# BV LOSS: BLOCK VERIFICATION-AWARE LOSS FOR BLOCK DIFFUSION SPECULATIVE DECODING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Diffusion drafters accelerate speculative decoding by proposing multiple tokens in parallel. Despite recent advances in speculative decoding through sequence-level drafting and verification, existing training objectives remain largely designed around token-level verification. To address this mismatch, we introduce **Block Verification-aware loss (BV loss)**, a training objective designed to maximize the expected acceptance length of a drafted sequence. BV loss is directly derived from the block verification acceptance rule, providing a principled connection between the drafter training objective and the inference-time verification mechanism at the sequence level. Across math, code, and chat benchmarks, BV loss increases the mean number of tokens accepted per verification call under block verification by 13.0–21.0% over cross-entropy loss training for DFlash and DSpark with Qwen3-4B and Qwen3-8B without changing the inference procedure. BV loss also outperforms tokenwise acceptance objectives such as TV loss and LK loss, and its gains extend to token verification and greedy decoding. These results demonstrate the benefit of training block diffusion drafters with an objective aligned with sequence-level verification, rather than optimizing each token independently.

## 1 INTRODUCTION

Speculative decoding accelerates large language model (LLM) inference by drafting candidate tokens using a lightweight drafter model and verifying them in parallel with a larger target model (Leviathan et al., 2023). Its efficiency largely depends on how many drafted tokens are accepted per verification, making both drafting and verification critical to the overall speedup. While the majority of related research focuses on improving the drafter, another line of work seeks additional speedup through better verification algorithms. Block verification (Sun et al., 2025) is a key approach that targets the verification side of speculative decoding. It introduces a sequence-level acceptance criterion and performs rejection sampling over the joint probabilities of candidate sequences. The conventional verification scheme, which we call token verification, terminates when a proposed token is rejected, causing all subsequent draft tokens to be discarded. By contrast, block verification evaluates the drafted sequence jointly and can accept tokens that would otherwise be discarded under token verification. Importantly, replacing token verification with block verification incurs negligible additional inference overhead, while achieving an equal or higher acceptance rate.

Despite the emergence of sequence-level verification in speculative decoding, existing drafters are still trained with token-level objectives. A prominent example is total variation (TV) loss, which is derived from the single-token acceptance probability under token verification Minimizing TV loss therefore provides a principled way to train a drafter for tokenwise rejection sampling. However, under block verification, acceptance is determined by the probability of the block-sized sequence, raising the question of whether token-level losses remain the appropriate training objectives.

To address this mismatch between the training objective and the verification criterion, we propose **Block Verification-aware loss (BV loss)**, a drafter training objective designed to directly maximize the expected acceptance length of a drafted sequence under block verification. Our formulation is derived from the sequence-level acceptance criterion used in block verification rather than the token-level rejection sampling. In this sense, BV loss serves as a sequence-level counterpart to TV loss: just as TV loss follows from the acceptance probability induced by token-level rejection sampling, BV loss is derived from the acceptance length induced by block verification (see Table 1).



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

Table 1: **Comparison of drafter training objectives.** CE and KL / RKL optimize token likelihood or distribution matching, whereas TV and LK loss explicitly target token-level acceptance. BV loss is instead derived from block verification and directly optimizes the expected accepted draft length, making it both verification-aware and block-aware.

| Drafter loss | Training objective | Uses teacher probabilities? | Verification-aware? | Block-aware? |
|---|---|---|---|---|
| CE | Maximize log-likelihood | ✗ | ✗ | ✗ |
| KL / RKL | Minimize KL divergence | ✓ | ✗ | ✗ |
| TV | Maximize token acceptance | ✓ | ✓ | ✗ |
| LK | Maximize token acceptance | ✓ | ✓ | ✗ |
| **BV (Ours)** | **Maximize block acceptance length** | ✓ | ✓ | ✓ |

We evaluate BV loss across multiple speculative-decoding configurations, drafter architectures, and target models. Drafters trained with BV loss consistently achieve longer accepted sequence lengths than those trained with conventional token-level objectives, demonstrating the benefit of aligning drafter training with sequence-level verification. Moreover, BV loss generalizes across different drafting and verification configurations. Drafters trained with BV loss outperform those trained with conventional objectives even under greedy decoding and token-level verification. These results show that BV loss is not only theoretically motivated by block verification, but also serves as an effective training objective across diverse speculative-decoding settings.

Our contributions are summarized below:

- **A sequence-level objective aligned with block verification.** We introduce BV loss, a sequence-level training objective derived from the block verification acceptance rule. We prove that minimizing BV loss is equivalent to maximizing the expected accepted draft length. The objective accounts for both partial and full block acceptance and recovers TV loss as the single-token special case.
- **Theoretical analysis and practical optimization.** We show that the same tokenwise acceptance rate can yield different expected accepted lengths under block verification, exposing a limitation of tokenwise objectives. Our gradient analysis shows how low acceptance at earlier positions weakens learning signals at later positions, motivating a logarithmic objective that preserves the optimum while strengthening optimization in low-acceptance regimes. We further derive blockwise training surrogates from target-generated trajectories, enabling training from scratch without differentiating through draft sampling.
- **Consistent gains across drafters and verification rules.** Across seven math, code, and chat benchmarks, BV loss increases the mean number of tokens retained per verification call by 13.0–21.0% over CE training for DFlash and DSpark with Qwen3-4B and Qwen3-8B under block verification. These gains extend to token verification and greedy decoding without changing the inference procedure.

## 2 Related Work

**Speculative Decoding.** Speculative decoding accelerates LLM inference by using a lightweight drafter to propose candidate tokens, while verification and correction by the target model preserve the target distribution and ensure lossless generation (Leviathan et al., 2023; Chen et al., 2023). Early approaches primarily relied on autoregressive drafters that generated candidate tokens sequentially (Leviathan et al., 2023; Li et al., 2024a;b; 2025). More recent work has explored parallel drafting to reduce the sequential overhead of proposal generation. DFlash (Chen et al., 2026) employs a block diffusion model to generate draft tokens in parallel, while DSpark (Cheng et al., 2026) introduces causal correction to capture dependencies among draft tokens. Our work is complementary to these architectural advances: rather than modifying the drafter architecture, we study how diffusion drafters should be trained when their proposals are evaluated by block verification.

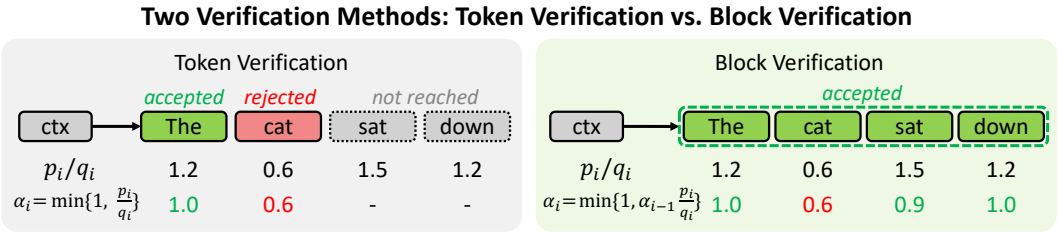


Figure 1: **Token Verification versus Block Verification.** Given the same sequence of target-to-draft probability ratios $p_i/q_i$, Token Verification accepts tokens sequentially and terminates at the first rejection; in this example, `cat` is rejected and the remaining tokens are never reached. Block Verification instead propagates the acceptance state across positions, $\alpha_i = \min\{1, \alpha_{i-1}p_i/q_i\}$, allowing a low ratio at one position to be compensated by sufficiently large ratios at subsequent positions. As a result, the entire draft block can be retained even when Token Verification would stop early.

**Verification for Speculative Decoding.** Exact verification in speculative decoding adapts rejection sampling to correct drafter proposals while ensuring that the resulting output distribution is identical to that of the target model. Figure 1 illustrates how block verification differs from the token verification. In this illustration, $p_i/q_i$ is the ratio of target and draft probabilities, while $\alpha_i$ denotes the acceptance probability. In the shown example, both methods verify the same draft: token verification could stop at the second token, where $p_i/q_i = 0.6$, whereas block verification retains the full block, yielding a longer accepted draft length. Our work focuses on the corresponding training problem: existing drafter objectives optimize tokens independently, whereas block verification determines acceptance jointly at the sequence level.

**Verification-aware Training Objectives.** Total Variation loss (TV loss) serves as a direct objective for the per-token acceptance rate, with its formulation derived from the rejection-sampling criterion used in token verification. Consequently, minimizing TV loss optimizes the expected token-level acceptance rate of the drafter under tokenwise verification. However, optimizing TV loss alone often leads to unstable or ineffective training. Building on this observation, LK losses (Samarin et al., 2026) propose a practical objective based on the negative logarithm of the token-level acceptance probability. While such objectives are well aligned with token verification, their advantage of optimizing the relevant acceptance rate does not naturally extend to block verification. This mismatch motivates a training objective derived from the sequence-level acceptance criterion used by block verification (see 2).

