DFLOW: ENABLING VERIFIER INFORMATION FLOW IN BLOCK DIFFUSION SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Block diffusion speculative decoding improves LLM inference efficiency by proposing a block of future tokens in parallel and verifying them with a single forward pass through the target model. However, existing methods retain only the accepted prefix and discard the rejected suffix, preventing the computation spent on these positions from benefiting subsequent drafting rounds and forcing the drafter to repeatedly reconstruct representations for future tokens from scratch. We observe that rejection only determines whether a proposed token can be committed, while the verifier representations at rejected positions can still provide useful information for subsequent predictions. Based on this observation, we propose DFlow, a simple yet effective framework that enables verifier information to flow across drafting rounds. DFlow reuses the hidden states produced by the target verifier for the rejected suffix to guide subsequent drafting. To effectively learn this information flow across drafting rounds, we introduce a self-condition train strategy that feeds verifier representations from earlier predictions back into subsequent predictions. Experiments on Qwen3 models across diverse benchmarks demonstrate that DFlow consistently improves both acceptance length and end-to-end decoding speedup over DFlash. Our code will be openly released on GitHub.

1 INTRODUCTION

Large language models (LLMs) have achieved remarkable performance across a wide range of tasks (Singh et al., 2025; DeepSeek-AI, 2026; Kimi Team, 2026), but autoregressive decoding remains inherently sequential and often becomes a major bottleneck in LLM inference (Miao et al., 2025; Nie et al., 2025). Speculative decoding alleviates this bottleneck by using a lightweight drafter to propose multiple future tokens and verifying them in parallel with the target model (Leviathan et al., 2023; Li et al., 2026). Among recent approaches, block diffusion speculative decoding is particularly attractive because it predicts an entire block of future tokens simultaneously (Sandler et al., 2025; Liu et al., 2026a), substantially reducing the sequential overhead of drafting while preserving exact target model outputs through verification. Despite this efficiency, its practical speedup still depends heavily on how many proposed tokens can be accepted in each drafting round (Zhang et al., 2026a; Wu et al., 2026), making draft quality a key factor in decoding acceleration.

However, existing block diffusion speculative decoding methods impose an information discontinuity across consecutive drafting rounds. Given a verified draft block, only the accepted prefix is committed as context for the next drafting round, while the remaining suffix and its intermediate representations are discarded (Chen et al., 2026a; Huang et al., 2026). In the next drafting round, the drafter predicts another block of future positions, some of which overlap with positions already predicted and processed in the previous round. Nevertheless, these overlapping positions are reset to mask embeddings and reconstructed from scratch, preventing the intermediate information obtained earlier from being carried forward. As a result, the rejection boundary serves not only as a necessary commit boundary for lossless speculative decoding, but also as an unnecessary information boundary that blocks information flow across consecutive drafting rounds.

Recent work suggests that speculative execution provides value beyond the tokens that are eventually accepted. Drafting and verification naturally produce rich intermediate artifacts, including future token predictions, attention patterns, routing signals, and contextual representations, which have increasingly been reused to anticipate KV access, prefetch experts or reduce subsequent com-

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.

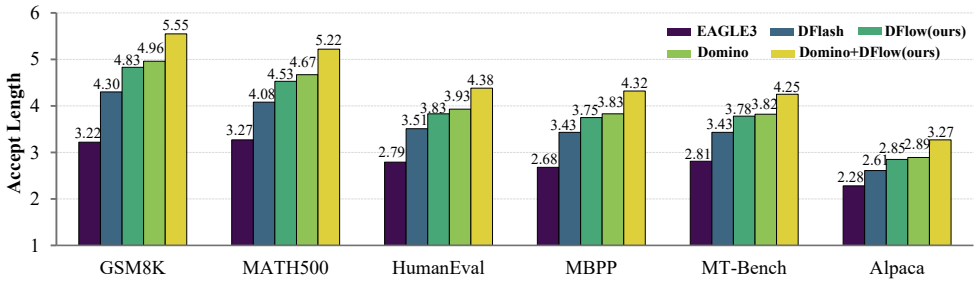

Figure 1: Average acceptance length of different methods on Qwen3-1.7B under greedy decoding.

putation (Liang et al., 2026; Chen et al., 2025; 2026b). These approaches suggest that speculative proposals can provide useful information beyond the tokens that are eventually accepted. A similar observation has been made in discrete diffusion language models, where intermediate representations can be carried across iterative generation steps to support subsequent predictions (Jo et al., 2026; Rozonoyer et al., 2026). This idea naturally extends to target verification. Rejecting a proposed token only means that it cannot be committed to the output, while the verifier representation computed at that position may still contain useful information about the continuation. The verifier representation is produced by the target model from the preceding context and the speculative continuation, and can therefore preserve contextual and semantic information even when the discrete proposal itself is rejected. As a result, the rejected suffix may still provide useful signals for subsequent drafting. Therefore, discarding these representations together with the rejected tokens may unnecessarily erase information that could benefit subsequent drafting. This motivates a simple question: can the verifier information already computed for rejected positions be carried forward to improve the next drafting round?

