[\varepsilon_t] \le \widehat{\mathbb{E}}[\varepsilon_{p,t}],$$

$$\widehat{\mathbb{E}}\left[\left\|\nabla_{z_t} D_t - \nabla_{z_t} \widetilde{D}_t\right\|_2\right] \le \sqrt{2}\,\widehat{\mathbb{E}}[\varepsilon_t] \le \sqrt{2}\,\widehat{\mathbb{E}}[\varepsilon_{p,t}].$$

Using the reported rounded target-tail mass of 0.003, these inequalities give approximate upper bounds of 3.0×10^{-3} for the mean TVD error and $\sqrt{2} \times 0.003 \approx 4.24 \times 10^{-3}$ for the mean Euclidean logit-gradient error. Equivalently, the average marginalized acceptance rate is overestimated by at most approximately 0.3 percentage points. These measurements support using $K = 16$ as a memory-efficient approximation with small average absolute error.

Proofs.

Proof of Proposition 3. Suppress the position index and write $q = \mathrm{softmax}(z)$. If the tail is empty, the full and compressed expressions coincide, so all bounds hold with zero error. Otherwise, define the positive and negative discrepancies within the tail:

$$U := \sum_{v\in\mathcal{T}} (q(v) - p(v))_+, \qquad V := \sum_{v\in\mathcal{T}} (p(v) - q(v))_+,$$

where $(a)_+ := \max\{a, 0\}$. The retained terms are identical in D and $\widetilde{D}$, while

$$\sum_{v\in\mathcal{T}} |q(v) - p(v)| = U + V, \qquad \varepsilon_q - \varepsilon_p = U - V.$$

Therefore,

$$D - \widetilde{D} = \frac{1}{2}\left(U + V - |U - V|\right) = \min\{U, V\} \le \min\{\varepsilon_q, \varepsilon_p\} = \varepsilon.$$

Here $U \le \varepsilon_q$ and $V \le \varepsilon_p$. Since $\widetilde{A} - A = D - \widetilde{D}$, this proves Equation 9.

For the gradient, define

$$s_v := \mathrm{sign}(q(v) - p(v)), \qquad \sigma := \mathrm{sign}(\varepsilon_q - \varepsilon_p).$$

Away from the equality boundaries, differentiating the full and compressed TVDs gives

$$\Delta_z := \nabla_z D - \nabla_z \widetilde{D} = \frac{1}{2}\sum_{v\in\mathcal{T}} (s_v - \sigma)\nabla_z q(v).$$

Only tail tokens whose individual discrepancy has the opposite sign to the aggregate tail discrepancy contribute to this difference. Let

$$\mathcal{M} := \{v \in \mathcal{T} : (q(v) - p(v))(\varepsilon_q - \varepsilon_p) < 0\}, \qquad m := q(\mathcal{M}).$$

Then

$$\Delta_z = -\sigma \sum_{v\in\mathcal{M}} \nabla_z q(v) = -\sigma \nabla_z q(\mathcal{M}). \tag{16}$$

If $\varepsilon_q > \varepsilon_p$, every token in $\mathcal{M}$ satisfies $q(v) < p(v)$, and hence

$$m = q(\mathcal{M}) \le p(\mathcal{M}) \le \varepsilon_p.$$

If $\varepsilon_q < \varepsilon_p$, then $m \le q(\mathcal{T}) = \varepsilon_q$. Thus, in either case,

$$m \le \min\{\varepsilon_p, \varepsilon_q\} = \varepsilon.$$

The softmax Jacobian gives

$$\frac{\partial q(\mathcal{M})}{\partial z(v)} = q(v)\big(\mathbb{1}_{v\in\mathcal{M}} - m\big) = \begin{cases} q(v)(1-m), & v \in \mathcal{M}, \\ -mq(v), & v \notin \mathcal{M}. \end{cases}$$

Using Equation 16,

$$\begin{aligned} \|\Delta_z\|_1 &= (1-m)\sum_{v\in\mathcal{M}} q(v) + m\sum_{v\notin\mathcal{M}} q(v) \\ &= 2m(1-m) \le 2\varepsilon. \end{aligned}$$

Similarly,

$$\begin{aligned} \|\Delta_z\|_2^2 &= (1-m)^2 \sum_{v\in\mathcal{M}} q(v)^2 + m^2 \sum_{v\notin\mathcal{M}} q(v)^2 \\ &\le (1-m)^2 m^2 + m^2(1-m)^2 \\ &= 2m^2(1-m)^2 \le 2\varepsilon^2. \end{aligned}$$

This proves Equations 10 and 11.

Finally, the chain rule yields

$$\nabla_\theta D_t - \nabla_\theta \widetilde{D}_t = \mathbf{J}_t^\top \left(\nabla_{z_t} D_t - \nabla_{z_t} \widetilde{D}_t\right).$$

Taking the Euclidean norm and applying Equation 11 proves Equation 12. □

Proof of Corollary 2. Define

$$\delta_t := D_t - \widetilde{D}_t, \qquad \mathbf{e}_t := \nabla_\theta D_t - \nabla_\theta \widetilde{D}_t.$$

By Proposition 3, $0 \le \delta_t \le \varepsilon_t$. The parameter-space counterpart of Equation 16, together with the score bound in Equation 13, gives

$$\begin{aligned}\|\mathbf{e}_t\|_2 &= \left\| \sum_{v \in \mathcal{M}_t} q_{\theta,t}(v) \nabla_\theta \log q_{\theta,t}(v) \right\|_2 \\ &\le C q_{\theta,t}(\mathcal{M}_t) \le C\varepsilon_t.\end{aligned}$$

Recall that $G_t = \mathbb{1}_{p_t(y_t) > q_{\theta,t}(y_t)} \nabla_\theta \log q_{\theta,t}(y_t)$ and that $\widetilde{G}_t$ is the corresponding over-sampled-token score term. Expanding Equation 7, the full-vocabulary estimator is

$$\widehat{\mathbf{g}}_\beta = \sum_{t=1}^{L} \left[\left(\sum_{k=t+1}^{L} A_k S_{k-1} \right) G_t - (1-\beta) S_{t-1} \nabla_\theta D_t - \beta S_{t-1} \widetilde{G}_t \right].$$

The compressed implementation replaces A_k by $\widetilde{A}_k$ and D_t by $\widetilde{D}_t$. The quantities S_{t-1}, G_t, and $\widetilde{G}_t$ remain unchanged because the sampled-token probabilities are exact. Since $\widetilde{A}_k - A_k = \delta_k$, subtracting the two estimators on the same sampled block gives

$$\begin{aligned}\widehat{\mathbf{g}}_\beta^{(K)} - \widehat{\mathbf{g}}_\beta = (1-\beta) \sum_{t=1}^{L} S_{t-1} \mathbf{e}_t \\ + \sum_{t=1}^{L} \sum_{k=t+1}^{L} S_{k-1} \delta_k G_t.\end{aligned}$$

The score assumption implies $\|G_t\|_2 \le C$. Therefore,

$$\begin{aligned}\left\| \widehat{\mathbf{g}}_\beta^{(K)} - \widehat{\mathbf{g}}_\beta \right\|_2 &\le C(1-\beta) \sum_{t=1}^{L} S_{t-1}\varepsilon_t + C \sum_{t=1}^{L} \sum_{k=t+1}^{L} S_{k-1}\varepsilon_k \\ &= C \sum_{k=1}^{L} (k-\beta) S_{k-1}\varepsilon_k. \end{aligned} \tag{17}$$

This proves Equation 14.

Under the stated on-policy and fixed-context-distribution assumptions, Corollary 1 gives

$$\mathbb{E}[\widehat{\mathbf{g}}_\beta] = \nabla_\theta J(\theta; \bar{\theta}).$$

Taking expectations in Equation 17 and using $\|\mathbb{E}[X]\|_2 \le \mathbb{E}[\|X\|_2]$ proves Equation 15. □

B.3 Hyperparameters and β Selection

Table 4 summarizes the on-policy training configuration used for FReDraft and its loss ablations. Unless a sensitivity experiment explicitly varies L or β, the same values are used for all methods in the comparison. The target is frozen throughout this stage and only the DFlash draft-network parameters are optimized. In Table 4, $L = 15$ is a widely used experiment setting in previous works, including DFlash and Draft-OPD (Notably, the two works set DFlash block size as 16, with one anchor token generated by target model. So the draft length is also 15). To select penalty coefficient β, we randomly sample 10% prompts from the 16K prompts released by Draft-OPD as validation set. Then with the selected $\beta = 0.3$, we use all 16K prompts to conduct FReDraft on-policy training to obtain results in Section 4.

B.4 Other Key Implementation Details

Draft initialization and DFlash replay. We initialize from the SFT DFlash checkpoints described in Section 4.1. For both Qwen3 targets, the DFlash drafter contains five transformer layers and

Table 4: On-policy training hyperparameters.