# 3 Method

## 3.1 Preliminaries: Tokenwise Draft Training

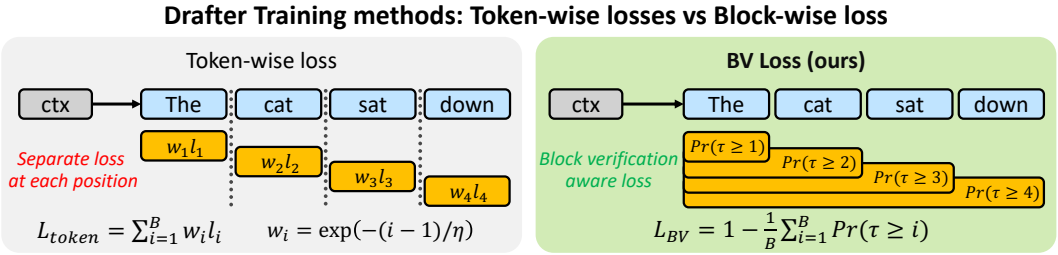


Figure 2: **Token-wise losses versus BV Loss.** Token-wise objectives optimize each draft position separately, typically through weighted local losses $w_i \ell_i$. BV Loss instead couples positions through the prefix-retention probabilities $\Pr(\tau \geq i)$ under Block Verification. Since $\sum_{i=1}^{B} \Pr(\tau \geq i)$ equals the expected number of accepted draft tokens, BV Loss directly optimizes the quantity of interest at verification time.

**Notations.** A drafter proposes $B$ tokens, which the target verifies in one call. For a target-generated training block $y_1, \ldots, y_B$, let $p_i(v)$ and $q_i(v)$ denote the target and draft probabilities of token $v$ at position $i$, evaluated on the same prefix. The target is frozen. We omit the shared context and model parameters from the notation.

**Tokenwise Optimization.** Common per-token loss functions include cross-entropy (CE), Kullback–Leibler divergence (KL), reverse KL divergence (RKL), and total variation (TV):

$$\begin{aligned} \ell_i^{\mathrm{CE}} &= -\log q_i(y_i), & \ell_i^{\mathrm{KL}} &= \textstyle\sum_v p_i(v) \log \frac{p_i(v)}{q_i(v)}, \\ \ell_i^{\mathrm{RKL}} &= \textstyle\sum_v q_i(v) \log \frac{q_i(v)}{p_i(v)}, & \ell_i^{\mathrm{TV}} &= \tfrac{1}{2} \textstyle\sum_v |p_i(v) - q_i(v)|. \end{aligned} \tag{1}$$

For a drafter with block size $B$, per-position loss $\ell_i$ is aggregated over $i = 1, \ldots, B$ as a weighted sum. A common training heuristic gives earlier positions more weight because later tokens are useful only if the earlier prefix is retained. For example, DFlash (Chen et al., 2026) applies exponential position decay to its cross-entropy loss:

$$\mathcal{L}_{\mathrm{token}} \triangleq \sum_{i=1}^{B} w_i \ell_i, \qquad w_i = \exp\left(-\frac{i-1}{\eta}\right), \qquad \eta > 0. \tag{2}$$

Smaller $\eta$ concentrates training more on earlier positions and less on later positions. We refer to training with $\mathcal{L}_{\mathrm{token}}$ as tokenwise optimization.

**Token Verification Awareness.** With tokenwise rejection sampling of token verification, a candidate token $v$ at position $i$, drawn from $q_i$ is accepted with probability $\alpha_i = \min\{1, p_i(v)/q_i(v)\}$. Averaging over the candidates at the same position gives the expected acceptance rate $\mathbb{E}_{\mathrm{TV}}[\alpha_i]$:

$$\mathbb{E}_{\mathrm{TV}}[\alpha_i] = \sum_v q_i(v) \min\left\{1, \frac{p_i(v)}{q_i(v)}\right\} = \sum_v \min\{p_i(v), q_i(v)\}. \tag{3}$$

From $\min\{p, q\} = (p + q - |p - q|)/2$, it follows that $\mathbb{E}_{\mathrm{TV}}[\alpha_i] = 1 - \ell_i^{\mathrm{TV}}$. Thus minimizing TV loss $\ell_i^{\mathrm{TV}}$ directly maximizes single-token acceptance. However, its gradients can be very small at random initialization, making pure TV optimization impractical. LK losses (Samarin et al., 2026) address this difficulty using LK loss:

$$\ell_i^{\mathrm{LK},\alpha} = -\log \mathbb{E}_{\mathrm{TV}}[\alpha_i] = -\log(1 - \ell_i^{\mathrm{TV}}). \tag{4}$$

$\ell_i^{\mathrm{LK},\alpha}$ minimizes the negative log value of the per-position expected acceptance rate $\mathbb{E}_{\mathrm{TV}}[\alpha_i]$. While TV loss and its variant LK loss have their advantage of directly optimizing token verification, their effectiveness becomes questionable when used with other verification methods.

### 3.2 Block Verification-aware Loss

In this section, we present **Block Verification-aware loss (BV loss)**, a drafter training objective that is designed to directly optimize the acceptance rate under block verification. The derivation of BV loss is introduced step-by-step in the following paragraphs.

**Block Verification Awareness.** As with token verification, we start from the expected acceptance rate at each prefix length of a drafted block $y_{1:B} \sim q$. The conditional probability of retaining the first $i$ drafted tokens is defined from the block-verification acceptance rule of Sun et al. (2025) as:

$$\alpha_i(y_{1:i}) = \min\left\{1, \alpha_{i-1}(y_{1:i-1}) \frac{p_i(y_i)}{q_i(y_i)}\right\}, \qquad \alpha_0 = 1. \tag{5}$$

Here, $\alpha_i(y_{1:i})$ is the acceptance probability of the length-$i$ drafted prefix, and $y_i$ is the $i$-th token in the drafted sequence block $y_{1:B} \sim q$. Let $\tau$ denote the number of accepted draft tokens. Averaging over drafted blocks gives the unconditional prefix acceptance rate: $\mathbb{E}_{\mathrm{BV}}[\alpha_i] \triangleq \mathbb{E}_{y_{1:B}\sim q}[\alpha_i(y_{1:i})] = \Pr_{\mathrm{BV}}(\tau \geq i)$. Then the expected acceptance length is given by:

$$\mathbb{E}_{\mathrm{BV}}[\tau] = \sum_{i=1}^{B} \Pr_{\mathrm{BV}}(\tau \geq i) = \sum_{i=1}^{B} \mathbb{E}_{\mathrm{BV}}[\alpha_i]. \tag{6}$$

**Draft Acceptance Approximation.** While directly evaluating $\mathbb{E}_{\mathrm{BV}}[\alpha_i]$ requires draft-generated prefixes $y_{1:i} \sim q$, the frozen-target training data provide prefixes $y'_{1:i} \sim p$. We therefore derive a form of Eq. (5) which permits estimating Eq. (6) on target-generated prefixes for training. Define the cumulative ratio through position $i$:

$$r_i \triangleq \prod_{j=1}^{i} \frac{q_j(y'_j)}{p_j(y'_j)}, \qquad r_0 = 1, \tag{7}$$

with the assumption that training paths are sampled from the target conditionals $p_i$ used in the ratios, $q_i$ is the proposal conditional used at inference, and $p_i$ has support wherever $q_i$ is nonzero. Multiplying $\alpha_i(y'_{1:i})$ by $r_i$ yields the acceptance score $\widetilde{\alpha}_i$:

$$\widetilde{\alpha}_i(y'_{1:i}) \triangleq r_i \alpha_i(y'_{1:i}) = \min\{r_i, \widetilde{\alpha}_{i-1}(y'_{1:i-1})\} = \min\{1, r_1, \ldots, r_i\}, \qquad \widetilde{\alpha}_0 = 1. \tag{8}$$

Assume that training paths are sampled from the target conditionals $p_i$ used in the ratios and that $q_i$ is the proposal conditional used at inference. Under these conditions, $\mathbb{E}_{y'_{1:B}\sim p}[\widetilde{\alpha}_i] = \mathbb{E}_{\mathrm{BV}}[\alpha_i]$ for every $i$, and summing over positions yields Eq. (9):

$$\mathbb{E}_{y'_{1:B}\sim p}\left[\sum_{i=1}^{B} \widetilde{\alpha}_i(y'_{1:i})\right] = \mathbb{E}_{\mathrm{BV}}[\tau]. \tag{9}$$

This identity also holds after averaging over any fixed distribution of training contexts.

**Block Verification-aware Loss.** To this end, we define Block Verification-aware loss (BV loss) as one minus the expected acceptance length normalized by the block size $B$:

$$\mathcal{L}_{\mathrm{BV}} \triangleq 1 - \frac{1}{B}\mathbb{E}_{\mathrm{BV}}[\tau]. \tag{10}$$

Using Eq. (9), it can be evaluated on target-generated training paths as

$$\mathcal{L}_{\mathrm{BV}} = 1 - \frac{1}{B}\mathbb{E}_{y'_{1:B}\sim p}\left[\sum_{i=1}^{B} \widetilde{\alpha}_i(y'_{1:i})\right]. \tag{11}$$

Minibatch training estimates the objective by averaging the per-block quantity across target-generated blocks. Minimizing the BV loss is exactly equivalent to maximizing expected BV acceptance length. For $B = 1$, BV loss reduces to TV loss $\ell^{\mathrm{TV}}$.

The cumulative ratios make BV loss a block objective. For example, for draft-to-target ratios $(1.25, 0.8)$, block verification gives $\widetilde{\alpha}_2 = \min(1, 1.25, 1) = 1$, whereas independently clipping the ratios gives $\min\{1, 1.25\}\min\{1, 0.8\} = 0.8$. A preceding likelihood surplus can therefore offset a later deficit. As a result, BV depends on how probability mass is arranged across prefixes, which the individual tokenwise overlaps do not capture.