To this end, we propose DFlow, a simple yet effective framework that enables verifier information to flow across drafting rounds in block diffusion speculative decoding. Rather than predicting overlapping future positions again from scratch, DFlow carries forward the verifier hidden states associated with the rejected suffix and uses them to guide subsequent drafting. These representations are obtained from the existing verification process and therefore introduce no additional target model forward pass. To support this cross round inference pattern, we further introduce Multi-Round Self-Conditioned Training, which explicitly unfolds consecutive drafting and verification rounds, allowing the training procedure to reproduce the verifier information flow that arises during DFlow inference. As shown in Figure 1, DFlow consistently improves draft quality over DFlash across diverse reasoning, coding, and dialogue tasks, demonstrating the effectiveness of reusing verifier information beyond the rejection boundary. Our main contributions are summarized as follows:

1. We identify an overlooked information discontinuity in block diffusion speculative decoding: Even though verifier representations beyond the rejection boundary may still provide useful information for subsequent drafting, they are discarded together with the rejected tokens.
2. We propose **DFlow**, a simple and effective framework that enables verifier information flow across drafting rounds by relaying rejected suffix representations to corresponding positions in the next round, with Multi-Round Self-Conditioned Training to learn this information reuse.
3. We validate DFlow on Qwen3-1.7B, Qwen3-4B, and Qwen3-8B across diverse reasoning, coding, and dialogue benchmarks. Under greedy decoding, DFlow improves the average acceptance length over DFlash by **10.4%**, **13.2%**, and **13.4%** on the three model scales respectively.

2 Related Work

Speculative Decoding for Efficient LLM Inference. Speculative decoding addresses the memory bound bottleneck of LLM decoding by amortizing target model memory accesses across multiple candidate tokens through parallel verification in a single forward pass (Chen et al., 2023; Miao et al., 2023; Zhang et al., 2026b). A representative line of work is the EAGLE series (Li et al., 2024a;b; 2025b), which performs autoregressive speculative drafting conditioned on target model features. To reduce the sequential overhead of such autoregressive drafting, DFlash introduces block diffusion speculative decoding, generating an entire draft block in parallel with a single draft model

forward pass (Chen et al., 2026a). As drafting latency decreases, draft quality becomes the primary bottleneck to further speedup, motivating recent efforts to strengthen block parallel drafters (Yang & Li, 2026). DFlare improves the use of target model representations by allowing each draft layer to learn its own combination of features from multiple target layers (Zhang et al., 2026a), which provides more suitable information for different draft depths and enables deeper draft models to scale effectively. Domino focuses on the weakened causal dependency modeling within a draft block and introduces a lightweight causal head to refine parallel predictions (Huang et al., 2026). DSpark and DFly further explore this direction and validate such dependency modeling mechanisms under practical serving workloads (Cheng et al., 2026; Liu et al., 2026b). More recently, DFlash 2 explores local dependency modeling with lightweight dynamic convolutions while preserving fully parallel drafting (Inco AI, 2026). Together, these methods improve draft quality and model capacity while maintaining the low drafting cost of block diffusion speculative decoding.

Intermediate Information Reuse in Speculative Decoding. Recent work has increasingly explored how intermediate information produced during speculative decoding can be reused beyond its immediate role in token generation or verification. On the draft side, speculative predictions, routing decisions, and hidden states have been exploited to anticipate KV access, prefetch experts, or support subsequent speculation (Liang et al., 2026; Li et al., 2025a; Chen et al., 2026b). On the verification side, attention patterns and KV importance exposed during target verification have been reused to guide sparse attention and context selection in later drafting steps (Zhao et al., 2026). However, existing reuse mechanisms mainly exploit intermediate artifacts to precompute or selectively perform subsequent operations, thereby reducing inference overhead. In contrast, DFlow introduces a different form of information reuse that directly connects target verification with subsequent drafting. It carries forward verifier representations from positions beyond the rejection boundary and uses them as guidance for the next drafting round. This enables information computed by the target model during verification to directly improve future draft quality.

3 PRELIMINARIES

To facilitate the understanding of DFlow, we first review the inference process of DFlash and introduce the notation used in this paper. DFlash performs generation through a sequence of speculative rounds, each consisting of block diffusion drafting and exact-match verification.

Block Diffusion Drafting. Let $\mathbf{y}_{1:t_r}$ denote the committed context at speculative round r, and let K denote the number of future tokens drafted in parallel. DFlash conditions a lightweight block diffusion drafter on hidden representations extracted from the target model. Specifically, let $\mathcal{L} = \{\ell_1, \ldots, \ell_m\}$ denote a fixed set of target layers sampled from shallow to deep. The hidden states from these layers are concatenated and projected into a fused target context representation:

$$\mathbf{H}_T^{(r)} = \mathrm{RMSNorm}\left(\mathbf{W}_c \left[\mathbf{H}_T^{(r,\ell_1)}; \ldots; \mathbf{H}_T^{(r,\ell_m)}\right]\right), \tag{1}$$

where $\mathbf{H}_T^{(r,\ell)}$ denotes the hidden states extracted from target layer ℓ, and $\mathbf{W}_c$ denotes the projection matrix. The fused representation $\mathbf{H}_T^{(r)}$ is then injected into the Key and Value projections of each draft layer, providing persistent conditioning from the committed context.