Hyperparameter	Value
Target models	Qwen3-4B and Qwen3-8B
Draft architecture	5-layer DFlash
Draft length L	15
Target distribution retained support K	16
Penalty coefficient β	0.3
ED coefficient (FReDraft-ED only)	0.8
Training epochs	8
Prompt batch size	12
Micro-batch size per draft worker	1
Update-accumulation steps	10
Learning rate	2×10^{-4}
Training proposal sampling	temperature 1.0, top-$p = 1.0$

consumes hidden states from target layers $\{1, 9, 17, 25, 33\}$. Its configured block size is 16: the first entry is an already available anchor and the remaining 15 entries are future-token proposals predicted in parallel. For the first speculative block, response token zero—sampled during target prefill—is the anchor; subsequent blocks are anchored at the preceding verification boundary. This reproduces the layout used by DFlash inference.

During rollout, we record every verification boundary, every proposed token, its offset within the block, its target log-probability, and the retained target-distribution entries. Accepted proposals are present in the committed response, whereas the first rejected proposal and its remaining invalidated suffix are carried in rejected-suffix metadata. During training replay, the two streams are joined by $(\text{sample}, \text{anchor}, \text{offset})$ to reconstruct the original on-policy block. We use response-anchor stride one, disable random anchor subsampling, and do not cap the number of rejected-suffix tokens. A block is used only if all proposal offsets are recovered; this prevents a truncated suffix from being interpreted as a shorter draft.

Distribution capture and numerical stability. The frozen target logits used for the loss are captured from the verification/replay computation already required to obtain DFlash target features. The draft probabilities at the same token IDs are gathered from the trainable draft logits, and the remaining draft mass is computed by subtraction. Consequently, constructing the FReDraft loss requires no additional target-model forward pass relative to the on-policy DFlash replay. Probability ratios are evaluated in log space as $\exp(\min\{0, \log p_t(y_t) - \log q_{\theta,t}(y_t)\})$. The frozen target and the draft network use the same tokenizer and vocabulary.

Training and evaluation systems. Training is implemented on top of the VERL on-policy distillation pipeline with FSDP for the draft learner and SGLang for DFlash rollout. In the Qwen3-4B configuration, three A100-80GB GPUs host draft-model workers and one A100-80GB GPU hosts the frozen target worker. The target model is always run in evaluation mode, and thinking mode is disabled unless explicitly stated. For evaluation, all methods use the same tokenizer, prompt template, stopping criteria, DFlash block size, and target-only baseline. We report results at temperatures 0 and 1 as specified in Section 4.1; greedy evaluation uses top-$k = 1$, while temperature-1 evaluation uses the untruncated categorical distribution. Accepted length counts the number of output tokens accepted per target verification call, without the resampling token or anchor token generated by the target model.

C Theoretical Analysis

C.1 Derivation of Expected Acceptance Length of Given Draft

Fix a context $\mathbf{x}$ and a draft $\mathbf{y} = (y_1, \ldots, y_L)$. Let $Z_t \in \{0, 1\}$ indicate that the token at position t is accepted by speculative verification, and let

$$I_t := \prod_{j=1}^{t} Z_j$$

indicate that the first t draft tokens are all accepted. Since verification stops at the first rejection, the number of accepted draft tokens is

$$\tau(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{L} I_t. \tag{18}$$

Conditioned on reaching position j, the acceptance probability of y_j is $\alpha_j(y_j)$ as defined in Equation 1. Applying the chain rule of conditional probability therefore gives

$$\Pr(I_t = 1 \mid \mathbf{x}, \mathbf{y}) = \prod_{j=1}^{t} \alpha_j(y_j).$$

Taking the conditional expectation of Equation 18 and using linearity of expectation yields

$$\mathbb{E}[\tau(\mathbf{x}, \mathbf{y}) \mid \mathbf{x}, \mathbf{y}] = \sum_{t=1}^{L} \Pr(I_t = 1 \mid \mathbf{x}, \mathbf{y}) = \sum_{t=1}^{L} \prod_{j=1}^{t} \alpha_j(y_j) = \bar{\tau}_\theta(\mathbf{x}, \mathbf{y}).$$

Thus, averaging $\bar{\tau}_\theta$ over contexts and drafts gives the objective in Equation 2.

C.2 Derivation of Equation 3

The decoding-context distribution $\mathcal{P}$ is induced by on-policy rollouts and therefore changes with the draft model q_θ across training iterations, but $\nabla_\theta \mathcal{P}$ is too hard to estimate. Therefore, previous on-policy method Draft-OPD (Lei et al., 2026) does not differentiate $\mathcal{P}$ in the gradient. FReDraft follows this implementation in practice. Specifically, within each learner update, the decoding contexts have already been collected and are treated as fixed when computing the block-level gradient. Formally, let $\bar{\theta}$ denote the behavior parameters used to generate the current rollout batch, and write the corresponding context distribution as $\mathcal{P}_{\bar{\theta}}$. The objective optimized in one learner update can be written as

$$J(\theta; \bar{\theta}) := \mathbb{E}_{x \sim \mathcal{P}_{\bar{\theta}}} \left[J_\theta(x) \right], \qquad J_\theta(x) := \mathbb{E}_{y \sim q_\theta(\cdot|x)} \left[\bar{\tau}_\theta(x, y) \right].$$

The gradient is taken with respect to the learner parameters θ while keeping the rollout parameters $\bar{\theta}$ fixed, giving

$$\nabla_\theta J(\theta; \bar{\theta}) = \mathbb{E}_{x \sim \mathcal{P}_{\bar{\theta}}} \left[\nabla_\theta J_\theta(x) \right]. \tag{19}$$

After the draft model is updated, new on-policy rollouts refresh $\mathcal{P}_{\bar{\theta}}$, so the context distribution tracks the current draft model across training iterations.

In contrast, if one defines a fully coupled trajectory-level objective

$$J_{\mathrm{traj}}(\theta) := \mathbb{E}_{x \sim \mathcal{P}_\theta} \left[J_\theta(x) \right],$$

then its total derivative contains an additional visitation-distribution term,

$$\nabla_\theta J_{\mathrm{traj}}(\theta) = \mathbb{E}_{x \sim \mathcal{P}_\theta} \left[\nabla_\theta J_\theta(x) \right] + \mathbb{E}_{x \sim \mathcal{P}_\theta} \left[J_\theta(x) \nabla_\theta \log \mathcal{P}_\theta(x) \right].$$

The second term assigns credit to earlier draft decisions through their influence on future verification boundaries and hence on which decoding contexts are subsequently visited. Estimating this term would require trajectory-level credit assignment across multiple speculative-decoding blocks, rather than the block-level credit assignment studied in this work, and would introduce additional long-horizon Monte Carlo variance and result in unstable training according to previous works about on-policy distillation (Agarwal et al., 2024). We therefore optimize the conditional block objective in Equation 19, while refreshing the context distribution through repeated on-policy rollout.

We now derive $\nabla_\theta J_\theta(x)$ for a fixed decoding context x. The outer expectation over $x \sim \mathcal{P}_{\bar{\theta}}$ can then be restored according to Equation 19. According to the definition of decoding context distribution $\mathcal{P}$ referring to Draft-OPD, $\mathcal{P}$ also depends on the draft model q_θ. Write $m_t(v) := \min\{p_t(v), q_{\theta,t}(v)\}$. For every length-$k$ prefix, its proposal probability multiplied by its survival probability satisfies

$$q_\theta(\mathbf{y}_{1:k} \mid \mathbf{x}) \prod_{j=1}^{k} \alpha_j(y_j) = \prod_{j=1}^{k} m_j(y_j). \tag{20}$$

Consequently, the conditional objective can be written as the finite sum

$$\mathcal{J}_\theta(\mathbf{x}) = \sum_{k=1}^{L} \sum_{\mathbf{y}_{1:k} \in \mathcal{V}^k} \prod_{j=1}^{k} m_j(y_j). \tag{21}$$

The target model is fixed, and, away from the measure-zero boundaries $p_t(v) = q_{\theta,t}(v)$,

$$\nabla_\theta m_t(v) = \mathbb{1}_{p_t(v) > q_{\theta,t}(v)} \nabla_\theta q_{\theta,t}(v) = m_t(v) \mathbb{1}_{p_t(v) > q_{\theta,t}(v)} \nabla_\theta \log q_{\theta,t}(v). \tag{22}$$

Differentiating Equation 21 with the product rule and then using Equations 20 and 22 gives

$$\begin{aligned}
\nabla_\theta \mathcal{J}_\theta(\mathbf{x}) &= \sum_{k=1}^{L} \sum_{\mathbf{y}_{1:k}} \left(\prod_{j=1}^{k} m_j(y_j) \right) \sum_{t=1}^{k} \mathbb{1}_{p_t(y_t) > q_{\theta,t}(y_t)} \nabla_\theta \log q_{\theta,t}(y_t) \\
&= \mathbb{E}_{\mathbf{y} \sim q_\theta} \left[\sum_{t=1}^{L} \sum_{k=t}^{L} \left(\prod_{j=1}^{k} \alpha_j(y_j) \right) \mathbb{1}_{p_t(y_t) > q_{\theta,t}(y_t)} \nabla_\theta \log q_{\theta,t}(y_t) \right] \\
&= \mathbb{E}_{\mathbf{y} \sim q_\theta} \left[\sum_{t=1}^{L} \sum_{k=t}^{L} S_k(\mathbf{y}) G_t \right] = \mathbb{E}_{\mathbf{y} \sim q_\theta} \left[\sum_{t=1}^{L} \sum_{k=t}^{L} \widehat{\mathbf{g}}_{\mathrm{dir}}^{(k,t)} \right].
\end{aligned}$$

Here t indexes the token receiving gradient credit, while $k \geq t$ indexes the surviving prefix whose contribution is assigned to that token. Restoring the outer expectation over $\mathbf{x} \sim \mathcal{P}$ proves Equation 3.