### 3.3 Gradient Analysis and the Log BV Loss

**Gradients of the accepted-length objective.** Let $n_i$ be the earliest position attaining the minimum in $\widetilde{\alpha}_i$, including position zero. Away from ties,

$$\nabla\widetilde{\alpha}_i = \widetilde{\alpha}_i \sum_{j=1}^{n_i} \nabla \log q_j(y_j), \qquad \nabla\mathbb{E}_{\mathrm{BV}}[\tau] = \mathbb{E}_{y\sim p}\left[\sum_i \nabla\widetilde{\alpha}_i\right]. \tag{12}$$

The derivative accounts for every prefix whose active minimum depends on the prediction. Its magnitude is scaled by the prefix scores $\widetilde{\alpha}_i$ and can be small when acceptance is low. We therefore normalize by the total expected accepted length.

**Log-scaled BV objective.** We define the log BV objective as

$$\mathcal{L}_{\mathrm{BV}}^{\log} = \log B - \log \mathbb{E}_{\mathrm{BV}}[\tau] \tag{13}$$

The constant $\log B$ has zero gradient and is included only to keep the loss nonnegative. Since $-\log$ is strictly decreasing, Eq. (13) preserves the minimizers of Eq. (10) in expectation.

Its gradient is

$$\nabla\mathcal{L}_{\mathrm{BV}}^{\log} = -\frac{\nabla\mathbb{E}_{\mathrm{BV}}[\tau]}{\mathbb{E}_{\mathrm{BV}}[\tau]} = -\mathbb{E}_{y\sim p}\left[\sum_i \frac{\widetilde{\alpha}_i}{\mathbb{E}_{\mathrm{BV}}[\tau]} \sum_{j=1}^{n_i} \nabla \log q_j(y_j)\right]. \tag{14}$$

The prefix weights sum to one in expectation. Log scaling removes the overall small-acceptance factor while preserving the direction of expected-length improvement and the prefix dependencies.

**Practical training setup.** We use an initial warm-up for stable training. Log normalization controls the overall gradient scale, but prefixes with small acceptance scores can still receive a negligible share of the update at initialization. To accelerate early learning of these prefixes, we anneal the aggregation of BV scores. For positive sampled-label scores, define the annealed BV objective as the negative log of their power mean:

$$\mathcal{L}_{\mathrm{BV}}^{\mathrm{anneal}}(\beta) = -\log\left[\left(\frac{1}{B}\sum_{i=1}^{B}\widetilde{\alpha}_i^{\beta}\right)^{1/\beta}\right], \qquad 0<\beta\le 1. \tag{15}$$

This mean interpolates between the geometric mean as $\beta \to 0$ and the arithmetic mean at $\beta = 1$. Defining the loss at $\beta = 0$ by continuity gives the endpoints

$$\mathcal{L}_{\mathrm{BV}}^{\mathrm{anneal}}(0) = \frac{1}{B}\sum_{i=1}^{B}\bigl(-\log\widetilde{\alpha}_i\bigr), \qquad \mathcal{L}_{\mathrm{BV}}^{\mathrm{anneal}}(1) = -\log\left(\frac{1}{B}\sum_{i=1}^{B}\widetilde{\alpha}_i\right).$$

Thus, annealing moves from averaging per-prefix log losses to taking the log loss of the average prefix score. The latter is the blockwise log BV surrogate; averaging it over blocks differs from taking the logarithm of expected length in Eq. (13). Its gradient is a weighted average of per-prefix log-loss gradients:

$$\nabla\mathcal{L}_{\mathrm{BV}}^{\mathrm{anneal}}(\beta) = \sum_{i=1}^{B}\frac{\widetilde{\alpha}_i^{\beta}}{\sum_{j=1}^{B}\widetilde{\alpha}_j^{\beta}}\nabla(-\log\widetilde{\alpha}_i). \tag{16}$$

At $\beta = 0$, every prefix contributes with coefficient $1/B$. At $\beta = 1$, its coefficient is proportional to its acceptance score. For a score below the block mean, the initial coefficient is therefore larger by a factor of $\sum_j \widetilde{\alpha}_j/(B\widetilde{\alpha}_i)$. This strengthens the relative learning signal for low-acceptance prefixes and can speed up their early improvement before returning to the block-log objective. In the main experiments, we increase $\beta$ linearly from 0 to 1 over the first three epochs, then fix it at 1; implementation details are given in Appendix A.3.

Table 2: Average performance over seven benchmarks on Qwen3-4B and Qwen3-8B under greedy decoding ($T = 0$) and block verification ($T = 1$). Average acceptance length $\tau$ includes correction/bonus tokens. Speedup is relative to autoregressive decoding. Boldface indicates the best average in each column within each target.

| | | $T = 0$ (greedy) | | | | $T = 1$ (block verification) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | DFlash | | DSpark | | DFlash | | DSpark | |
| Target | Loss | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup |
| Qwen3-4B | CE | 4.65 | 3.71× | 5.21 | 3.97× | 4.09 | 3.21× | 4.77 | 3.47× |
| | KL | 4.80 | 3.76× | 5.49 | 4.15× | 4.17 | 3.23× | 4.97 | 3.67× |
| | RKL | 4.05 | 3.24× | 4.87 | 3.73× | 3.87 | 3.09× | 4.68 | 3.45× |
| | TV | 1.98 | 1.60× | 2.23 | 1.74× | 1.96 | 1.57× | 2.21 | 1.66× |
| | LK | 4.79 | 3.75× | 5.54 | 4.24× | 4.25 | 3.34× | 5.15 | 3.77× |
| | **BV (Ours)** | **5.22** | **4.01**× | **5.62** | **4.28**× | **4.95** | **3.84**× | **5.39** | **3.88**× |
| Qwen3-8B | CE | 4.64 | 3.60× | 5.10 | 3.82× | 4.15 | 3.29× | 4.71 | 3.48× |
| | KL | 4.81 | 3.71× | 5.58 | 4.09× | 4.19 | 3.36× | 5.07 | 3.76× |
| | RKL | 3.10 | 2.42× | 3.50 | 2.69× | 3.02 | 2.44× | 3.40 | 2.52× |
| | TV | 1.01 | 0.81× | 1.02 | 0.80× | 1.01 | 0.81× | 1.02 | 0.77× |
| | LK | 4.82 | 3.71× | 5.57 | 4.18× | 4.28 | 3.38× | 5.20 | 3.82× |
| | **BV (Ours)** | **5.27** | **4.15**× | **5.67** | **4.38**× | **4.98** | **3.88**× | **5.43** | **4.00**× |

# 4 EXPERIMENTS

## 4.1 EXPERIMENTAL SETUP

**Baselines and training setups.** We train five-layer DFlash and DSpark drafters with block size $B = 15$ proposals from scratch using frozen Qwen3-4B and Qwen3-8B target models. All methods are trained for six epochs on the 100K target-generated responses generated from OpenPerfectBlend which is an open-sourced version of perfectBlend (Xu et al., 2024). Table 2 compares six training objectives introduced in Section 3.1: CE, KL, RKL, TV, LK, and BV. All experiments reported as BV use the log BV objective in Eq. (15). For brevity, we refer to this objective simply as BV throughout the experimental section. Appendix A provides further implementation details.

**Benchmarks and evaluation.** Following the evaluation protocol of previous work (Chen et al., 2026), we evaluate across three task categories: **Math** with GSM8K (Cobbe et al., 2021), MATH-500 (Lightman et al., 2024), and AIME25 (Zhang & Math-AI, 2025); **Code** with HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), and LiveCodeBench (Jain et al., 2025); and **Chat** with MT-Bench (Zheng et al., 2023). We evaluate both greedy decoding ($T = 0$) and sampling ($T = 1$) under block verification. For each benchmark, we compute the average acceptance length $\tau$ as the total number of accepted tokens divided by the number of verification calls, including correction or bonus tokens when applicable. Speedup is the ratio of speculative to target-only autoregressive generation throughput (tokens/s), measured by an NVIDIA B200 GPU using FlashAttention-2 (Dao, 2024). Table 2 reports the average performance across the seven benchmarks.

## 4.2 MAIN RESULTS

**Consistent gains across target models and drafter methods.** BV achieves the highest average acceptance length $\tau$ in every target–drafter configuration in Table 2. Under block verification at $T = 1$, it improves over CE on all seven benchmarks for both drafters and both targets. On Qwen3-4B, DFlash improves from 4.09 to 4.95 (**+21.0%**), and DSpark from 4.77 to 5.39 (**+13.0%**). On Qwen3-8B, the corresponding gains are 4.15 to 4.98 (**+20.0%**) and 4.71 to 5.43 (**+15.3%**). These results support the motivation of Section 3.2: training for sequence-level acceptance improves the number of tokens retained by a block verifier. Appendix C.2 reports the benchmark-level results.

**Beyond token acceptance optimization.** LK already addresses weak TV gradients through log scaling, but still optimizes acceptance separately at each position. For DFlash on Qwen3-4B, BV increases mean $\tau$ under block verification from LK's 4.25 to 4.95 (**+16.5%**) with the same training

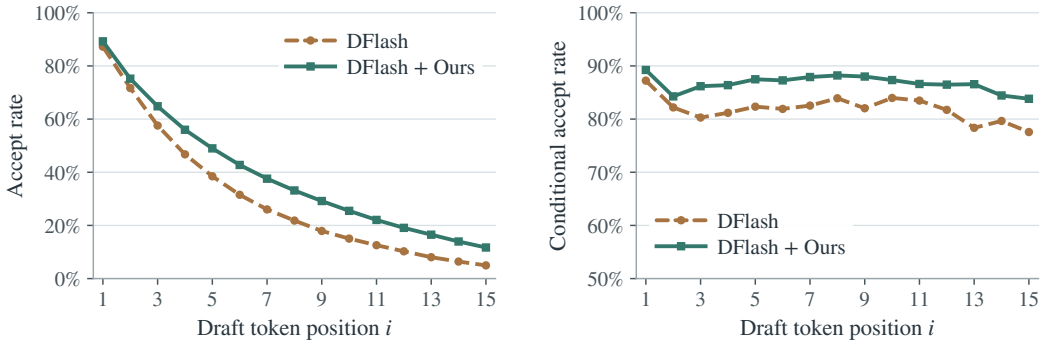


Figure 3: **Position-wise draft retention under block verification.** We compare DFlash and DFlash + Ours on 512 GSM8K prompts using Qwen3-4B. **Left:** Unconditional prefix retention, $\Pr_{\mathrm{BV}}(\tau \geq i)$, measuring the probability that the draft is retained through position $i$. **Right:** Conditional prefix retention, $\Pr_{\mathrm{BV}}(\tau \geq i \mid \tau \geq i-1)$, measuring the probability of extending an already retained prefix by one additional token.