To predict the next K tokens, DFlash initializes the future positions with mask embeddings and jointly predicts the draft block conditioned on the fused target context representation:

$$\hat{\mathbf{y}}^{(r)} = \mathcal{D}_\theta\left(\mathbf{E}_{\mathrm{mask}}^{(r)}; \mathbf{H}_T^{(r)}\right), \tag{2}$$

where $\mathcal{D}_\theta$ denotes the block diffusion drafter and $\mathbf{E}_{\mathrm{mask}}^{(r)}$ denotes the mask representations for the K future positions. While $\mathbf{H}_T^{(r)}$ provides target information from the committed context, the future positions are initialized from mask embedding at each speculative round, even when some of them overlap with positions already drafted and processed in the preceding round.

Exact Match Verification. Given the draft block $\hat{\mathbf{y}}^{(r)}$, the target model verifies the proposed tokens from left to right using exact token matching. Let a_r denote the number of consecutive proposals that match the corresponding target predictions before the first mismatch. Accordingly, the verified

draft block can be partitioned as

$$\hat{\mathbf{y}}^{(r)} = \Big[\underbrace{\hat{\mathbf{y}}^{(r)}_{1:a_r}}_{\text{accepted prefix}} , \underbrace{\hat{y}^{(r)}_{a_r+1}}_{\text{first rejection}} , \underbrace{\hat{\mathbf{y}}^{(r)}_{a_r+2:K}}_{\text{rejected suffix}} \Big] . \tag{3}$$

The first a_r proposals are committed as the *accepted prefix*. At the first rejected position, the draft proposal is replaced by the corresponding target prediction, denoted by $b^{(r)}$ and referred to as the *bonus token*. The remaining proposals, referred to as the *rejected suffix*, are discarded after verification. The rejected suffix may overlap with the beginning of the subsequent drafting window, so some future positions are drafted from scratch again in the next round. Yet the information obtained at these positions is not retained across rounds, introducing an information discontinuity.

4 DFLOW

DFlow addresses the information discontinuity across speculative rounds by preserving verifier representations associated with rejected draft positions and propagating them into subsequent drafting. During inference, the verifier representations are propagated to their corresponding future positions and incorporated into the input of the subsequent drafting round. To accommodate this information flow across multiple drafting rounds, we further introduce a self-conditioning strategy that trains the drafter under both standard mask prediction and prediction with verifier information.

4.1 INFORMATION DISCONTINUITY ACROSS DRAFTING ROUNDS

Existing block diffusion speculative decoding suffers from an information discontinuity across drafting rounds. During exact match verification, only the accepted prefix and the bonus token are committed, whereas the rejected suffix is discarded. However, positions in the rejected suffix remain unresolved after the current round and may need to be predicted again as generation proceeds. Although the target model has already processed these positions and produced contextual representations during verification, such information is discarded together with the rejected proposals. Consequently, the rejection boundary not only determines which tokens can be committed, but also becomes an information boundary between consecutive drafting rounds.

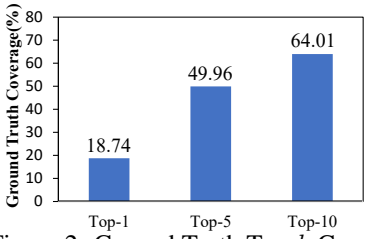

Figure 2: Ground Truth Top-k Coverage in the Rejected Suffix.

Importantly, rejection is a decision over discrete token commitment rather than a direct measure of information utility. As illustrated in Figure 2, tokens beyond the first rejection can still contain useful predictive signals, with some positions remaining consistent with the target continuation despite belonging to the rejected suffix. More importantly, during verification, each draft proposal participates in the target model's causal computation, inducing a contextual representation that integrates information from the preceding context and previously proposed tokens. Such representations are directly involved in predicting subsequent tokens and may therefore retain useful information even when the corresponding discrete proposals are rejected. This motivates the following hypothesis:

Main Hypothesis. Verifier representations at rejected positions can provide useful guidance for subsequent drafting, improving draft quality by preserving information that would otherwise be discarded.