C.3 Proof of Proposition 2

For compactness, define $G_t := \mathbb{1}_{p_t(y_t) > q_{\theta,t}(y_t)} \nabla_\theta \log q_{\theta,t}(y_t)$, $S_t := \prod_{j=1}^{t} \alpha_j(y_j)$, and $\mathcal{F}_r := \sigma(\mathbf{x}, \mathbf{y}_{<r})$. Recall that $\widehat{\mathbf{g}}_{\mathrm{dir}}^{(k,t)} = S_k G_t$ for $1 \leq t \leq k \leq L$. We show that each term in Equation 5 is the exact conditional expectation of its corresponding direct prefix-credit contribution.

First consider the current contribution $k = t$ and condition on $\mathcal{F}_t$, i.e., on $(\mathbf{x}, \mathbf{y}_{<t})$. Since S_{t-1} is then fixed and $\alpha_t(v) = 1$ whenever $p_t(v) > q_{\theta,t}(v)$,

$$\begin{aligned}
\mathbb{E}_{y_t \sim q_{\theta,t}} \left[\widehat{\mathbf{g}}_{\mathrm{dir}}^{(t,t)} \mid \mathcal{F}_t \right] &= \mathbb{E}_{y_t \sim q_{\theta,t}} [S_t G_t \mid \mathcal{F}_t] \\
&= S_{t-1} \sum_{v : p_t(v) > q_{\theta,t}(v)} \nabla_\theta q_{\theta,t}(v) \\
&= -S_{t-1} \nabla_\theta D_{\mathrm{TV}}(p_t, q_{\theta,t}) = \widehat{\mathbf{g}}_{\mathrm{marg}}^{(t,t)}.
\end{aligned} \tag{23}$$

The penultimate equality follows, almost everywhere, from

$$\nabla_\theta D_{\mathrm{TV}}(p_t, q_{\theta,t}) = \sum_{v : q_{\theta,t}(v) > p_t(v)} \nabla_\theta q_{\theta,t}(v) = - \sum_{v : p_t(v) > q_{\theta,t}(v)} \nabla_\theta q_{\theta,t}(v), \tag{24}$$

where the second identity uses $\sum_v \nabla_\theta q_{\theta,t}(v) = 0$.

For a future contribution $k > t$, condition on $\mathcal{F}_k$, i.e., on $(\mathbf{x}, \mathbf{y}_{<k})$. Both S_{k-1} and G_t are measurable with respect to this prefix. Hence, by Equation 4,

$$\begin{aligned}
\mathbb{E}_{y_k \sim q_{\theta,k}} \left[\widehat{\mathbf{g}}_{\mathrm{dir}}^{(k,t)} \mid \mathcal{F}_k \right] &= \mathbb{E}_{y_k \sim q_{\theta,k}} [S_k G_t \mid \mathcal{F}_k] \\
&= S_{k-1} G_t \, \mathbb{E}_{y_k \sim q_{\theta,k}} [\alpha_k(y_k)] \\
&= A_k S_{k-1} G_t = \widehat{\mathbf{g}}_{\mathrm{marg}}^{(k,t)}.
\end{aligned} \tag{25}$$

Combining Equations 23 and 25, for every $1 \le t \le k \le L$,

$$\widehat{\mathbf{g}}^{(k,t)}_{\mathrm{marg}} = \mathbb{E}\left[\widehat{\mathbf{g}}^{(k,t)}_{\mathrm{dir}} \mid \mathcal{F}_k\right]. \tag{26}$$

Taking expectations in Equation 26, summing over t and k, and using Proposition 1 gives

$$\mathbb{E}\left[\sum_{t=1}^{L}\sum_{k=t}^{L}\widehat{\mathbf{g}}^{(k,t)}_{\mathrm{marg}}\right] = \mathbb{E}\left[\sum_{t=1}^{L}\sum_{k=t}^{L}\widehat{\mathbf{g}}^{(k,t)}_{\mathrm{dir}}\right] = \nabla_\theta \mathcal{J}_\theta. \tag{27}$$

It remains to establish the termwise variance claim. Fix $1 \le t \le k \le L$ and any fixed direction $\mathbf{a}$ of compatible dimension, and assume the corresponding scalar gradient contribution has finite second moment. Because $\mathcal{F}_t \subseteq \mathcal{F}_k$, the conditional law of total variance gives

$$\begin{aligned}
&\mathrm{Var}\left(\mathbf{a}^\top \widehat{\mathbf{g}}^{(k,t)}_{\mathrm{dir}} \mid \mathcal{F}_t\right) \\
&\quad = \mathrm{Var}\left(\mathbb{E}\left[\mathbf{a}^\top \widehat{\mathbf{g}}^{(k,t)}_{\mathrm{dir}} \mid \mathcal{F}_k\right] \mid \mathcal{F}_t\right) + \mathbb{E}\left[\mathrm{Var}\left(\mathbf{a}^\top \widehat{\mathbf{g}}^{(k,t)}_{\mathrm{dir}} \mid \mathcal{F}_k\right) \mid \mathcal{F}_t\right] \\
&\quad = \mathrm{Var}\left(\mathbf{a}^\top \widehat{\mathbf{g}}^{(k,t)}_{\mathrm{marg}} \mid \mathcal{F}_t\right) + \mathbb{E}\left[\mathrm{Var}\left(\mathbf{a}^\top \widehat{\mathbf{g}}^{(k,t)}_{\mathrm{dir}} \mid \mathcal{F}_k\right) \mid \mathcal{F}_t\right].
\end{aligned}$$

The second term on the right-hand side is nonnegative, and therefore

$$\mathrm{Var}\left(\mathbf{a}^\top \widehat{\mathbf{g}}^{(k,t)}_{\mathrm{marg}} \mid \mathbf{x}, \mathbf{y}_{<t}\right) \le \mathrm{Var}\left(\mathbf{a}^\top \widehat{\mathbf{g}}^{(k,t)}_{\mathrm{dir}} \mid \mathbf{x}, \mathbf{y}_{<t}\right).$$

Thus, marginalization removes the single-token sampling variance associated with each current or future prefix-credit contribution. This is a termwise guarantee for each (k, t); it does not imply a general variance ordering after summing correlated contributions over k and t.

C.4 Proof of Corollary 1

We first prove Equation 6. Conditioned on $(\mathbf{x}, \mathbf{y}_{<t})$, define the sampled over-sampled-token score term

$$\widetilde{G}_t := \mathbb{1}_{p_t(y_t) < q_{\theta,t}(y_t)} \nabla_\theta \log q_{\theta,t}(y_t).$$

Then the score-function identity gives

$$\begin{aligned}
\mathbb{E}[\widetilde{G}_t \mid \mathbf{x}, \mathbf{y}_{<t}] &= \sum_{v : p_t(v) < q_{\theta,t}(v)} q_{\theta,t}(v) \nabla_\theta \log q_{\theta,t}(v) \\
&= \sum_{v : p_t(v) < q_{\theta,t}(v)} \nabla_\theta q_{\theta,t}(v) \\
&= \nabla_\theta D_{\mathrm{TV}}(p_t, q_{\theta,t}),
\end{aligned}$$

where the last equality is valid almost everywhere and follows from Equation 24.

Because $S_{t-1}(\mathbf{y})$ depends only on $(\mathbf{x}, \mathbf{y}_{<t})$,

$$\mathbb{E}\left[S_{t-1}(\mathbf{y})\left(\nabla_\theta D_{\mathrm{TV}}(p_t, q_{\theta,t}) - \widetilde{G}_t\right)\right] = 0. \tag{28}$$

Summing Equation 28 over t and multiplying by any $\beta \in [0, 1]$ shows that the over-sampled-token correction added once per token position has zero expectation. Together with Equation 27, this yields

$$\mathbb{E}[\widehat{\mathbf{g}}_\beta] = \mathbb{E}\left[\sum_{t=1}^{L}\sum_{k=t}^{L}\widehat{\mathbf{g}}^{(k,t)}_{\mathrm{marg}}\right] = \nabla_\theta \mathcal{J}_\theta,$$

which proves Corollary 1 for the sampled over-sampled-token representation in Equation 6.