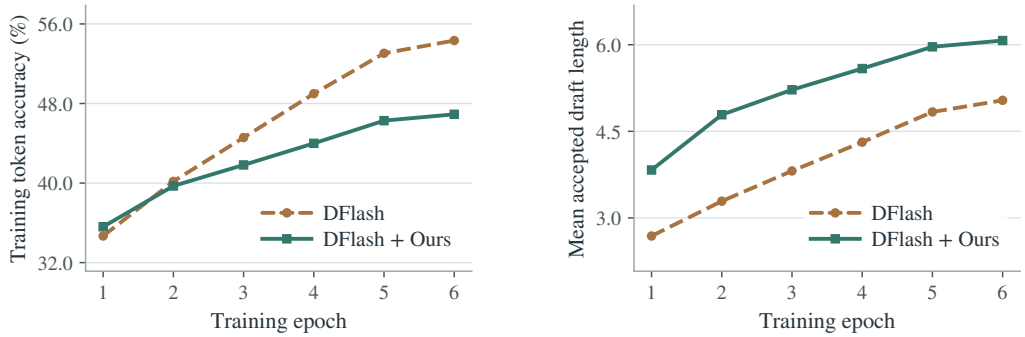


Figure 4: **Token accuracy and block acceptance exhibit different training dynamics.** We evaluate DFlash and DFlash + Ours on a fixed subset of 512 Qwen3-4B training examples. **Left:** Token accuracy measured by argmax agreement with stored target tokens. **Right:** Mean accepted draft length under block verification. While DFlash (CE) eventually achieves higher training token accuracy, BV consistently produces longer accepted drafts, highlighting the distinction between token-level prediction quality and sequence-level acceptance. Accepted length excludes anchor and correction/bonus tokens.

data and epoch budget. The same comparison on Qwen3-8B improves from 4.28 to 4.98 (**+16.4%**). DSpark also improves over LK on both targets (5.15 to 5.39 and 5.20 to 5.43). The gains over LK support extending acceptance-aware training from individual tokens to draft prefixes, beyond the benefit of log scaling alone.

**Decoding speed and greedy evaluation.** For DFlash, the block-verification gains translate into higher measured speedups: $3.21\times$ to $3.84\times$ on Qwen3-4B and $3.29\times$ to $3.88\times$ on Qwen3-8B. DSpark improves from $3.47\times$ to $3.88\times$ on Qwen3-4B and from $3.48\times$ to $4.00\times$ on Qwen3-8B. BV also improves $\tau$ over CE on every benchmark under greedy decoding, with mean gains of 12.3% and 7.9% for DFlash and DSpark on Qwen3-4B, and 13.6% and 11.2%, respectively, on Qwen3-8B.

### 4.3 Analysis and Ablation

**Acceptance across positions.** Figure 3 compares the final DFlash CE and BV checkpoints on 512 GSM8K prompts with Qwen3-4B and block verification. The left panel measures unconditional prefix retention, $\Pr_{\mathrm{BV}}(\tau \geq i) = \mathbb{E}_{\mathrm{BV}}[\alpha_i]$. First-position retention is similar (87.2% for CE and 89.2% for BV), while retention at position 8 increases from 21.8% to 33.2%. By Eq. (6), the sum of these probabilities equals the mean accepted draft length, which rises from 4.56 to 5.86 (**+28.4%**). The right panel measures the probability of extending an already retained prefix. BV improves this

Table 3: Generalization to token verification at $T = 1$ with the same checkpoints as Table 2. Entries are seven-benchmark mean $\tau$, using the same counting convention. Bold marks the best mean in each row, including ties.

| Target | Drafter | $\tau$ under token verification | | | | | |
|---|---|---|---|---|---|---|---|
| | | CE | KL | RKL | TV | LK | **BV (Ours)** |
| Qwen3-4B | DFlash | 4.01 | 4.11 | 3.86 | 1.96 | 4.17 | **4.85** |
| | DSpark | 4.64 | 4.84 | 4.64 | 2.20 | 4.98 | **5.20** |
| Qwen3-8B | DFlash | 4.04 | 4.13 | 3.00 | 1.01 | 4.19 | **4.86** |
| | DSpark | 4.63 | 4.90 | 3.38 | 1.02 | 5.05 | **5.26** |

rate at all 15 positions, including an increase from 77.6% to 83.8% at the final position. Complete-block retention more than doubles, from 4.97% to 11.69%. The improvement therefore extends beyond the first token to the longer prefixes credited by BV loss.

**Training dynamics.** Figure 4 compares token accuracy and accepted draft length on a fixed subset of 512 training examples, using eight shared blocks per example across all six epochs. Here, accuracy measures argmax agreement with sampled target tokens. At epoch six, CE matches 54.33% of the stored target tokens, compared with 46.91% for BV, while BV retains 20.6% more draft tokens (6.07 versus 5.04). The reversed ranking shows that token accuracy is an incomplete proxy for block acceptance, motivating optimization with the block verification-aware training.

**Gains extend to token verification.** Table 3 evaluates the same checkpoints with token verification at $T = 1$. BV achieves the highest mean $\tau$ in all target–drafter configurations, improving over CE by 20.9% and 12.1% for DFlash and DSpark on Qwen3-4B, and by 20.3% and 13.6% on Qwen3-8B. Although BV is derived from block verification, these results show that the learned improvements also benefit token verification without retraining.

## 5 Conclusion

We introduced Block Verification-aware loss (BV loss), a sequence-level drafter training objective derived from the block verification acceptance rule. The objective directly targets expected accepted draft length and recovers TV loss in the single-token case. A reformulation on target-generated trajectories enables training without differentiating through draft sampling or verification, while a blockwise logarithmic surrogate and early-training annealing support training from scratch.

Across math, code, and chat benchmarks, BV loss improves the mean number of tokens retained per verification call by 13.0–21.0% over CE training for DFlash and DSpark with Qwen3-4B and Qwen3-8B under block verification, with corresponding gains in decoding throughput. It also outperforms tokenwise acceptance objectives such as TV and LK, and its gains extend to token verification and greedy decoding without changing the inference procedure. Our analyses show improved retention beyond the first token and demonstrate that training token accuracy alone is an incomplete proxy for accepted draft length. Together, these findings highlight the benefit of training parallel drafters for sequence-level acceptance rather than optimizing individual token predictions in isolation.

## 6 Limitations

Our study focuses on parallel block drafters, while the broader principle of verification-aligned training may apply beyond this setting. In particular, BV loss may also be applicable to autoregressive drafters such as EAGLE (Li et al., 2024a;b; 2025), although our empirical evaluation is limited to diffusion-based drafters. More broadly, this perspective could be adapted to tree-based proposal and verification schemes, including methods based on Traversal Verification (Weng et al., 2025). Exploring these extensions is beyond the scope of this work and is left for future research.

### AI USE STATEMENT

In this work, we used generative AI tools to assist with writing experimental code. The authors formulated the research ideas and hypotheses, designed the experiments, and interpreted the results without generative AI assistance. Additionally, we used generative AI tools to draft and revise the manuscript and to prepare its LaTeX source. The authors verified the evaluation code and reviewed the final manuscript, including the AI-assisted text. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

### REPRODUCIBILITY STATEMENT

Section 3 defines the BV objective and its training formulation, with the assumptions and supporting derivations detailed in Appendix B. Section 4.1 describes the models, benchmarks, and evaluation metrics. Appendix A documents the compared objectives, training data and preprocessing, optimization settings, annealing schedule, and numerical implementation. The diagnostic evaluation and speedup measurement protocols are detailed in Appendices A.5 and A.6, respectively, and Appendix C.2 provides the full per-benchmark results for the compared objectives.

## REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling, 2023. URL https://arxiv.org/abs/2302.01318.

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=Oz335dV48X.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Xin Cheng, Xingkai Yu, Chenze Shao, Jiashi Li, Yunfan Xiong, Yi Qian, Jiaqi Zhu, Shirong Ma, Xiaokang Zhang, Jiasheng Ye, Qinyu Chen, Chengqi Deng, Jiping Yu, Damai Dai, Zhengyan Zhang, Yixuan Wei, Yixuan Tan, Wenkai Yang, Runxin Xu, Yu Wu, Zhean Xu, Xuanyu Wang, Muyang Chen, Rui Tian, Xiao Bi, Zhewen Hao, Shaoyuan Chen, Huanqi Cao, Wentao Zhang, Anyi Xu, Huishuai Zhang, Dongyan Zhao, and Wenfeng Liang. DSpark: Confidence-scheduled speculative decoding with semi-autoregressive generation, 2026. URL https://arxiv.org/abs/2607.05147.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=mZn2Xyh9Ec.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

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

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 28935–28948. PMLR, 21–27 Jul 2024a. URL https://proceedings.mlr.press/v235/li24bt.html.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 7421–7432, 2024b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/c7b5a35ea98b62512a869c19ea7b03cb-Abstract-Conference.html.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Alexander Samarin, Sergei Krutikov, Anton Shevtsov, Sergei Skvortsov, Filipp Fisin, and Alexander Golubev. LK losses: Direct acceptance rate optimization for speculative decoding. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=ZqnVidXtxV.