4.2 VERIFIER INFORMATION RELAY

Based on the above hypothesis, DFlow relays verifier representations associated with the rejected suffix into subsequent drafting rounds and reuses them at the corresponding overlapping positions as auxiliary information, allowing these positions to inherit information from previous verification rather than reconstructing their representations from scratch in each round. To make the relayed information compatible with subsequent drafting, DFlow first transforms verifier states into the drafter

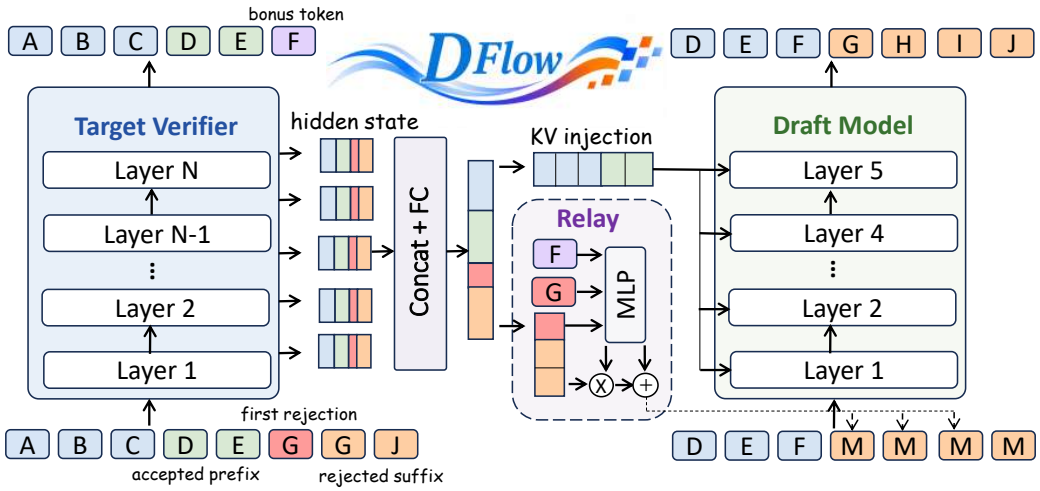

Figure 3: **Pipeline of DFlow.** DFlow reuses verifier hidden states from rejected suffix positions through the relay module to enhance the corresponding mask embeddings in the next drafting round, without additional target model forward passes.

hidden space while maintaining their original prediction correspondence, and then refines them according to the semantic change introduced at the rejection boundary.

Verifier Representation Alignment. To make the verifier hidden states compatible with the drafter hidden space, DFlow shares the same feature projection used by DFlash for target context conditioning. Specifically, hidden states from the selected target layers over the rejected suffix are concatenated and transformed by the shared projection matrix $\mathbf{W}_c$ followed by RMSNorm, yielding verifier representations $\mathbf{h}_i^{(r)}$ for subsequent reuse. Since target verification already produces the required hidden states, DFlow requires no additional target model forward pass. Moreover, the reusable suffix states are jointly projected with the target features required by DFlash, introducing only a small number of additional positions.

The target verifier and the block diffusion drafter further differ in how their representations correspond to predicted tokens. The causal target model performs next token prediction, where the hidden state at position i predicts the token at position $i+1$, whereas the drafter performs mask token prediction, where each masked position predicts the token at the same position. To align verifier representations with the drafter, DFlow shifts the verifier representations before injecting them into the drafter, so that both representations correspond to the same predicted token. The aligned representations are then injected into the corresponding unresolved positions of the subsequent draft.

Reject Conditioned Draft Initialization. Although the aligned verifier representations contain useful information about the rejected suffix, they are produced under a speculative context that differs from the context used in the subsequent drafting round. During verification, the rejected suffix is processed following the first rejected proposal $\hat{y}_{a_r+1}^{(r)}$, whereas the next round continues from the target correction $b^{(r)}$ at the same position. The difference between these two boundary tokens therefore provides a direct signal of how the context underlying the verifier representations changes across rounds.

DFlow uses this boundary information to determine how the verifier representation at each reusable position should be retained and corrected. Specifically, we combine the embeddings of the target correction and the first rejected proposal with $\mathbf{h}_i^{(r)}$, and use a lightweight MLP to produce channel-wise scaling and correction vectors:

$$\left[\boldsymbol{\beta}_i^{(r)}; \boldsymbol{\gamma}_i^{(r)}\right] = \mathrm{MLP}_\phi\left(\left[\mathbf{E}\left(b^{(r)}\right); \mathbf{E}\left(\hat{y}_{a_r+1}^{(r)}\right); \mathbf{h}_i^{(r)}\right]\right), \tag{4}$$

where $\boldsymbol{\beta}_i^{(r)}$ controls the contribution of the verifier representation and $\boldsymbol{\gamma}_i^{(r)}$ provides an additive correction. This channel-wise modulation allows DFlow to selectively preserve, suppress, or adjust the relayed information according to the change at the rejection boundary.

The verifier representation is first normalized and then adjusted by the learned scaling and correction vectors before being added to the standard mask embedding, forming the input for the subsequent drafting round:

$$\mathbf{x}_i^{(r+1)} = \mathbf{E}_{\text{mask}} + \boldsymbol{\beta}_i^{(r)} \odot \text{LN}\left(\mathbf{h}_i^{(r)}\right) + \boldsymbol{\gamma}_i^{(r)}, \tag{5}$$

where $\text{LN}(\cdot)$ denotes LayerNorm. This augmented input is used only at positions with valid reusable verifier representations, while the remaining positions continue to use the standard mask embedding. In the first drafting round, where no verifier information is available, or when the previous verification produces no reusable rejected suffix, DFlow simply uses mask embeddings as the draft inputs.