C.5 Discussion of Over-Sampled Token Penalty

We further discuss why the dense TVD gradient alone is not always desirable, and why redistributing part of it to the sampled over-sampled-token penalty can improve optimization, even though the

sampled representation itself introduces additional Monte Carlo variance. For a fixed decoding context and draft prefix $h_t = (x, y_{<t})$, denote

$$d_t := \nabla_\theta D_{\mathrm{TV}}(p_t, q_{\theta,t}), \qquad \widetilde{G}_t := \mathbb{1}_{p_t(y_t)<q_{\theta,t}(y_t)} \nabla_\theta \log q_{\theta,t}(y_t).$$

Equation 6 gives

$$\mathbb{E}[\widetilde{G}_t \mid h_t] = d_t.$$

Therefore, the corresponding current-token TVD contribution in Equation 7 can be written as

$$-S_{t-1}\big[(1-\beta)d_t + \beta\widetilde{G}_t\big].$$

Hence, β does not introduce a new optimization signal in expectation, but controls how the same expected TVD gradient is represented: $\beta = 0$ uses the fully dense and analytically computed TVD gradient, whereas increasing β redistributes an increasing fraction of this gradient to an explicit score-function update on the actually sampled over-sampled token.

Why the dense TVD gradient alone can be suboptimal. An important property of TVD is that its gradient remains strong even when the draft and target distributions become close. To see this more clearly, consider the gradient with respect to the draft logits z_t, with $q_{\theta,t} = \mathrm{softmax}(z_t)$, and define

$$\mathcal{O}_t := \{v : q_{\theta,t}(v) > p_t(v)\}, \qquad Q_{\mathcal{O}_t} := \sum_{v\in\mathcal{O}_t} q_{\theta,t}(v).$$

Away from the boundaries $p_t(v) = q_{\theta,t}(v)$,

$$D_{\mathrm{TV}}(p_t, q_{\theta,t}) = \sum_{v\in\mathcal{O}_t} \big(q_{\theta,t}(v) - p_t(v)\big),$$

and its logit gradient is

$$\frac{\partial D_{\mathrm{TV}}(p_t, q_{\theta,t})}{\partial z_t(v)} = q_{\theta,t}(v)\left(\mathbb{1}_{v\in\mathcal{O}_t} - Q_{\mathcal{O}_t}\right). \tag{29}$$

Unlike the forward-KL logit gradient,

$$\nabla_{z_t} D_{\mathrm{KL}}(p_t \| q_{\theta,t}) = q_{\theta,t} - p_t,$$

whose magnitude continuously vanishes as $q_{\theta,t} \to p_t$, the TVD gradient in Equation 29 does not shrink merely because the probability discrepancy $|q_{\theta,t}(v) - p_t(v)|$ becomes small. This non-vanishing property is beneficial for directly optimizing speculative acceptance, since

$$A_t = 1 - D_{\mathrm{TV}}(p_t, q_{\theta,t}),$$

and therefore TVD continues to provide an effective optimization signal even in the late stage of training.

However, the same property can also introduce an undesirable late-stage effect. When the draft distribution is already close to the target distribution, many vocabulary coordinates may have only tiny probability discrepancies, while the dense TVD gradient still applies non-negligible updates according to the sign of these discrepancies. Moreover, the active over-sampled set $\mathcal{O}_t$ depends on the sign of $q_{\theta,t}(v) - p_t(v)$. For coordinates close to equality, small changes in the sampled prefix, model parameters, or local distribution can switch a token between the over- and under-sampled sets, changing the direction of its TVD gradient. Consequently, across different on-policy sampled prefixes, a large number of individually small discrepancies can induce fluctuating vocabulary-wide gradient directions. We refer to this effect as *late-stage dense-gradient jitter*. It provides a possible explanation for the empirical observation in Figure 5 that the normalized gradient variance of *w/o penalty* grows substantially faster in the later stage of training, even though it uses the analytically marginalized TVD gradient.

Redistributing TVD gradient toward trajectory-relevant tokens. The over-sampled-token penalty alleviates this issue by reducing the strength of the dense vocabulary-wide TVD component from d_t to $(1-\beta)d_t$, while reallocating the remaining fraction to $\widetilde{G}_t$. For an actually sampled over-sampled token y_t, the logit-space score is

$$\nabla_{z_t} \log q_{\theta,t}(y_t) = e_{y_t} - q_{\theta,t},$$

where e_{y_t} is the one-hot vector of the sampled token. Thus, compared with the dense TVD gradient, the sampled representation places a much stronger direct correction on the token that actually appears in the current on-policy trajectory. It does not completely eliminate updates to unsampled logits because of the softmax coupling through $-q_{\theta,t}$, but it substantially reduces the vocabulary-wide component and concentrates more of the optimization signal on the sampled coordinate.

Such a redistribution is particularly well aligned with speculative decoding. For an over-sampled token $v \in \mathcal{O}_t$, its acceptance probability is

$$\alpha_t(v) = \frac{p_t(v)}{q_{\theta,t}(v)} < 1.$$

Its contribution to the expected rejection probability is

$$q_{\theta,t}(v)\left(1 - \frac{p_t(v)}{q_{\theta,t}(v)}\right) = q_{\theta,t}(v) - p_t(v).$$

Summing over all over-sampled tokens gives

$$\sum_{v \in \mathcal{O}_t} \big(q_{\theta,t}(v) - p_t(v)\big) = D_{\mathrm{TV}}(p_t, q_{\theta,t}) = 1 - A_t.$$

Therefore, over-sampled tokens exactly carry the rejection mass of speculative verification. Moreover, because on-policy trajectories sample $y_t \sim q_{\theta,t}$, over-sampled tokens with large draft probability are encountered more frequently and can repeatedly become verification bottlenecks. Assigning an explicit penalty to the actually sampled over-sampled token therefore focuses optimization on discrepancies that are directly realized by the current draft trajectory, rather than distributing the entire correction over all vocabulary coordinates.

Variance trade-off and control-variate interpretation. Conditioned on h_t, d_t is deterministic, whereas $\widetilde{G}_t$ is stochastic, so replacing part of d_t by $\widetilde{G}_t$ necessarily introduces additional conditional sampling variance for this component. The benefit instead arises from its interaction with the complete stochastic gradient estimator.

Specifically, define the zero-mean correction

$$Z := \sum_{t=1}^{L} S_{t-1}(d_t - \widetilde{G}_t), \qquad \mathbb{E}[Z] = 0,$$

so that Equation 7 can be written as

$$\widehat{\mathbf{g}}_\beta = \widehat{\mathbf{g}}_{\mathrm{marg}} + \beta Z.$$

For any fixed gradient direction a, let

$$X = a^\top \widehat{\mathbf{g}}_{\mathrm{marg}}, \qquad Z_a = a^\top Z.$$

Then

$$\mathrm{Var}(X + \beta Z_a) = \mathrm{Var}(X) + 2\beta\,\mathrm{Cov}(X, Z_a) + \beta^2 \mathrm{Var}(Z_a). \tag{30}$$

The last term in Equation 30 is the additional variance introduced by the sampled penalty, whereas the covariance term can be negative because the correction redistributes gradient mass from dense vocabulary-wide updates toward sampled-token updates that are complementary to the existing sampled gradient terms. Thus, the penalty acts as an expectation-preserving, control-variate-like *variance-shaping* mechanism rather than as a component-wise variance-reduction technique. A moderate β can reduce the variance of the complete stochastic gradient when the favorable covariance outweighs the newly introduced sampling variance, while an excessively large β eventually makes the $\beta^2\mathrm{Var}(Z_a)$ term dominate. Empirical results in Section 4.5 show that normalized gradient variance of w/o penalty increases to a much higher level than FReDraft, which can partially act as an evidence supporting the interpretation above. This also explains why an intermediate value of β performs best empirically, rather than either $\beta = 0$ or fully replacing the dense TVD contribution.

To further support above claim, we conduct an additional same-checkpoint, paired comparison of raw gradient variance between FReDraft and w/o penalty on last 3 checkpoints of FReDraft. Compared

with the variance in Figure 5, the variance here is measured on the same last 3 checkpoints (i.e. checkpoints at epoch 6, 7 and 8) of FReDraft and the variance is not normalized. The measured gradient variance of FReDraft and w/o penalty on last 3 checkpoints of FReDraft are 0.1669, 0.1642, 0.1729, and 0.3877, 0.4181 and 0.4100 respectively. The raw gradient variance of w/o penalty is much higher than that of FReDraft with the same training data and checkpoint. Considering that FReDraft introduces $\beta^2 \mathrm{Var}(Z_a)$ in the total variance compared with w/o penalty, this comparison supports the interpretation that a negative covariance contribution can more than offset the additional variance introduced by the sampled correction, yielding lower total estimator variance.. Notably, the variance measured here does not reflect practical training process for w/o penalty, because These measurements evaluate both estimators at checkpoints from the FReDraft training trajectory and they do not describe the raw variance along the separately trained w/o penalty trajectory. The comparison just aims to support above claim.

Relation to KL-based gradients. The above redistribution also retains an attractive property of TVD compared with common sampled KL gradients. For the sampled TVD representation, the stochastic weight is only the binary indicator $\mathbb{1}_{p_t(y_t)<q_{\theta,t}(y_t)}$. In logit space,

$$\|\nabla_{z_t} \log q_{\theta,t}(y_t)\|_2 = \|e_{y_t} - q_{\theta,t}\|_2 \le \sqrt{2},$$

so the per-position sampled correction is bounded and contains neither an importance ratio nor a log-ratio multiplier. In contrast, sample-based KL estimators may involve factors such as $p_t(y_t)/q_{\theta,t}(y_t)$ or $\log(q_{\theta,t}(y_t)/p_t(y_t))$, which can amplify stochastic updates when the two probabilities are highly mismatched. Therefore, FReDraft preserves the non-vanishing, verification-aligned optimization signal of TVD while using β to trade part of its dense vocabulary-wide update for a bounded and trajectory-focused sampled correction.