Ziteng Sun, Uri Mendlovic, Yaniv Leviathan, Asaf Aharoni, Jae Hun Ro, Ahmad Beirami, and Ananda Theertha Suresh. Block verification accelerates speculative decoding. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=frsg32u0rO.

Yepeng Weng, Qiao Hu, Xujie Chen, Li Liu, Dianwen Mei, Huishi Qiu, Jiang Tian, and zhongchao shi. Traversal verification for speculative tree decoding. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=8nOMhDFpkU.

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

Yifan Zhang and Team Math-AI. American invitational mathematics examination (AIME) 2025. Hugging Face dataset, 2025. URL https://huggingface.co/datasets/math-ai/aime25.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

# A Additional Implementation Details

## A.1 Training Objectives

We compare CE, KL, RKL, TV, and LK as tokenwise alternatives to BV. The tokenwise losses follow Eqs. (1) and (4) and use the position weights in Eq. (2), with $\eta = 7$ for DFlash and $\eta = 4$ for DSpark. In training, the weighted losses are normalized by the total weight over valid prediction positions. The target model remains frozen for every objective.

**CE, KL, and RKL.** CE predicts the stored target-generated token at each supervised position. KL instead matches the full target distribution by minimizing $D_{\mathrm{KL}}(p_i \| q_i)$; with a frozen target, this is equivalent to soft-target cross-entropy up to a constant. RKL reverses the direction to $D_{\mathrm{KL}}(q_i \| p_i)$, weighting the discrepancy by the draft distribution. Both divergences use target and draft probabilities aligned to the same prediction position and are evaluated over the full vocabulary.

**TV and LK.** TV minimizes the total variation distance between target and draft distributions, equivalently maximizing their tokenwise overlap $\sum_v \min\{p_i(v), q_i(v)\}$. LK applies a negative logarithm to this overlap, increasing the gradient scale when acceptance is low. These objectives also use full-vocabulary distributions.

**BV loss.** BV loss aggregates prefix-retention scores into an estimate of accepted draft length, coupling predictions across positions. The main BV runs use the vocabulary-integrated estimate in Appendix B.4 within a blockwise log objective. The initial aggregation schedule is described in Appendix A.3.

## A.2 Training and Evaluation Protocol

**Training data and optimization.** We build the training corpora from the same 100K conversations selected from Open-PerfectBlend, an open-source reproduction of the instruction dataset described in PerfectBlend (Xu et al., 2024). For each target (Qwen3-4B or Qwen3-8B), we regenerate the assistant responses at $T = 1$. The main-table runs for a given target share this corpus and train from scratch for six epochs. They use AdamW with peak learning rate $6 \times 10^{-4}$, global batch size 32, local batch size four, 4% learning-rate warmup followed by cosine decay, and seed 42. The maximum training sequence length is 3,072 tokens.

**Drafter configurations.** All compared models use five-layer drafters with $B = 15$ draft tokens, where the anchor token is not counted toward $B$. DSpark retains its causal Markov correction, and the losses are applied to the resulting corrected proposal distribution. The correction parameters are trained with the drafter, while the confidence head is disabled across the compared DSpark objectives.

**Decoding and reporting.** All objectives are evaluated with a 256-token generation limit and thinking disabled, under greedy decoding or full-vocabulary sampling at $T = 1$. For generation call $j$, let $R_j$ be the number of retained tokens, including correction/bonus tokens when present, and let $V_j$ be its number of verification calls. For benchmark $d$, the table metric is

$$\tau_d^{\mathrm{table}} = \frac{\sum_{j \in d} R_j}{\sum_{j \in d} V_j}, \qquad \mathrm{Avg.} = \frac{1}{7} \sum_{d=1}^{7} \tau_d^{\mathrm{table}}.$$

We pool verification calls within each benchmark and give equal weight to the seven benchmarks in the final average. Tables abbreviate $\tau_d^{\mathrm{table}}$ as $\tau$. Counts exclude the initial prefill token and are recorded before the final generation is cropped to the output budget. The evaluation includes 512 GSM8K, 500 MATH-500, 30 AIME25, 164 HumanEval, 257 MBPP, 256 LiveCodeBench, and 80 two-turn MT-Bench examples, giving 1,879 generation calls per decoding condition. Appendix A.6 describes the throughput measurement.

### A.3 Initial Annealing Schedule

The main BV runs linearly increase $\beta$ from zero to one over the first three epochs, then use the blockwise log loss for the remaining three epochs. This schedule applies Eqs. (15) and (16) to the vocabulary-integrated scores $\bar{\alpha}_i$ in place of $\widetilde{\alpha}_i$ for both drafters. It gradually shifts from equal weighting of prefix log-scores to weighting proportional to their acceptance scores. This aggregation schedule is separate from the learning-rate warmup; unannealed BV runs use $\beta = 1$ throughout.

During the ramp, the implementation differentiates a negative weighted mean of log-scores while holding the normalized power weights fixed in the backward pass. For positive scores with inactive numerical floors, this yields the gradient in Eq. (16), although the logged scalar need not equal Eq. (15). DFlash floors scores at $10^{-6}$ during the ramp; DSpark computes the annealed log-scores without this floor.

### A.4 Numerical Stability

BV cumulative ratios and prefix minima are computed in FP32 log space, with gradients propagated through the prefix states. In the block-log phase, log summands are floored at $-80$ and the summed length estimate at $10^{-6}$. The theoretical identities use the unclamped expressions.

### A.5 Diagnostic Evaluations

The analyses in Figures 3 and 4 compare DFlash (CE) and DFlash + Ours (BV) to examine how the training objective affects retained prefixes and token predictions.

**Prefix-retention analysis (Figure 3).** The final six-epoch DFlash CE and annealed BV checkpoints are evaluated on 512 GSM8K prompts with Qwen3-4B, block verification, $T = 1$, and $B = 15$. Accepted-draft counts exclude correction/bonus tokens and prefill, respect EOS, and precede final output-budget cropping. We pool verification calls across prompts to estimate unconditional prefix survival, $\Pr_{\mathrm{BV}}(\tau \geq i)$, in the left panel. The right panel reports the ratio to $\Pr_{\mathrm{BV}}(\tau \geq i-1)$, with denominator one at $i = 1$, giving the probability of extending a retained prefix. These statistics average over each checkpoint's decoding trajectories; unlike $\alpha_i(y_{1:i})$ in Eq. (5), they do not fix the proposed token values.

**Training dynamics (Figure 4).** Both models are evaluated at 4,096 fixed contexts across all six epochs. We construct these contexts by uniformly sampling 512 of the 98,499 retained Qwen3-4B training examples (seed 0) and choosing eight full assistant blocks per example. Cached labels and the 3,072-token length limit are preserved. Blocks may overlap, but each prefix is evaluated independently and the same blocks are used for both models. Token accuracy pools 61,440 unweighted argmax predictions against the stored target tokens. Accepted draft length averages one block-verification draw per context using full-vocabulary target and draft distributions ($T = 1$, $B = 15$), respecting EOS and excluding anchors and correction/bonus tokens. The CE curve uses the original epoch checkpoints; the BV curve uses a same-seed repeat with checkpoints retained at every epoch. The repeat preserves global batch size 32 and the optimizer schedule across a change in GPU count. Both curves use a single training seed.

### A.6 Speedup Measurement

**Execution and timing.** Each inference run uses a single NVIDIA B200 GPU at batch size one, with BF16 model weights and FlashAttention-2. Both speculative decoding (SD) and target-only autoregressive decoding (AR) run in inference mode with fresh KV caches, matching prompts, sampling and stopping rules, and a 256-token generation limit. We time the same decoding implementation used to measure $\tau$, with CUDA synchronization at the boundaries of each timed region. Elapsed time includes prefill, draft generation, verification, sampling, and cache updates; model loading, prompt formatting, tokenization, and file I/O are excluded. After a warmup for each dataset, we make one measured pass over its evaluation examples.

**Aggregation.** For generation call $i$ in benchmark $d$, let $n_i^m$ and $t_i^m$ denote the generated token count and total elapsed time for $m \in \{\mathrm{SD}, \mathrm{AR}\}$. We pool token counts and times within each

benchmark before taking the throughput ratio:

$$\mathrm{Speedup}_d = \frac{\sum_{i\in d} n_i^{\mathrm{SD}} / \sum_{i\in d} t_i^{\mathrm{SD}}}{\sum_{i\in d} n_i^{\mathrm{AR}} / \sum_{i\in d} t_i^{\mathrm{AR}}}.$$

# B Additional Derivations

We first establish the target-path identity underlying BV and illustrate why tokenwise overlap does not determine block acceptance. The remaining derivations explain prefix credit, vocabulary integration, and the relationship between the population objective and its training surrogate.

## B.1 Expected Acceptance Length on Target-Generated Paths

We follow Section 3.2: $y$ denotes a proposed draft and $y'$ a target-generated training path. Fix a verification context $c$, a frozen target, and positive target and proposal probabilities. Let $q_i$ be the conditionals of the actual proposal process and $p_i$ the target conditionals used to generate the training paths. For any prefix $x_{1:i}$, write $p(x_{1:i} \mid c) = \prod_{j=1}^{i} p_j(x_j)$ and $q(x_{1:i} \mid c) = \prod_{j=1}^{i} q_j(x_j)$.