4.3 Multi-Round Self-Conditioned Training

Standard DFlash training treats each sampled drafting round independently and initializes every new round from mask embedding, whereas DFlow inference conditions later drafts on verifier representations produced in preceding rounds. To bridge this gap, DFlow introduces Multi-Round Self-Conditioned Training, consisting of a Multi-Round Rollout that explicitly unfolds consecutive drafting and verification rounds to propagate verifier information, and a Trajectory-Guided Transition that determines the starting position of each subsequent round by matching draft proposals with the training trajectory.

Multi-Round Rollout. Following DFlash, we sample anchor positions from trajectories generated by the target model and construct multiple draft blocks that can be packed for parallel training. For each sampled anchor, DFlow extends the original single round training into a sequence of consecutive drafting rounds. The initial round follows standard DFlash training and predicts the draft block from mask embeddings, preserving the fundamental mask prediction capability of the drafter while producing the proposals required for subsequent verification. These proposals are processed by the frozen target model to obtain verifier representations over the rejected suffix, which are then relayed to the next round. Subsequent rounds perform reject conditioned prediction, allowing the drafter to learn how to exploit verifier information from the preceding verification rather than relying solely on mask embeddings. The reject conditioned stage can continue for multiple rounds, with the rollout depth k controlling how many such rounds are included during training. All unfolded rounds share the same drafter parameters, and their prediction losses jointly optimize this shared parameter set. In practice, we unfold $k = 2$ reject conditioned rounds after the initial mask prediction.

Target Trajectory Transition. Each transition in the multi-round rollout requires determining both where the next drafting round begins and what verifier information is carried forward from the current round. The starting position of the next round depends on the acceptance result of the current draft, with the new round anchor placed at the bonus token corresponding to the first mismatch. Although target verification can directly determine the accepted prefix and bonus token, its verification result during training may occasionally differ from the precomputed target trajectory due to sampling or numerical differences. We therefore determine the round transition by exact matching the draft proposals against the target trajectory and use the target token at the first mismatch as the bonus token. This keeps subsequent rounds aligned with the same target trajectory, allowing the target context computed for each training sample to be directly reused while verification in later rounds only processes the newly proposed block. Importantly, the target trajectory is used only to determine the transition between consecutive training rounds. The verifier information carried into the next round remains tied to the actual draft proposals produced in the current round. These proposals are processed by the frozen target model, and the verifier representations associated with the rejected suffix are extracted from this verification pass for subsequent reject conditioned prediction.

Following DFlash, we apply an exponential decay to the cross entropy loss as the draft position moves farther from the anchor, assigning greater importance to earlier predictions within each block. The overall training objective sums the loss of the initial mask prediction and those of all subsequent reject conditioned rounds:

$$\mathcal{L}_{\text{DFlow}} = \mathcal{L}^{(0)} + \sum_{q=1}^{k} \mathcal{L}^{(q)}, \tag{6}$$

where k denotes the rollout depth, $\mathcal{L}^{(0)}$ is the loss of the initial mask prediction, and $\mathcal{L}^{(q)}$ for $q \geq 1$ denotes the loss of the q-th reject conditioned round. The losses from all rounds are summed before backpropagation and jointly optimize the same shared trainable drafter parameters. The target

Table 1: End-to-end decoding speedup over standard autoregressive decoding and average acceptance length τ under low-concurrency settings for greedy and sampling decoding.