Overall, the over-sampled-token penalty serves two complementary roles. First, it gives frequently occurring over-sampled tokens an explicit correction that is absent from the sampled term of Equation 5. Second, it attenuates late-stage vocabulary-wide TVD updates whose directions can fluctuate when many small probability discrepancies change sign across on-policy prefixes. This provides a mechanism consistent with the empirical results: *w/o penalty* remains competitive in early training, but develops substantially larger normalized gradient variance in the later stage, whereas FReDraft with a moderate penalty coefficient maintains a lower-noise optimization trajectory and converges to a better accepted length.

C.6 Discussion about Comparison with D-PACE

We first clarify what D-PACE optimizes. For a target-generated block $\mathbf{z}^\star = (z_1^\star, \dots, z_L^\star)$, define the draft probability assigned to its token at position t as

$$r_{\theta,t} := q_\theta(z_t^\star \mid \mathbf{x}, \mathbf{z}_{<t}^\star)\,.$$

If a sequence is sampled from the draft model, the probability that it matches the first k tokens of $\mathbf{z}^\star$ is

$$\Pr_{\mathbf{z}\sim q_\theta}(\mathbf{z}_{1:k} = \mathbf{z}_{1:k}^\star \mid \mathbf{x}) = \prod_{i=1}^{k} r_{\theta,i}.$$

Therefore, the expected length for which the fixed target-generated block is reachable under the draft distribution is

$$\mathcal{R}_\theta(\mathbf{x}, \mathbf{z}^\star) := \sum_{k=1}^{L} \prod_{i=1}^{k} r_{\theta,i}. \tag{31}$$

Its gradient can be written as

$$\nabla_\theta \mathcal{R}_\theta = \sum_{t=1}^{L} W_t \nabla_\theta \log r_{\theta,t}, \qquad W_t := \sum_{k=t}^{L} \prod_{i=1}^{k} r_{\theta,i}.$$

Consequently, without smoothing, the detached weighted cross-entropy

$$\mathcal{L}_{\mathrm{reach}}(\theta) = -\sum_{t=1}^{L} \mathrm{sg}(W_t) \log r_{\theta,t}$$

has gradient $-\nabla_\theta \mathcal{R}_\theta$. This shows that the position weights of D-PACE are obtained by optimizing the reachability of target-generated sequences under the draft distribution, rather than the accepted length of drafts sampled from the draft model. In its stabilized implementation, D-PACE computes the detached weights from $\widetilde{r}_{\theta,t} = (1-\rho)r_{\theta,t} + \rho$ while retaining the unsmoothed cross-entropy $-\log r_{\theta,t}$ (Wu et al., 2026); this is a smoothed approximation to the reachability-gradient construction above.

We next relate this reachability objective to our proposed verification-aligned objective. For a fixed context $\mathbf{x}$, Equation 21 gives

$$\mathcal{J}_\theta(\mathbf{x}) = \sum_{k=1}^{L} \sum_{\mathbf{z}_{1:k}} \prod_{i=1}^{k} \min\{p_i(z_i), q_{\theta,i}(z_i)\}.$$

Using the target model as an importance-sampling proposal for every length-k prefix, we obtain the exact identity

$$\begin{aligned}\mathcal{J}_\theta(\mathbf{x}) &= \sum_{k=1}^{L} \mathbb{E}_{\mathbf{z}_{1:k}\sim p}\left[\prod_{i=1}^{k} \frac{\min\{p_i(z_i), q_{\theta,i}(z_i)\}}{p_i(z_i)}\right] \\ &= \mathbb{E}_{\mathbf{z}^\star\sim p}\left[\sum_{k=1}^{L}\prod_{i=1}^{k} \min\left\{1, \frac{q_{\theta,i}(z_i^\star)}{p_i(z_i^\star)}\right\}\right]. \end{aligned} \tag{32}$$

Now make the one-hot-target approximation that the target probability of every target-generated token is 1.0, i.e., $p_i(z_i^\star) = 1$. Since $q_{\theta,i}(z_i^\star) \le 1$, Equation 32 reduces to

$$\mathcal{J}_\theta(\mathbf{x}) \approx \mathbb{E}_{\mathbf{z}^\star\sim p}\left[\sum_{k=1}^{L}\prod_{i=1}^{k} q_{\theta,i}(z_i^\star)\right] = \mathbb{E}_{\mathbf{z}^\star\sim p}\left[\mathcal{R}_\theta(\mathbf{x}, \mathbf{z}^\star)\right].$$

Thus, the D-PACE reachability objective is an importance-sampling estimation and one-hot-target approximation to our objective after replacing the target distribution on target-generated tokens by a point mass. Empirical averaging over a corpus of target-generated blocks provides the corresponding Monte Carlo estimate. However, importance sampling introduces larger variance and the one-hot-target approximation also introduces bias.

Counterexample under the target-generated training distribution. The mismatch between reference reachability and verification-aligned accepted length persists even after averaging over the complete target-generated training-data distribution. Consider a length-2 block with vocabulary $\mathcal{V} = \{A, B\}$. Assume that the target distribution is independent across positions and satisfies

$$p(A) = 0.6, \qquad p(B) = 0.4.$$

The corresponding target-generated training distribution is therefore

$$p(AA) = 0.36, \qquad p(AB) = 0.24, \qquad p(BA) = 0.24, \qquad p(BB) = 0.16.$$

Consider two position-independent draft distributions

$$q^{(1)} = (0.6, 0.4), \qquad q^{(2)} = (0.9, 0.1),$$

where $q^{(1)} = p$ exactly matches the target distribution.

For an arbitrary position-independent draft distribution q, let

$$c(q) := \mathbb{E}_{z\sim p}[q(z)] = \sum_{v\in\mathcal{V}} p(v)q(v).$$

Averaging the reachability in Equation 31 over the target-generated training distribution gives

$$\mathbb{E}_{\mathbf{z}^\star\sim p}\left[\mathcal{R}_\theta(\mathbf{x}, \mathbf{z}^\star)\right] = c(q) + c(q)^2.$$

For the two draft distributions above,

$$\begin{aligned} c(q^{(1)}) &= 0.6 \times 0.6 + 0.4 \times 0.4 = 0.52, \\ c(q^{(2)}) &= 0.6 \times 0.9 + 0.4 \times 0.1 = 0.58, \end{aligned}$$

and hence

$$\mathbb{E}_{\mathbf{z}^\star\sim p}\left[\mathcal{R}(q^{(1)},\mathbf{z}^\star)\right] = 0.52 + 0.52^2 = 0.7904,$$
$$\mathbb{E}_{\mathbf{z}^\star\sim p}\left[\mathcal{R}(q^{(2)},\mathbf{z}^\star)\right] = 0.58 + 0.58^2 = 0.9164.$$

Thus, after averaging over all target-generated training blocks, the reachability objective prefers $q^{(2)}$ over $q^{(1)}$.

However, speculative verification depends on the probability overlap between the target and draft distributions. By Equation 4, the expected token acceptance rate is

$$A(q) = \sum_{v\in\mathcal{V}} \min\{p(v), q(v)\}.$$

Since the two positions are independent and identically distributed, the expected accepted length is

$$\mathcal{J}(q) = A(q) + A(q)^2.$$

For $q^{(1)} = p$, we have $A(q^{(1)}) = 1$ and therefore

$$\mathcal{J}(q^{(1)}) = 2.$$

For $q^{(2)}$,

$$A(q^{(2)}) = \min\{0.6, 0.9\} + \min\{0.4, 0.1\} = 0.7,$$

which gives

$$\mathcal{J}(q^{(2)}) = 0.7 + 0.7^2 = 1.19.$$

Consequently,

$$\mathbb{E}_{\mathbf{z}^\star\sim p}\left[\mathcal{R}(q^{(2)},\mathbf{z}^\star)\right] > \mathbb{E}_{\mathbf{z}^\star\sim p}\left[\mathcal{R}(q^{(1)},\mathbf{z}^\star)\right], \qquad \mathcal{J}(q^{(2)}) < \mathcal{J}(q^{(1)}).$$

Therefore, averaging reference reachability over the target-generated training distribution does not in general preserve the preference induced by the true verification objective. In particular, $q^{(1)}$ already achieves perfect speculative acceptance because it exactly matches p, whereas the reachability surrogate assigns a larger value to $q^{(2)}$.

The underlying reason is that target-data averaging turns reference-token confidence into $\sum_v p(v)q(v)$, which rewards concentrating additional draft probability on frequently sampled target tokens. Speculative verification instead depends on $\sum_v \min\{p(v), q(v)\}$: once $q(v)$ exceeds $p(v)$, assigning additional probability to v no longer improves its acceptance and instead creates excess proposal mass that can cause rejection. Thus, reference reachability can mistake confidence on likely target-generated tokens for verification quality, whereas the verification-aligned objective directly evaluates the probability overlap encountered by draft-generated proposals. This counterexample isolates the mismatch of the unsmoothed reachability objective in Equation 31; the stabilized D-PACE implementation additionally smooths the detached reachability weights as described above.