The block-verification rule in Eq. (5) gives the retention probability conditional on the proposed prefix, averaging over proposal suffixes and verifier randomness (Sun et al., 2025). Here, conditioning fixes the proposed token values rather than an earlier acceptance event. Multiplying by the cumulative ratio in Eq. (7) gives $r_i\alpha_i = \min\{r_i, r_{i-1}\alpha_{i-1}\} = \widetilde{\alpha}_i$. Consequently, the joint mass of proposing and retaining a prefix is

$$\begin{aligned}\Pr_{\mathrm{BV}}(y_{1:i} = x_{1:i}, \tau \geq i \mid c) &= q(x_{1:i} \mid c)\alpha_i(x_{1:i}) \\ &= p(x_{1:i} \mid c)\widetilde{\alpha}_i(x_{1:i}).\end{aligned} \tag{17}$$

Summing over prefixes yields $\Pr_{\mathrm{BV}}(\tau \geq i \mid c) = \mathbb{E}_{y'\sim p}[\widetilde{\alpha}_i \mid c]$. Applying the tail-sum identity establishes Eq. (9), and normalizing by $B$ gives the training form in Eq. (11). The same result holds after averaging over a fixed distribution of training contexts. Under these sampling conditions, a minibatch average of $1 - B^{-1}\sum_i \widetilde{\alpha}_i$ is an unbiased estimate of the population BV loss.

For $B = 1$, the expected score is $\sum_v p_1(v)\min\{1, q_1(v)/p_1(v)\} = \sum_v \min\{p_1(v), q_1(v)\}$. Thus the linear BV loss recovers TV exactly, while the fully integrated block-log loss recovers LK.

## B.2 Equal Tokenwise Overlap, Different Block Acceptance

Consider a two-token block with independent binary target distributions, $p_1(a) = 0.2$ and $p_2(a) = 0.1$. The two proposals below have identical tokenwise overlaps; in each distribution, the other token receives the remaining probability. Enumerating the four possible blocks using Eq. (17) gives

| Quantity | Draft A | Draft B |
|---|---|---|
| $(q_1(a), q_2(a))$ | $(0.3, 0.2)$ | $(0.1, 0.2)$ |
| Tokenwise overlaps | $(0.9, 0.9)$ | $(0.9, 0.9)$ |
| BV prefix survival | $(0.9, 0.83)$ | $(0.9, 0.89)$ |
| BV $\mathbb{E}[\tau]$ | 1.73 | 1.79 |

Both proposals have TV loss $0.1$ and LK loss $-\log(0.9)$ at each position, yet their expected block acceptance lengths differ. Under token verification, both have survival probabilities $(0.9, 0.81)$ and expected accepted draft length $1.71$. Tokenwise overlap therefore determines token-verification acceptance in this example, but not block acceptance. The latter depends on how likelihood ratios combine across prefixes, which a fixed weighted sum of these tokenwise losses cannot distinguish.

### B.3 Prefix Credit and Conditional-Logit Gradients

For a fixed target-generated block, let $S = \sum_i \widetilde{\alpha}_i > 0$ and let $n_i$ be the active prefix minimum from Section 3.3. Away from ties, Eq. (12) gives

$$\nabla(\log B - \log S) = -\sum_{j=1}^{B} c_j \nabla \log q_j(y'_j), \qquad c_j = \frac{\sum_{i:n_i \geq j} \widetilde{\alpha}_i}{S} \in [0,1]. \tag{18}$$

The coefficient $c_j$ collects the scores whose active prefix minimum depends on position $j$. For conditional softmax logits $z_{j,v}$, holding the path and other conditional logits fixed,

$$\frac{\partial(\log B - \log S)}{\partial z_{j,v}} = -c_j\big(\mathbf{1}\{v = y'_j\} - q_j(v)\big), \qquad \left|\frac{\partial(\log B - \log S)}{\partial z_{j,v}}\right| \leq 1. \tag{19}$$

For this sampled-label surrogate, log scaling removes the overall small-score factor while keeping the conditional-logit derivatives bounded. At ties, convex combinations of the active gradients satisfy the same bound. This statement concerns conditional logits; it does not bound gradients with respect to shared neural-network parameters.

### B.4 Vocabulary-Integrated Length Estimation

The sampled-label estimate is $\widehat{\tau}_{\text{sample}} = \sum_i \widetilde{\alpha}_i(y'_{1:i})$. To use the full target distribution at the current position, we can integrate out that token while keeping the sampled prefix fixed:

$$\begin{aligned} \bar{\alpha}_i &:= \mathbb{E}_{y'_i \sim p_i}[\widetilde{\alpha}_i \mid y'_{<i}, c] \\ &= \sum_v \min\{\widetilde{\alpha}_{i-1} p_i(v), r_{i-1} q_i(v)\}, \qquad \widehat{\tau}_{\text{int}} = \sum_i \bar{\alpha}_i. \end{aligned} \tag{20}$$

The formula follows by substituting $r_i = r_{i-1} q_i(y'_i)/p_i(y'_i)$ into Eq. (8). By total expectation, both length estimates are unbiased for $\mathbb{E}_{\text{BV}}[\tau \mid c]$ under the conditions of Appendix B.1. Only the current score is integrated: $r_i$ and $\widetilde{\alpha}_i$ continue to evolve along the sampled target path, preserving its prefix dependencies. The main BV runs use $\widehat{\tau}_{\text{int}}$ in both the blockwise log objective and its annealed form.

### B.5 Population and Blockwise Log Objectives

The population objective in Eq. (13) takes the negative logarithm after averaging accepted length. Practical training instead averages the normalized negative log of each block estimate, $\ell^{\log}_{\text{BV}} = \log B - \log \widehat{\tau}$. For a positive, unbiased length estimate without numerical floors, Jensen's inequality gives

$$\mathbb{E}[\ell^{\log}_{\text{BV}}] \geq \log B - \log \mathbb{E}[\widehat{\tau}] = \mathcal{L}^{\log}_{\text{BV}}. \tag{21}$$

Thus, the expected blockwise log loss upper-bounds the population log objective. An unbiased length estimate need not yield an unbiased log loss or the same optimum within a restricted drafter family. Both objectives share the ideal solution $q = p$, when it is representable: every prefix score is one and $\widehat{\tau} = B$. The annealing schedule returns to the blockwise log surrogate at $\beta = 1$.

## C Additional Experimental Results

### C.1 Comparison of BV Training Recipes

Table 4 compares CE, BV without annealing, and the main annealed BV recipe. Importantly, BV already provides substantial acceptance gains without annealing, showing that the annealing schedule is not essential for successful optimization. Annealing provides an additional improvement, yielding the best results across the reported settings.

On Qwen3-4B at $T = 1$, unannealed BV increases DFlash's mean $\tau$ from 4.09 to 4.74 (**+15.9%**) and DSpark's from 4.77 to 5.17 (**+8.4%**), both under block verification. Annealing further improves these values to 4.95 and 5.39, respectively. Qwen3-8B shows the same pattern: unannealed BV reaches 4.80 for DFlash and 5.13 for DSpark, while the annealed recipes reach 4.98 and 5.43. A similar trend is observed under greedy decoding. Overall, these results suggest that annealing is a useful optimization aid rather than a prerequisite for BV training, consistent with the discussion in Section 3.3 on increasing the relative training weight of low-scoring prefixes early in training.

Table 4: BV and annealing ablation. Entries are average $\tau$ over the seven benchmarks, with the counting convention of Table 2. At $T = 1$, both drafters use block verification.

| | Qwen3-4B | | | | Qwen3-8B | | | |
|---|---|---|---|---|---|---|---|---|
| | DFlash | | DSpark | | DFlash | | DSpark | |
| Training loss | $T = 0$ | $T = 1$ | $T = 0$ | $T = 1$ | $T = 0$ | $T = 1$ | $T = 0$ | $T = 1$ |
| Baseline (CE) | 4.65 | 4.09 | 5.21 | 4.77 | 4.64 | 4.15 | 5.10 | 4.71 |
| BV w/o annealing | 4.98 | 4.74 | 5.37 | 5.17 | 5.07 | 4.80 | 5.36 | 5.13 |
| BV with annealing (Ours) | **5.22** | **4.95** | **5.62** | **5.39** | **5.27** | **4.98** | **5.67** | **5.43** |

### C.2 Full Benchmark Results

Tables 5 and 6 provide the per-benchmark results for every training objective with Qwen3-4B and Qwen3-8B. They expand the averages in Tables 2 and 3 across greedy decoding, token verification at $T = 1$, and block verification at $T = 1$, using the same $\tau$ convention and autoregressive speedup reference as Section 4.1. BV improves over CE in all 84 benchmark-level acceptance-length comparisons across the two targets, two drafters, and three settings. BV also achieves the highest mean acceptance length among the compared objectives in each configuration. The gains over CE therefore extend across individual math, code, and chat benchmarks, as well as their aggregate results.