		Math				Code				Chat				Overall	
Model	**Method**	GSM8K		MATH-500		HumanEval		MBPP		MT-Bench		Alpaca		Avg.	
Temperature = 0		Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ
Qwen3-1.7B	EAGLE-3	2.34	3.22	2.38	3.27	2.03	2.79	1.95	2.68	1.90	2.81	1.61	2.28	2.03	2.84
	DFlash	3.32	4.30	3.17	4.08	2.71	3.51	2.63	3.43	2.38	3.43	1.95	2.61	2.69	3.56
	DFlow	**3.66**	**4.83**	**3.49**	**4.53**	**2.92**	**3.83**	**2.85**	**3.75**	**2.58**	**3.78**	**2.07**	**2.85**	**2.93**	**3.93**
Qwen3-4B	EAGLE-3	2.81	3.56	2.88	3.63	2.57	3.27	2.45	3.10	2.10	2.91	1.85	2.45	2.44	3.15
	DFlash	3.57	4.41	3.54	4.39	3.08	3.86	2.93	3.61	2.40	3.39	2.01	2.61	2.92	3.71
	DFlow	**4.09**	**5.13**	**3.90**	**4.92**	**3.41**	**4.33**	**3.30**	**4.12**	**2.62**	**3.80**	**2.19**	**2.90**	**3.25**	**4.20**
Qwen3-8B	EAGLE-3	3.17	4.00	3.27	4.12	2.79	3.50	2.74	3.47	2.25	3.09	2.00	2.66	2.70	3.47
	DFlash	3.57	4.40	3.53	4.36	3.10	3.82	2.88	3.55	2.33	3.21	2.02	2.61	2.90	3.66
	DFlow	**4.08**	**5.10**	**3.93**	**4.94**	**3.43**	**4.28**	**3.27**	**4.08**	**2.55**	**3.58**	**2.22**	**2.93**	**3.25**	**4.15**
Temperature = 1		Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ
Qwen3-1.7B	EAGLE-3	2.09	3.06	2.10	3.08	1.85	2.71	1.80	2.63	1.66	2.61	1.44	2.18	1.82	2.71
	DFlash	2.75	3.66	2.51	3.34	2.28	2.96	2.22	2.87	1.92	2.82	1.65	2.27	2.22	2.99
	DFlow	**3.03**	**4.04**	**2.72**	**3.65**	**2.46**	**3.21**	**2.41**	**3.17**	**2.01**	**3.04**	**1.75**	**2.42**	**2.40**	**3.26**
Qwen3-4B	EAGLE-3	2.56	3.42	2.54	3.40	2.35	3.15	2.30	3.06	1.91	2.74	1.69	2.35	2.23	3.02
	DFlash	3.06	3.84	2.78	3.53	2.68	3.33	2.52	3.12	2.00	2.77	1.76	2.32	2.47	3.15
	DFlow	**3.45**	**4.40**	**3.10**	**3.97**	**2.94**	**3.70**	**2.84**	**3.55**	**2.15**	**3.13**	**1.89**	**2.49**	**2.73**	**3.54**
Qwen3-8B	EAGLE-3	2.84	3.76	2.81	3.75	2.46	3.27	2.47	3.30	1.96	2.84	1.79	2.51	2.39	3.24
	DFlash	3.05	3.84	2.80	3.54	2.62	3.21	2.49	3.06	1.93	2.67	1.73	2.30	2.44	3.10
	DFlow	**3.51**	**4.31**	**3.03**	**3.96**	**2.84**	**3.58**	**2.73**	**3.44**	**2.06**	**2.92**	**1.89**	**2.50**	**2.68**	**3.45**

model remains frozen throughout training, while the verifier representations and round transitions are detached from the computation graph, preventing gradients from propagating through target verification or across transitions between consecutive rounds.

5 Experiments

5.1 Experimental Setup

Models and Evaluation. We evaluate DFlow on Qwen3-1.7B, Qwen3-4B, and Qwen3-8B across three representative task categories: mathematical reasoning, code generation, and open-ended dialogue. For mathematical reasoning, we use GSM8K (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2021); for code generation, we evaluate on HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021); and for dialogue, we use MT-Bench (Zheng et al., 2023) and Alpaca (Taori et al., 2023). We report average acceptance length τ to characterize draft quality and end-to-end decoding speedup over standard autoregressive decoding.

Baselines and Comparisons. We compare DFlow with vanilla autoregressive decoding and representative speculative decoding methods, including EAGLE-3 (Li et al., 2025b) and DFlash (Chen et al., 2026a). EAGLE-3 represents autoregressive drafting, whereas DFlash represents parallel drafting. To ensure a fair comparison, all speculative decoding methods are trained on ShareGPT[1].

Training and Implementation Details. We train a DFlow with five layers and a draft block size of 16 from scratch for all target models. Unless otherwise specified, Multi-Round Self-Conditioned Training uses $k = 2$ reject conditioned rounds following the initial mask prediction. All experiments are conducted on NVIDIA A800 GPUs. Further details are provided in Appendix A.1.

5.2 Main Results

Low Concurrency. As shown in Table 1, DFlow consistently outperforms DFlash in both acceptance length and end-to-end decoding speedup across model scales and task categories. The improvements remain consistent under both greedy and sampling decoding, indicating that verifier information from rejected suffixes reliably improves draft quality and that these gains translate into practical inference acceleration.

High Concurrency. We further evaluate DFlow under concurrent serving workloads using SGLang. Table 2 reports serving throughput on Qwen3-4B and Qwen3-8B under concurrency levels from 2 to

[1] https://huggingface.co/datasets/anon8231489123/ShareGPT_Vicuna_unfiltered

Table 2: Throughput under varying concurrency levels on Qwen3 models, reporting absolute TPS together with speedup over the autoregressive baseline.