D Experiment Details

D.1 Baselines

We select the baselines in Section 4.1 because they are recent SOTA representative off-policy verification-aware methods and on-policy method. We do not choose other methods discussed in Appendix A.2 (e.g. VSD, GTO, VAT, etc.), because they are either designed specifically for Eagle-style tree-structured drafts and there is no public open-source DFlash implementation, or they are concurrent works. For the selected baselines, we summarize their implementation details as follows:

Eagle-3. We refer to its official open-source code [1] to implement the training pipeline. We use both Qwen3-4B-Instruct-100K and the 16K prompts released by Draft-OPD to SFT Eagle-3.

[1] https://github.com/SafeAILab/EAGLE

DFlash. We use the official open-source weights [2] trained on Qwen3-4B-Instruct-100K as the initial model and further SFT it on 16K prompts released by Draft-OPD referring to the implementation provided by SpecForge [3].

D-PACE. We use the official open-source weights trained on Qwen3-4B-Instruct-100K as the initial model and further train it on 16K prompts released by Draft-OPD referring to the official open-source code [4].

LK-Losses. We use the official open-source weights [5] trained on Qwen3-4B-Instruct-100K as the initial model and further SFT it on 16K prompts released by Draft-OPD referring to the implementation provided by SpecForge [6].

Draft-OPD. We use the official open-source weights [7] trained on Qwen3-4B-Instruct-100K as the initial model and further train it on 16K prompts released by Draft-OPD referring to the official open-source code [8].

D.2 Ablation Variants

All ablations use the same SFT initialization as the full method. We implement the four variants as follows.

w/o marg. This variant implements the direct Monte Carlo estimator in Equation 3. For each block, it computes $S_t = \prod_{j=1}^{t} \alpha_j(y_j)$ and the inclusive reverse cumulative sum

$$U_t = \sum_{k=t}^{L} S_k.$$

The minimized block loss is

$$\mathcal{L}_{\text{w/o marg}} = -\sum_{t=1}^{L} \mathrm{sg}\big(U_t \mathbb{1}_{p_t(y_t) > q_{\theta,t}(y_t)}\big) \log q_{\theta,t}(y_t).$$

In particular, $U_L = S_L$ is retained. The variant has neither a dense TVD term nor an over-sampled-token penalty; it uses only the exact target and draft probabilities of sampled tokens.

w/o penalty. We set $\beta = 0$ in the full implementation, and define

$$b_t = \sum_{k=t+1}^{L} A_k S_{k-1}.$$

Then the resulting w/o penalty loss is

$$\mathcal{L}_{\text{w/o penalty}} = \sum_{t=1}^{L} \left[-\mathrm{sg}\big(b_t \mathbb{1}_{p_t(y_t) > q_{\theta,t}(y_t)}\big) \log q_{\theta,t}(y_t) + \mathrm{sg}(S_{t-1}) D_{\mathrm{TV}}(p_t, q_{\theta,t}) \right],$$

which corresponds to the marginalized estimator in Equation 5 under the compressed distribution representation. Thus, over-sampled tokens still participate in the dense distribution-alignment term, but receive no explicit sampled-token correction.

[2] https://github.com/z-lab/dflash
[3] https://github.com/sgl-project/SpecForge/issues/465
[4] https://github.com/Lucas-TY/D-PACE
[5] https://github.com/z-lab/dflash
[6] https://github.com/sgl-project/SpecForge/issues/485
[7] https://github.com/z-lab/dflash
[8] https://github.com/Simplified-Reasoning/Draft-OPD

FReDraft-ED. This variant removes prefix-aware credit assignment and replaces it with the static exponential weight $e_t = \rho^{t-1}$, using $\rho = 0.8$ as in the position-decay configuration. It retains the same TVD-gradient redistribution as the full method:

$$\mathcal{L}_{\mathrm{ED}} = \sum_{t=1}^{L} e_t \left[\beta \mathbb{1}_{p_t(y_t) < q_{\theta,t}(y_t)} \log q_{\theta,t}(y_t) + (1-\beta) D_{\mathrm{TV}}(p_t, q_{\theta,t})\right].$$

The negative expected gradient of the bracketed expression is the exponentially weighted ascent direction for distribution overlap. Neither the sampled prefix survival S_{t-1} nor the acceptance-to-go b_t is used as a loss coefficient, thereby isolating the effect of adaptive prefix-aware weighting.

FReDraft-KL. This variant retains the sampled prefix survival S_{t-1} and marginalized $A_k S_{k-1}$ for sampled token and S_{t-1} for vocabulary gradient, but replaces the dense TVD loss with the mixed KL recipe used by Draft-OPD. For a proposal that is accepted in the realized verification, we use a forward KL on the sampled token. For a rejected or discarded proposal, we use the reverse KL.

Let κ_t select forward KL for accepted proposals and reverse KL otherwise. The implemented loss is

$$\mathcal{L}_{\mathrm{KL}} = \sum_{t=1}^{L} \left[-\mathrm{sg}\big(b_t \mathbb{1}_{p_t(y_t) > q_{\theta,t}(y_t)}\big) \log q_{\theta,t}(y_t) + \mathrm{sg}(S_{t-1})\kappa_t \right].$$

We do not apply the β correction in this variant because the expectation-preserving identity in Equation 6 is specific to TVD. This construction therefore isolates the choice of distribution discrepancy while preserving FReDraft's prefix-aware credit assignment. Compared with w/o penalty, the only difference is that the dense TVD term is replaced by a mixed KL term.

D.3 Benchmarks

We use the public evaluation splits of GSM8K[9], MATH-500[10], AIME25[11], MBPP[12], HumanEval[13], SWE-Lite (SWE-bench Lite)[14], and MT-Bench[15]. Table 5 summarizes the evaluation size of each benchmark. For MT-Bench, one prompt denotes one two-turn conversation; thus, its 80 prompts contain 160 user turns in total.

Table 5: Statistics of the evaluation benchmarks.

Category	Benchmark	Number of prompts
Mathematics	GSM8K	1,319
	MATH-500	500
	AIME25	30
Coding	MBPP	500
	HumanEval	164
	SWE-Lite	300
Chat	MT-Bench	80

D.4 Normalized Gradient Variance

We use normalized gradient variance to quantify the amount of Monte Carlo noise in the stochastic learning signal produced by different training objectives and gradient estimators. Raw gradient

[9] https://github.com/openai/grade-school-math

[10] https://huggingface.co/datasets/HuggingFaceH4/MATH-500

[11] https://huggingface.co/datasets/opencompass/AIME2025

[12] https://github.com/google-research/google-research/tree/master/mbpp

[13] https://github.com/openai/human-eval

[14] https://www.swebench.com/lite.html

[15] https://github.com/lm-sys/FastChat/tree/main/fastchat/llm_judge/data/mt_bench

variance alone is not sufficient for this comparison because it scales quadratically with the overall gradient magnitude: an estimator with uniformly smaller gradients can exhibit a smaller raw variance even when its noise is large relative to its useful optimization signal. We therefore normalize the gradient variance by the squared magnitude of the corresponding mean gradient.

Concretely, for each measurement context c and a model checkpoint, we keep the input context and model parameters unchanged and independently sample N draft sequences from the current draft distribution. Let

$$\mathbf{g}_c^{(m)} = \nabla_\theta \mathcal{L}^{(m)}, \qquad m = 1, \dots, N,$$

denote gradient at draft position t under the m-th Monte Carlo sample. The sample mean gradient is

$$\bar{\mathbf{g}}_{c,t} = \frac{1}{N} \sum_{m=1}^{N} \mathbf{g}_c^{(m)}.$$

We estimate the trace of the gradient covariance by

$$\widehat{V}_c = \frac{1}{N-1} \sum_{m=1}^{N} \left\| \mathbf{g}_c^{(m)} - \bar{\mathbf{g}}_c \right\|_2^2.$$

This quantity equals the sum of the coordinate-wise gradient variances and therefore measures the total stochastic fluctuation of the gradient at position t.

The squared norm of the sample mean, $\|\bar{\mathbf{g}}_{c,t}\|_2^2$, is an upward-biased estimator of the squared norm of the true mean gradient because the sample mean itself contains Monte Carlo noise. We remove this finite-sample bias using

$$\widehat{S}_c = \|\bar{\mathbf{g}}_c\|_2^2 - \frac{\widehat{V}_c}{N},$$

which estimates $\|\mathbb{E}[\mathbf{g}_c]\|_2^2$. We then aggregate signal and noise before taking their ratio and report

$$\widehat{\mathrm{NGV}} = \frac{\sum_c \widehat{V}_c}{\sum_c \max\left(\widehat{S}_c, 0\right) + \epsilon},$$

where $\epsilon = 10^{-5}$ is a small constant for numerical stability. Aggregating before normalization is important: individually dividing by $\widehat{S}_c$ can become unstable for positions whose expected gradient is close to zero and can cause a few nearly-zero-signal positions to dominate the reported statistic. The above form is equivalently the ratio between the trace of the covariance of the concatenated parameter gradient vector and the squared norm of its mean, i.e., the inverse of the squared-gradient signal-to-noise ratio. A smaller normalized gradient variance therefore indicates less sampling noise per unit of useful gradient signal.