Table 5: Full benchmark results for DFlash and DSpark on Qwen3-4B. Here, $\tau$ is the reported verification-call length, including correction/bonus when present; the Method uses draft-only $\tau$. Avg. retains the reported seven-benchmark mean. Speedup is relative to autoregressive decoding. Bold marks the best value for each metric within each drafter and setting..

| Drafter | Loss | Math | | | | | | Code | | | | | | Chat | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | GSM8K | | MATH-500 | | AIME25 | | HumanEval | | MBPP | | LCB | | MT-Bench | | Avg. | |
| $T=0$ | | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ |
| DFlash | CE | 4.99× | 6.32 | 4.46× | 5.64 | 3.63× | 4.50 | 3.53× | 4.45 | 3.41× | 4.27 | 3.59× | 4.44 | 2.34× | 2.94 | 3.71× | 4.65 |
| | KL | 5.05× | 6.51 | 4.54× | 5.83 | 3.68× | 4.67 | 3.60× | 4.61 | 3.46× | 4.40 | 3.61× | 4.56 | 2.40× | 3.03 | 3.76× | 4.80 |
| | RKL | 4.52× | 5.69 | 3.84× | 4.81 | 3.09× | 3.85 | 3.05× | 3.82 | 2.99× | 3.71 | 3.22× | 3.98 | 2.00× | 2.51 | 3.24× | 4.05 |
| | TV | 1.95× | 2.42 | 2.01× | 2.48 | 1.78× | 2.19 | 1.39× | 1.71 | 1.44× | 1.77 | 1.34× | 1.63 | 1.32× | 1.64 | 1.60× | 1.98 |
| | LK | 5.08× | 6.55 | 4.52× | 5.81 | 3.64× | 4.63 | 3.59× | 4.61 | 3.45× | 4.39 | 3.61× | 4.55 | 2.38× | 2.99 | 3.75× | 4.79 |
| | **BV (Ours)** | **5.57**× | **7.34** | **4.84**× | **6.36** | **3.94**× | **5.15** | **3.79**× | **4.93** | **3.61**× | **4.70** | **3.76**× | **4.87** | **2.54**× | **3.20** | **4.01**× | **5.22** |
| DSpark | CE | 5.28× | 7.03 | 4.84× | 6.38 | 3.87× | 5.08 | 3.84× | 5.07 | 3.66× | 4.79 | 3.78× | 4.92 | 2.50× | 3.23 | 3.97× | 5.21 |
| | KL | 5.55× | 7.42 | 5.07× | 6.77 | 4.08× | 5.41 | 4.00× | 5.30 | 3.81× | 5.03 | 3.88× | 5.11 | 2.63× | 3.39 | 4.15× | 5.49 |
| | RKL | 5.19× | 6.84 | 4.50× | 5.90 | 3.66× | 4.79 | 3.51× | 4.61 | 3.41× | 4.44 | 3.53× | 4.56 | 2.27× | 2.93 | 3.73× | 4.87 |
| | TV | 1.95× | 2.52 | 2.16× | 2.78 | 1.93× | 2.48 | 1.58× | 2.03 | 1.58× | 2.04 | 1.53× | 1.95 | 1.43× | 1.82 | 1.74× | 2.23 |
| | LK | **5.71**× | 7.53 | 5.17× | 6.79 | 4.21× | 5.49 | **4.08**× | **5.35** | **3.90**× | 5.04 | 3.99× | 5.14 | 2.63× | **3.41** | 4.24× | 5.54 |
| | **BV (Ours)** | **5.71**× | **7.75** | **5.25**× | **6.91** | **4.33**× | **5.64** | **4.08**× | 5.33 | **3.90**× | **5.07** | **4.03**× | **5.20** | **2.64**× | **3.41** | **4.28**× | **5.62** |
| $T=1$, token verify | | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ |
| DFlash | CE | 4.21× | 5.40 | 3.70× | 4.74 | 2.91× | 3.70 | 3.09× | 3.97 | 2.93× | 3.71 | 3.15× | 3.95 | 2.06× | 2.59 | 3.15× | 4.01 |
| | KL | 4.29× | 5.53 | 3.79× | 4.80 | 2.97× | 3.75 | 3.27× | 4.19 | 3.04× | 3.84 | 3.23× | 4.02 | 2.08× | 2.62 | 3.24× | 4.11 |
| | RKL | 4.15× | 5.31 | 3.53× | 4.47 | 2.88× | 3.60 | 2.96× | 3.73 | 2.89× | 3.62 | 3.11× | 3.85 | 1.93× | 2.43 | 3.06× | 3.86 |
| | TV | 1.86× | 2.37 | 1.92× | 2.44 | 1.71× | 2.17 | 1.35× | 1.72 | 1.39× | 1.76 | 1.30× | 1.62 | 1.31× | 1.63 | 1.55× | 1.96 |
| | LK | 4.41× | 5.61 | 3.82× | 4.85 | 3.05× | 3.84 | 3.33× | 4.24 | 3.07× | 3.85 | 3.33× | 4.15 | 2.14× | 2.69 | 3.31× | 4.17 |
| | **BV (Ours)** | **5.14**× | **6.70** | **4.44**× | **5.74** | **3.59**× | **4.61** | **3.66**× | **4.69** | **3.47**× | **4.49** | **3.63**× | **4.67** | **2.45**× | **3.09** | **3.77**× | **4.85** |
| DSpark | CE | 4.49× | 6.25 | 4.06× | 5.63 | 3.27× | 4.49 | 3.30× | 4.57 | 3.15× | 4.31 | 3.20× | 4.31 | 2.18× | 2.94 | 3.38× | 4.64 |
| | KL | 4.71× | 6.52 | 4.29× | 5.91 | 3.45× | 4.71 | 3.49× | 4.79 | 3.27× | 4.45 | 3.36× | 4.53 | 2.21× | 3.00 | 3.54× | 4.84 |
| | RKL | 4.62× | 6.38 | 3.97× | 5.48 | 3.24× | 4.48 | 3.26× | 4.50 | 3.18× | 4.33 | 3.32× | 4.48 | 2.10× | 2.84 | 3.38× | 4.64 |
| | TV | 1.81× | 2.50 | 1.99× | 2.73 | 1.74× | 2.39 | 1.48× | 2.03 | 1.49× | 2.02 | 1.45× | 1.94 | 1.34× | 1.80 | 1.62× | 2.20 |
| | LK | 4.97× | 6.74 | 4.38× | 5.99 | **3.57**× | **4.85** | 3.59× | 4.88 | 3.41× | 4.58 | 3.55× | 4.70 | 2.30× | 3.11 | 3.68× | 4.98 |
| | **BV (Ours)** | **5.14**× | **7.15** | **4.60**× | **6.33** | 3.53× | **4.85** | **3.69**× | **5.08** | **3.54**× | **4.85** | **3.57**× | **4.88** | **2.38**× | **3.23** | **3.78**× | **5.20** |
| $T=1$, block verify | | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ |
| DFlash | CE | 4.32× | 5.55 | 3.78× | 4.82 | 3.05× | 3.88 | 3.14× | 4.01 | 2.97× | 3.76 | 3.15× | 3.96 | 2.09× | 2.65 | 3.21× | 4.09 |
| | KL | 4.32× | 5.63 | 3.79× | 4.91 | 3.05× | 3.92 | 3.22× | 4.20 | 3.01× | 3.87 | 3.15× | 4.00 | 2.10× | 2.67 | 3.23× | 4.17 |
| | RKL | 4.22× | 5.35 | 3.57× | 4.50 | 2.82× | 3.49 | 2.99× | 3.77 | 2.92× | 3.64 | 3.16× | 3.91 | 1.94× | 2.46 | 3.09× | 3.87 |
| | TV | 1.91× | 2.39 | 1.95× | 2.43 | 1.74× | 2.16 | 1.37× | 1.71 | 1.42× | 1.76 | 1.33× | 1.62 | 1.30× | 1.62 | 1.57× | 1.96 |
| | LK | 4.48× | 5.74 | 3.94× | 5.04 | 3.08× | 3.90 | 3.29× | 4.24 | 3.12× | 3.93 | 3.31× | 4.14 | 2.16× | 2.73 | 3.34× | 4.25 |
| | **BV (Ours)** | **5.27**× | **6.85** | **4.56**× | **5.92** | **3.78**× | **4.80** | **3.70**× | **4.79** | **3.52**× | **4.53** | **3.61**× | **4.68** | **2.42**× | **3.09** | **3.84**× | **4.95** |
| DSpark | CE | 4.66× | 6.47 | 4.20× | 5.80 | 3.39× | 4.67 | 3.37× | 4.65 | 3.21× | 4.38 | 3.24× | 4.40 | 2.20× | 3.01 | 3.47× | 4.77 |
| | KL | 4.99× | 6.80 | 4.49× | 6.08 | 3.50× | 4.77 | 3.62× | 4.91 | 3.40× | 4.56 | 3.45× | 4.57 | 2.26× | 3.10 | 3.67× | 4.97 |
| | RKL | 4.76× | 6.51 | 4.08× | 5.57 | 3.23× | 4.37 | 3.31× | 4.53 | 3.27× | 4.41 | 3.37× | 4.51 | 2.13× | 2.90 | 3.45× | 4.68 |
| | TV | 1.88× | 2.50 | 2.05× | 2.74 | 1.82× | 2.42 | 1.54× | 2.05 | 1.52× | 2.02 | 1.48× | 1.95 | 1.34× | 1.80 | 1.66× | 2.21 |
| | LK | 5.10× | 7.00 | 4.50× | 6.19 | 3.78× | 5.22 | 3.67× | 5.02 | 3.50× | 4.73 | 3.53× | 4.71 | 2.31× | 3.16 | 3.77× | 5.15 |
| | **BV (Ours)** | **5.22**× | **7.28** | **4.74**× | **6.60** | **3.83**× | **5.39** | **3.73**× | **5.26** | **3.55**× | **4.91** | **3.66**× | **5.05** | **2.39**× | **3.26** | **3.88**× | **5.39** |

Table 6: Full benchmark results for DFlash and DSpark on Qwen3-8B. Here, $\tau$ is the reported verification-call length, including correction/bonus when present; the Method uses draft-only $\tau$. Avg. retains the reported seven-benchmark mean. Speedup is relative to autoregressive decoding. Bold marks the best value for each metric within each drafter and setting.