Task	Method	Concurrency				
		2	4	8	16	32
Qwen3-4B						
GSM8K	Baseline	290.83	557.28	1082.11	2004.83	3577.60
	DFlash	686.68$_{2.36\times}$	1236.19$_{2.22\times}$	2042.07$_{1.89\times}$	2766.79$_{1.38\times}$	3520.41$_{0.98\times}$
	DFlow	**748.76**$_{2.57\times}$	**1361.85**$_{2.44\times}$	**2221.44**$_{2.05\times}$	**3039.66**$_{1.52\times}$	**3902.61**$_{1.09\times}$
HumanEval	Baseline	289.75	552.34	1060.28	1941.07	3343.50
	DFlash	599.38$_{2.07\times}$	1090.03$_{1.97\times}$	1818.10$_{1.71\times}$	2463.64$_{1.27\times}$	3082.04$_{0.92\times}$
	DFlow	**636.06**$_{2.20\times}$	**1160.09**$_{2.10\times}$	**1937.10**$_{1.83\times}$	**2649.94**$_{1.37\times}$	**3353.55**$_{1.00\times}$
Qwen3-8B						
GSM8K	Baseline	183.33	360.18	696.11	1313.57	2352.60
	DFlash	489.45$_{2.67\times}$	916.72$_{2.55\times}$	1501.79$_{2.16\times}$	2045.76$_{1.56\times}$	2371.70$_{1.01\times}$
	DFlow	**542.34**$_{2.96\times}$	**1007.86**$_{2.80\times}$	**1662.23**$_{2.39\times}$	**2272.52**$_{1.73\times}$	**2641.66**$_{1.12\times}$
HumanEval	Baseline	182.80	358.21	688.28	1287.21	2262.50
	DFlash	434.24$_{2.38\times}$	824.63$_{2.30\times}$	1355.09$_{1.97\times}$	1830.63$_{1.42\times}$	2125.48$_{0.94\times}$
	DFlow	**459.51**$_{2.51\times}$	**870.19**$_{2.43\times}$	**1420.23**$_{2.06\times}$	**1959.67**$_{1.52\times}$	**2269.77**$_{1.00\times}$

Table 3: Ablation of verifier information relay.

Training	Mask	Relay
DFlash	3.56	–
DFlash + Self-Conditioning	3.67	–
DFlow	3.66	**3.93**

Table 4: Compatibility with Domino.

Method	τ
Domino	4.02
+ DFlow (w/o Online Rollout)	2.22
+ DFlow (Online Rollout)	**4.50**

32. Across both model scales and workloads, DFlow consistently achieves higher throughput than DFlash, demonstrating that improvements in draft quality translate into practical serving efficiency. These results also indicate that the overhead introduced by verifier information relay remains modest compared with the gain from accepting more draft tokens, allowing DFlow to retain its advantage as concurrency increases.

5.3 Ablation Studies

Effect of Verifier Information Relay. To quantify the respective contributions of Multi-Round Self-Conditioned Training and verifier information relay, we conduct controlled ablations. We first retain DFlow's multi-round rollout while resetting each new round to mask embeddings, thereby isolating the benefit of multi-round rollout from verifier information reuse. This improves the average acceptance length from 3.56 to 3.67, indicating that multi-round training alone provides limited benefit. We then disable verifier relay in the trained DFlow model, which yields a similar acceptance length of 3.66. In contrast, enabling verifier information relay increases the acceptance length to 3.93 under the same trained model. These results show that the main improvement of DFlow comes from reusing verifier information across drafting rounds.

Runtime Overhead Analysis. We decompose the latency of each speculative round to quantify the additional cost introduced by DFlow. As shown in Figure 4, the dominant computation remains draft model execution and target verification, while verifier information relay introduces only 0.51 ms of additional latency, corresponding to merely 1.2% overhead over DFlash. This low overhead is mainly because the required hidden states are already produced during target verification, and DFlow directly reuses the target feature projection of DFlash. The additional computation is therefore limited to processing a small number of reusable rejected positions and applying the lightweight relay module. Compared with the substantial improvement in acceptance length, this additional cost is negligible, allowing the improved draft quality to translate into higher decoding throughput.

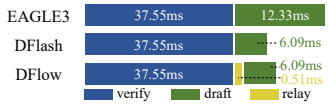

Figure 4: Runtime overhead breakdown of different methods on Qwen3 8B.

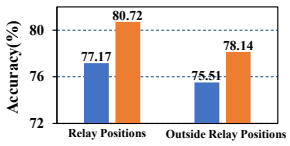

Figure 5: Analysis of accuracy from verifier information relay in DFlow.

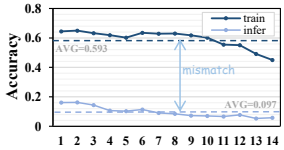

Figure 6: Accuracy mismatch by distance from the first rejection in Domino.

6 Discussion

Analysis of accuracy gains from verifier information relay. To understand where the gains from verifier information relay arise, we divide each draft block into Relay Positions, where reusable verifier representations are available from the preceding round, and Outside Relay Positions, where no such representations exist and mask embeddings are used. We measure the conditional acceptance rate, i.e., the probability of accepting the current position given that all previous positions have been accepted. As shown in Figure 5, relay improves the conditional acceptance rate from 77.17% to 80.72% in Relay Positions, showing that reusing verifier representations at overlapping positions directly translates into higher acceptance probability. Interestingly, Outside Relay Positions also improve from 75.51% to 78.14%, suggesting that the benefit of relay can propagate beyond directly reused positions by providing more informative preceding representations for later positions within the same draft block.