For every measurement, all N Monte Carlo samples use the same context and the same model parameters and differ only in the sampled draft tokens. We set $N = 256$ and evaluate the normalized gradient variance every 250 training steps. We use the same measurement contexts and the same number of Monte Carlo samples for all methods. This protocol isolates stochasticity introduced by draft-token sampling rather than changes in data or model state. Normalization further makes the statistic invariant to a global rescaling of the loss or gradient: if every gradient is multiplied by a constant a, both the numerator and denominator are multiplied by a^2. The statistic is approximately invariant to global gradient rescaling when the stabilization constant is negligible. Consequently, the metric is more suitable than raw variance for comparisons across training stages or methods whose gradient magnitudes differ.

For the direct and marginalized estimators of the same expected accepted-length objective, both estimators target the same expected gradient, so a smaller normalized gradient variance directly indicates a lower-noise Monte Carlo estimator of the same optimization signal. When comparing objectives that do not share exactly the same expected gradient (e.g., external baselines), normalized gradient variance should instead be interpreted as a scale-normalized diagnostic of optimization noise, rather than by itself as evidence that one objective is superior. In the main text, we therefore use it together with convergence and accepted-length results to characterize optimization efficiency.

Table 6: Throughput (tok/s) and average acceptance length (τ) on SGLang.

Task	Method	Q3-4B						Q3-8B					
		Concurrency					Avg. τ	Concurrency					Avg. τ
		1	4	8	16	32		1	4	8	16	32	
AIME25	DFlash	478	1440	2574	4558	6234	6.06	326	1003	1876	3192	5123	7.22
	Draft-OPD	508	1560	2748	4862	6703	7.24	368	1124	2003	3578	5601	7.89
	FReDraft	**532**	**1594**	**2852**	**5005**	**6830**	**7.96**	**397**	**1206**	**2179**	**3796**	**5914**	**8.18**
HumanEval	DFlash	490	1468	2627	4612	6323	6.10	313	989	1832	3099	5003	6.40
	Draft-OPD	507	1512	2698	4737	6525	6.43	336	1048	1930	3271	5274	6.66
	FReDraft	**523**	**1565**	**2784**	**4889**	**6690**	**7.18**	**348**	**1092**	**2013**	**3392**	**5462**	**6.85**
MT-Bench	DFlash	410	1308	2496	4277	5901	5.22	211	761	1353	2217	4138	3.38
	Draft-OPD	418	1332	2536	4333	5972	5.27	229	820	1477	2398	4492	3.50
	FReDraft	**423**	**1347**	**2566**	**4400**	**6047**	**5.45**	**237**	**852**	**1512**	**2462**	**4596**	**3.65**

E Extra Experiment Results

E.1 Deployment Efficiency on SGLang.

Referring to DFlash (Chen et al., 2026a) and Draft-OPD (Lei et al., 2026), we also evaluate FReDraft on open-source inference framework SGLang (Zheng et al., 2024) to demonstrate the efficiency of our method in practical deployment scenarios. We report the average accepted length and throughput under concurrency across 1 to 32 in Table 6. All experiments are conducted on a single A100 GPU. FReDraft consistently outperforms DFlash and Draft-OPD in terms of average accepted length and throughput under different concurrency, demonstrating the effectiveness of our proposed draft model training method in practical deployment.

E.2 Spearman Correlation of Different Metrics to Speedup Ratio

We examine how closely different draft-quality metrics track decoding speedup. Using Qwen3-4B as the target and the DFlash drafter trained with FReDraft, we collect query-level statistics on the seven benchmarks in Appendix D.3. We use temperature 1.0 with thinking mode disabled, so that the comparison includes stochastic verification. For each query, speedup is the ratio of target-only decoding time to speculative decoding time under the same evaluation configuration. We compare the following four metrics:

- **Average acceptance rate.** We average the token-level acceptance probabilities $\alpha_t(y_t)$ over all positions in all collected draft blocks, including positions after the first rejection. This measures token-level alignment, as targeted by LK-Losses, without weighting a position by the probability that verification reaches it.
- **Expected reachable length.** We average $\mathcal{R}_\theta(\mathbf{x}, \mathbf{z}^\star)$ from Equation 31 over length-L chunks of target-only responses to the same queries. This is the unsmoothed reachability quantity underlying D-PACE; it measures the likelihood of matching a target-generated prefix under the draft distribution.
- **Accepted length.** We average the realized number of accepted draft tokens per verification call, denoted by τ. As in the main experiments, this excludes the target-generated correction or bonus token and the anchor token.
- **Expected accepted length.** We average $\bar{\tau}_\theta(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{L} \prod_{j=1}^{t} \alpha_j(y_j)$ over the same collected draft blocks. This is the conditional expectation in Equation 2, which FReDraft optimizes under on-policy draft sampling.

We compute Spearman's rank correlation ρ_s between each metric and query-level speedup, assigning average ranks to ties. Table 7 reports correlations within each benchmark and across all pooled queries; the pooled value is computed from the pooled ranks, rather than by averaging benchmark-level correlations.

Table 7: Spearman correlation with query-level speedup on Qwen3-4B at temperature 1.0.

Benchmark	Average acceptance rate	Expected reachable length	Expected accepted length	Accepted length
GSM8K	0.73	0.85	0.97	0.98
MATH-500	0.70	0.82	0.96	0.98
AIME25	0.66	0.79	0.95	0.97
MBPP	0.74	0.86	0.97	0.99
HumanEval	0.72	0.84	0.97	0.98
SWE-Lite	0.63	0.77	0.94	0.96
MT-Bench	0.60	0.74	0.93	0.95
Pooled	0.69	0.81	0.96	0.98

The correlations follow the same ordering across benchmarks: average acceptance rate $<$ expected reachable length $<$ expected accepted length $<$ realized accepted length. In particular, expected accepted length is closely associated with speedup, with a pooled correlation of 0.96, compared with 0.98 for realized accepted length. The weaker association of average acceptance rate is consistent with its omission of first-rejection truncation: improving a late token has limited effect when its prefix rarely survives. Expected reachable length accounts for prefix structure, but evaluates target-generated chunks through their draft probabilities, rather than the actual verification ratios of on-policy drafts.

Although realized accepted length has the highest correlation with the speedup of the same decoding run, it also depends on the random acceptance decisions made during verification. For a fixed context and sampled draft, $\bar{\tau}_\theta$ integrates out these decisions exactly, removing this source of sampling variance while retaining the dependence on prefix survival. Randomness from sampled drafts and visited contexts remains. Thus, expected accepted length provides a less noisy training signal than a single realized acceptance outcome while remaining closely related to the decoding quantity of interest. This motivates its use as an optimization objective; the correlation alone does not imply that it captures every hardware- or workload-dependent contribution to speedup.

E.3 Results on Thinking Mode

We further evaluate Qwen3-4B and Qwen3-8B with thinking mode enabled. We use the same trained draft checkpoints, seven benchmarks, and draft length $L = 15$ as in Table 2, and report both greedy decoding (temperature 0) and stochastic decoding (temperature 1.0). The target-only baseline also uses thinking mode, and decoding time includes the generated reasoning trace as well as the final answer. Within each comparison, all methods share the prompt template, generation limit, and stopping criteria. This evaluation tests how the trained drafters perform when the target generates explicit reasoning trajectories.

Table 8 follows the overall trend in Table 2. DFlash provides higher speedup than EAGLE-3, and on-policy training improves the mean performance of the DFlash drafter. FReDraft achieves the highest speedup on each benchmark and the largest mean accepted length in all four model–temperature configurations. For Qwen3-4B, its mean speedups are $4.34\times$ and $3.69\times$ at temperatures 0 and 1.0, respectively, compared with $4.01\times$ and $3.47\times$ for Draft-OPD. For Qwen3-8B, the corresponding values are $5.75\times$ and $4.84\times$, compared with $5.37\times$ and $4.46\times$.

All methods also have higher speedup and mean accepted length than in the corresponding non-thinking configuration. One possible explanation is that explicit reasoning changes the generated token patterns in ways that make longer prefixes easier to predict; longer responses can additionally amortize fixed decoding overhead. These are possible contributing factors, rather than mechanisms isolated by this comparison. The continued advantage of FReDraft is consistent with prefix-aware optimization remaining useful when thinking mode changes the distribution of decoding contexts.

Table 8: Decoding speedup and average accepted draft length (τ) on Qwen3 models with thinking mode enabled.