| Drafter | Loss | Math | | | | | | Code | | | | | | Chat | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | GSM8K | | MATH-500 | | AIME25 | | HumanEval | | MBPP | | LCB | | MT-Bench | | Avg. | |
| $T = 0$ | | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ |
| DFlash | CE | 4.70× | 6.11 | 4.28× | 5.57 | 3.53× | 4.55 | 3.54× | 4.61 | 3.26× | 4.20 | 3.58× | 4.58 | 2.32× | 2.88 | 3.60× | 4.64 |
| | KL | 4.86× | 6.37 | 4.42× | 5.79 | 3.59× | 4.67 | 3.69× | 4.80 | 3.37× | 4.34 | 3.67× | 4.72 | 2.39× | 2.97 | 3.71× | 4.81 |
| | RKL | 2.95× | 3.82 | 2.80× | 3.63 | 2.34× | 3.01 | 2.33× | 3.03 | 2.34× | 3.01 | 2.44× | 3.11 | 1.71× | 2.12 | 2.42× | 3.10 |
| | TV | 0.81× | 1.02 | 0.81× | 1.01 | 0.80× | 1.01 | 0.80× | 1.00 | 0.79× | 1.01 | 0.82× | 1.01 | 0.82× | 1.01 | 0.81× | 1.01 |
| | LK | 4.90× | 6.43 | 4.38× | 5.77 | 3.58× | 4.66 | 3.64× | 4.80 | 3.37× | 4.36 | 3.70× | 4.77 | 2.38× | 2.97 | 3.71× | 4.82 |
| | **BV (Ours)** | **5.67**× | **7.18** | **4.98**× | **6.36** | **4.12**× | **5.27** | **4.07**× | **5.17** | **3.71**× | **4.71** | **3.96**× | **5.00** | **2.56**× | **3.21** | **4.15**× | **5.27** |
| DSpark | CE | 4.97× | 6.70 | 4.61× | 6.21 | 3.76× | 5.02 | 3.80× | 5.11 | 3.45× | 4.60 | 3.75× | 4.96 | 2.43× | 3.10 | 3.82× | 5.10 |
| | KL | 5.33× | 7.32 | 4.90× | 6.83 | 4.11× | 5.58 | 4.10× | **5.57** | 3.64× | 5.08 | 3.90× | 5.31 | 2.62× | 3.34 | 4.09× | 5.58 |
| | RKL | 3.20× | 4.22 | 3.14× | 4.13 | 2.75× | 3.59 | 2.62× | 3.44 | 2.48× | 3.23 | 2.78× | 3.60 | 1.82× | 2.31 | 2.69× | 3.50 |
| | TV | 0.80× | 1.02 | 0.80× | 1.02 | 0.79× | 1.01 | 0.80× | 1.02 | 0.80× | 1.02 | 0.82× | 1.04 | 0.81× | 1.01 | 0.80× | 1.02 |
| | LK | 5.46× | 7.44 | 5.02× | 6.79 | 4.10× | 5.47 | 4.15× | 5.55 | 3.85× | 5.06 | 4.07× | 5.36 | 2.60× | 3.34 | 4.18× | 5.57 |
| | **BV (Ours)** | **5.78**× | **7.64** | **5.29**× | **6.95** | **4.41**× | **5.69** | **4.32**× | 5.56 | **3.99**× | **5.09** | **4.23**× | **5.38** | **2.63**× | **3.37** | **4.38**× | **5.67** |
| $T = 1$, token verify | | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ |
| DFlash | CE | 4.23× | 5.31 | 3.74× | 4.66 | 2.95× | 3.65 | 3.31× | 4.12 | 3.03× | 3.75 | 3.39× | 4.16 | 2.11× | 2.62 | 3.25× | 4.04 |
| | KL | 4.26× | 5.44 | 3.75× | 4.76 | 3.07× | 3.89 | 3.30× | 4.18 | 3.00× | 3.78 | 3.36× | 4.20 | 2.12× | 2.65 | 3.27× | 4.13 |
| | RKL | 2.98× | 3.69 | 2.80× | 3.46 | 2.29× | 2.82 | 2.34× | 2.89 | 2.38× | 2.92 | 2.58× | 3.13 | 1.68× | 2.08 | 2.44× | 3.00 |
| | TV | 0.84× | 1.02 | 0.84× | 1.01 | 0.84× | 1.01 | 0.83× | 1.00 | 0.84× | 1.01 | 0.84× | 1.01 | 0.82× | 1.01 | 0.83× | 1.01 |
| | LK | 4.39× | 5.56 | 3.85× | 4.85 | 3.12× | 3.87 | 3.33× | 4.20 | 3.11× | 3.86 | 3.47× | 4.30 | 2.17× | 2.71 | 3.35× | 4.19 |
| | **BV (Ours)** | **5.05**× | **6.53** | **4.43**× | **5.68** | **3.60**× | **4.60** | **3.76**× | **4.82** | **3.48**× | **4.43** | **3.83**× | **4.90** | **2.43**× | **3.05** | **3.80**× | **4.86** |
| DSpark | CE | 4.63× | 6.14 | 4.18× | 5.52 | 3.46× | 4.55 | 3.48× | 4.58 | 3.23× | 4.24 | 3.43× | 4.47 | 2.17× | 2.90 | 3.51× | 4.63 |
| | KL | 4.74× | 6.53 | 4.32× | 5.91 | 3.45× | 4.70 | 3.58× | 4.92 | 3.32× | 4.50 | 3.51× | 4.71 | 2.26× | 3.03 | 3.60× | 4.90 |
| | RKL | 3.03× | 4.09 | 2.89× | 3.89 | 2.53× | 3.39 | 2.44× | 3.30 | 2.34× | 3.13 | 2.72× | 3.59 | 1.68× | 2.24 | 2.52× | 3.38 |
| | TV | 0.76× | 1.02 | 0.77× | 1.02 | 0.76× | 1.01 | 0.76× | 1.02 | 0.76× | 1.02 | 0.79× | 1.04 | 0.77× | 1.01 | 0.77× | 1.02 |
| | LK | 4.97× | 6.79 | 4.47× | 6.08 | 3.44× | 4.66 | 3.77× | 5.11 | 3.46× | 4.65 | 3.72× | 4.96 | 2.32× | 3.11 | 3.74× | 5.05 |
| | **BV (Ours)** | **5.18**× | **7.13** | **4.57**× | **6.27** | **3.74**× | **5.14** | **3.83**× | **5.18** | **3.55**× | **4.77** | **3.80**× | **5.13** | **2.38**× | **3.19** | **3.86**× | **5.26** |
| $T = 1$, block verify | | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ |
| DFlash | CE | 4.29× | 5.49 | 3.77× | 4.80 | 3.10× | 3.91 | 3.32× | 4.21 | 3.03× | 3.81 | 3.36× | 4.18 | 2.13× | 2.67 | 3.29× | 4.15 |
| | KL | 4.38× | 5.52 | 3.90× | 4.89 | 3.29× | 4.07 | 3.35× | 4.20 | 3.08× | 3.82 | 3.39× | 4.17 | 2.13× | 2.67 | 3.36× | 4.19 |
| | RKL | 2.95× | 3.68 | 2.78× | 3.46 | 2.36× | 2.91 | 2.35× | 2.95 | 2.40× | 2.94 | 2.57× | 3.11 | 1.66× | 2.07 | 2.44× | 3.02 |
| | TV | 0.82× | 1.02 | 0.81× | 1.01 | 0.82× | 1.01 | 0.81× | 1.00 | 0.81× | 1.01 | 0.82× | 1.01 | 0.81× | 1.01 | 0.81× | 1.01 |
| | LK | 4.38× | 5.70 | 3.92× | 4.98 | 3.14× | 3.92 | 3.39× | 4.27 | 3.14× | 3.96 | 3.51× | 4.36 | 2.21× | 2.79 | 3.38× | 4.28 |
| | **BV (Ours)** | **5.28**× | **6.79** | **4.46**× | **5.82** | **3.82**× | **4.89** | **3.83**× | **4.91** | **3.49**× | **4.47** | **3.85**× | **4.90** | **2.45**× | **3.08** | **3.88**× | **4.98** |
| DSpark | CE | 4.66× | 6.38 | 4.16× | 5.65 | 3.32× | 4.49 | 3.45× | 4.70 | 3.23× | 4.35 | 3.37× | 4.50 | 2.18× | 2.93 | 3.48× | 4.71 |
| | KL | 4.94× | 6.75 | 4.50× | 6.11 | 3.66× | 4.93 | 3.77× | 5.13 | 3.47× | 4.65 | 3.62× | 4.80 | 2.34× | 3.14 | 3.76× | 5.07 |
| | RKL | 3.03× | 4.10 | 2.90× | 3.93 | 2.51× | 3.40 | 2.49× | 3.38 | 2.35× | 3.16 | 2.67× | 3.55 | 1.69× | 2.27 | 2.52× | 3.40 |
| | TV | 0.77× | 1.02 | 0.77× | 1.02 | 0.77× | 1.01 | 0.77× | 1.02 | 0.77× | 1.02 | 0.80× | 1.04 | 0.76× | 1.01 | 0.77× | 1.02 |
| | LK | 5.12× | 7.07 | 4.62× | 6.35 | 3.68× | 4.98 | 3.73× | 5.11 | 3.48× | 4.71 | 3.70× | 4.98 | 2.38× | 3.20 | 3.82× | 5.20 |
| | **BV (Ours)** | **5.34**× | **7.33** | **4.72**× | **6.49** | **3.95**× | **5.33** | **4.04**× | **5.47** | **3.67**× | **4.92** | **3.87**× | **5.19** | **2.41**× | **3.25** | **4.00**× | **5.43** |