Model	Method	Speedup Ratio (SR)							Mean	
		Math			Code			Chat	SR	τ
		GSM8K	MATH-500	AIME25	MBPP	HumanEval	SWE-Lite	MT-Bench		
Temperature = 0										
Q3-4B	EAGLE-3	2.28×	2.12×	2.14×	1.99×	2.11×	1.89×	1.93×	2.07×	6.12
	DFlash	4.02×	4.72×	4.28×	3.52×	3.83×	2.61×	2.26×	3.61×	6.57
	D-PACE	4.21×	4.97×	4.55×	3.68×	3.76×	2.68×	2.22×	3.72×	6.75
	LK-Losses	4.40×	5.20×	4.78×	3.76×	3.97×	2.76×	2.32×	3.88×	6.94
	Draft-OPD	4.69×	5.48×	4.92×	3.85×	4.03×	2.74×	2.38×	4.01×	7.13
	FReDraft	**4.87**×	**5.89**×	**5.41**×	**4.10**×	**4.54**×	**3.01**×	**2.59**×	**4.34**×	**7.91**
Q3-8B	EAGLE-3	2.24×	2.14×	2.16×	1.95×	2.18×	2.02×	1.85×	2.08×	6.03
	DFlash	5.20×	6.26×	5.84×	4.57×	5.20×	3.17×	2.80×	4.72×	6.55
	D-PACE	5.48×	6.52×	6.02×	4.76×	5.16×	3.25×	2.73×	4.85×	6.68
	LK-Losses	5.73×	6.85×	6.35×	4.95×	5.41×	3.45×	3.03×	5.11×	6.98
	Draft-OPD	6.01×	7.19×	6.61×	5.23×	5.67×	3.64×	3.22×	5.37×	7.32
	FReDraft	**6.39**×	**7.73**×	**7.08**×	**5.61**×	**6.05**×	**3.89**×	**3.47**×	**5.75**×	**8.14**
Temperature = 1.0										
Q3-4B	EAGLE-3	2.13×	2.00×	1.96×	1.88×	1.96×	1.71×	1.88×	1.93×	5.68
	DFlash	3.68×	3.87×	3.26×	3.30×	3.57×	2.28×	2.11×	3.15×	6.02
	D-PACE	3.90×	4.05×	3.36×	3.24×	3.48×	2.32×	2.04×	3.20×	6.17
	LK-Losses	4.01×	4.20×	3.48×	3.39×	3.62×	2.38×	2.24×	3.33×	6.35
	Draft-OPD	4.23×	4.51×	3.79×	3.50×	3.70×	2.35×	2.21×	3.47×	6.52
	FReDraft	**4.36**×	**4.71**×	**4.08**×	**3.66**×	**4.13**×	**2.51**×	**2.40**×	**3.69**×	**6.94**
Q3-8B	EAGLE-3	2.11×	1.97×	1.94×	1.85×	1.98×	1.85×	1.77×	1.92×	5.39
	DFlash	4.64×	4.91×	3.97×	3.88×	4.37×	2.49×	2.39×	3.81×	5.24
	D-PACE	4.78×	5.10×	4.10×	4.03×	4.33×	2.62×	2.36×	3.90×	5.42
	LK-Losses	5.06×	5.38×	4.48×	4.22×	4.76×	2.77×	2.57×	4.18×	5.74
	Draft-OPD	5.49×	5.79×	4.74×	4.42×	5.02×	2.93×	2.80×	4.46×	6.10
	FReDraft	**5.99**×	**6.34**×	**5.11**×	**4.74**×	**5.35**×	**3.26**×	**3.10**×	**4.84**×	**6.67**

E.4 Generalization to Other Target Models

To assess generalization beyond the Qwen3 family, we repeat the comparison with Llama-3.2-3B-Instruct (Grattafiori et al., 2024)[16], abbreviated as L3-3B in Table 9. We train target-specific draft models using the same prompt sets and training recipes, regenerating the supervised responses with the new target. The drafter uses the target's tokenizer and compatible hidden-state features; the draft length remains $L = 15$, and FReDraft retains $\beta = 0.3$. We evaluate all seven benchmarks at temperatures 0 and 1.0, using autoregressive decoding by the same Llama target as the speedup baseline. This comparison evaluates the portability of the training method across target families, with a separately trained drafter for each target.

The ranking in Table 9 is consistent with the Qwen3 results. DFlash improves speedup over EAGLE-3, and both on-policy methods outperform the off-policy DFlash-based baselines on average. FReDraft obtains the highest SR and τ on all seven benchmarks at both temperatures. At temperature 0, its mean speedup increases from Draft-OPD's $3.11\times$ to $3.38\times$, while mean accepted length increases from 5.76 to 6.31. At temperature 1.0, the corresponding improvements are from $2.75\times$ to $2.98\times$ and from 5.02 to 5.48. The same qualitative advantage across the two target families supports the applicability of expected accepted-length optimization beyond the Qwen3 configuration, although the magnitude of the gain depends on the target and workload.

[16] https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct

Table 9: Decoding speedup and average accepted draft length (τ) on Llama3.2-3B-Instruct.

Model	Method	Speedup Ratio (SR)							Mean	
		Math			Code			Chat	SR	τ
		GSM8K	MATH-500	AIME25	MBPP	HumanEval	SWE-Lite	MT-Bench		
Temperature = 0										
L3-3B	EAGLE-3	1.82×	1.75×	1.61×	1.69×	1.78×	1.53×	1.66×	1.69×	4.92
	DFlash	3.08×	3.44×	2.91×	2.86×	3.03×	1.98×	1.89×	2.74×	5.13
	D-PACE	3.19×	3.61×	3.04×	2.94×	2.99×	2.03×	1.85×	2.81×	5.27
	LK-Losses	3.35×	3.80×	3.25×	3.04×	3.16×	2.10×	1.99×	2.96×	5.48
	Draft-OPD	3.60×	4.01×	3.47×	3.18×	3.32×	2.14×	2.06×	3.11×	5.76
	FReDraft	**3.87**×	**4.36**×	**3.82**×	**3.43**×	**3.64**×	**2.31**×	**2.22**×	**3.38**×	**6.31**
Temperature = 1.0										
L3-3B	EAGLE-3	1.70×	1.62×	1.48×	1.56×	1.64×	1.42×	1.53×	1.56×	4.46
	DFlash	2.79×	2.98×	2.34×	2.61×	2.74×	1.71×	1.70×	2.41×	4.49
	D-PACE	2.90×	3.12×	2.47×	2.67×	2.69×	1.78×	1.67×	2.47×	4.62
	LK-Losses	3.04×	3.27×	2.60×	2.78×	2.88×	1.84×	1.79×	2.60×	4.81
	Draft-OPD	3.23×	3.49×	2.78×	2.92×	3.02×	1.91×	1.88×	2.75×	5.02
	FReDraft	**3.46**×	**3.78**×	**3.05**×	**3.14**×	**3.31**×	**2.06**×	**2.03**×	**2.98**×	**5.48**

E.5 Wall-Clock Time Efficiency

We compare the on-policy training cost of Draft-OPD and FReDraft on Qwen3-4B using the hardware and training pipeline described in Appendix B.4. Both methods start from the same SFT checkpoint and use the same prompts, batch size, draft length, sampling configuration, and response-length limit. Rollout time includes draft generation and target verification; update time includes training replay, loss construction, backpropagation, and the optimizer update. We synchronize workers at timing boundaries and exclude evaluation and checkpoint-writing time.

To distinguish the cost of processing a draft from the amount of work needed for a query, we report amortized times at both levels. For accumulated rollout and update times T_{roll} and T_{update}, processing N_{draft} drafts from N_{query} queries, the per-draft times are $T_{\text{roll}}/N_{\text{draft}}$ and $T_{\text{update}}/N_{\text{draft}}$. The total time per query is $(T_{\text{roll}} + T_{\text{update}})/N_{\text{query}}$. These are amortized training costs, rather than the latency of an individual query under concurrent execution. We also report the mean number of drafts per query and their mean accepted length to help interpret the timing.

Table 10: Amortized on-policy training cost on Qwen3-4B. Rollout and update times are reported separately.

Metric	Draft-OPD	FReDraft
Rollout time / draft (ms)	15.0	15.0
Update time / draft (ms)	8.0	8.1
Total time per draft (ms)	23.0	23.1
Mean drafts per query	80.0	74.0
Mean accepted length τ	5.40	5.92
Rollout time / query (s)	1.200	1.110
Update time / query (s)	0.640	0.599
Total time per query (s)	1.840	1.709

Table 10 shows comparable per-draft costs for the two methods: rollout time is unchanged at the reported precision, while update time increases by approximately 1.3% for FReDraft. This is consistent with reusing the target distributions already collected during verification; the prefix-survival and penalty computations add a small amount of arithmetic to the shared training pipeline. The comparable costs are an empirical observation under this configuration, rather than a consequence guaranteed solely by using the same number of target-model forward passes.

Despite the slightly higher update cost per draft, FReDraft reduces the amortized total time per query from $1.840\,\mathrm{s}$ to $1.709\,\mathrm{s}$, a reduction of approximately 7.1%. The accompanying increase in accepted length is consistent with the decrease from 80.0 to 74.0 drafts per query, or 7.5% fewer blocks to generate, verify, and replay. For comparable response lengths, accepting more tokens per block reduces the number of verification rounds and can therefore lower both rollout and update work per query. Together with the training dynamics in Figure 5, this comparison supports the practical efficiency of the proposed objective: its additional loss computations have modest per-draft cost, while improved draft quality can reduce the amount of work required during on-policy collection.