Compatibility with Domino. Recent studies have shown that autoregressive heads can effectively recover causal dependencies among parallel draft tokens and substantially improve draft quality (Cheng et al., 2026). We therefore examine whether DFlow remains beneficial when combined with this stronger drafting paradigm, using Domino as a representative method that decouples causal dependency modeling from expensive autoregressive draft execution. Domino progressively refines predictions across positions within each drafting round (Huang et al., 2026), whereas DFlow propagates verifier information across rounds. To combine the two, we introduce an *online rollout* during DFlow training to ensure that the rejected suffix used for relay construction follows the same draft trajectory as inference.

Domino is trained with teacher forcing, where each causal prediction is conditioned on the ground truth prefix and contributes to the cross entropy objective of the current round. Directly using these predictions to determine the rejected suffix and extract its verifier representations for DFlow, however, introduces a substantial mismatch with inference, where each position is conditioned on previously generated draft tokens. As shown in Figure 6, the rejected suffix obtained under teacher forcing exhibits substantially higher prediction accuracy than that encountered during inference, resulting in unrealistically high quality rejected suffixes for relay training. We therefore retain teacher forcing for optimizing Domino, while separately performing online rollout with its inference-time predictions to determine the rejection boundary and extract the corresponding verifier representations for DFlow. Without online rollout, the average acceptance length drops to 2.22. With online rollout, DFlow and Domino can be effectively combined, as shown in Table 4. On Qwen3 1.7B at $T = 0$, Domino achieves an average acceptance length of 4.02, compared with 3.56 for DFlash, while adding DFlow further improves it to 4.50. This result shows that verifier information reuse remains effective when combined with stronger causal dependency modeling.

7 Conclusion

We propose DFlow, a block diffusion speculative decoding framework that enables verifier information to flow across consecutive drafting rounds. DFlow relays verifier representations from the rejected suffix to subsequent unresolved positions and introduces Multi-Round Self-Conditioned Training to learn this information reuse, without requiring additional target model forward passes. Experiments on Qwen3 models across diverse benchmarks show consistent gains over DFlash in acceptance length and decoding efficiency, while remaining compatible with stronger drafting methods such as Domino. These results suggest that verifier information beyond the rejection boundary can be effectively reused to improve subsequent drafting for LLM inference acceleration.

AI Assistance Statement

Generative AI tools were used to assist with language polishing, text editing, and improving the clarity and presentation of the manuscript. All research ideas, methodological designs, experimental results, and scientific conclusions were developed and verified by the authors. The authors reviewed all AI assisted content and take full responsibility for the final manuscript.

Reproducibility Statement

We provide the model configurations, training setup, datasets, decoding settings, and evaluation protocols in Appendix A.1. We report implementation details and runtime settings necessary to reproduce the main results. The code will be released upon publication.

Ethics Statement

This work focuses on improving the inference efficiency of large language models and does not involve human subjects or the collection of personally identifiable information. DFlow does not alter the underlying capabilities or safety mechanisms of the target model. As with other techniques that reduce inference cost, improved efficiency may facilitate broader deployment of language models. The downstream use of these models remains subject to the risks and safeguards associated with the underlying models.

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. *arXiv preprint arXiv:2108.07732*, 2021.

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. *arXiv preprint arXiv:2602.06036*, 2026a.

Liangkun Chen, Zijian Wen, Tian Wu, Xiaoxi Zhang, and Chuan Wu. Sp-moe: Speculative decoding and prefetching for accelerating moe-based model inference. *arXiv preprint arXiv:2510.10302*, 2025.

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. *arXiv preprint arXiv:2107.03374*, 2021.

Yuetao Chen, Xuliang Wang, Xinzhou Zheng, Ming Li, Peng Wang, and Hong Xu. Make every draft count: Hidden state based speculative decoding. *arXiv preprint arXiv:2602.21224*, 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, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence. *CoRR*, abs/2606.19348, 2026. doi: 10.48550/ARXIV.2606.19348. URL https://doi.org/10.48550/arXiv.2606.19348.

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. *arXiv preprint arXiv:2103.03874*, 2021.

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

Inco AI. Dflash 2: Keep drafting parallel, August 2026. URL https://inco.ai/blog/dflash2/.

Mingyu Jo, Jaesik Yoon, Justin Deschenaux, Caglar Gulcehre, and Sungjin Ahn. Loopholing discrete diffusion: Deterministic bypass of the sampling wall. In *International Conference on Learning Representations*, volume 2026, pp. 93890–93918, 2026.

Kimi Team. Kimi K3: open frontier intelligence. *CoRR*, abs/2607.24653, 2026. doi: 10.48550/ARXIV.2607.24653. URL https://doi.org/10.48550/arXiv.2607.24653.

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.

Yan Li, Pengfei Zheng, Shuang Chen, Zewei Xu, Yuanhao Lai, Yunfei Du, and Zhengang Wang. Speculative moe: Communication efficient parallel moe inference with speculative token and expert pre-scheduling. *arXiv preprint arXiv:2503.04398*, 2025a.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 28935–28948. PMLR, 2024a.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7421–7432. Association for Computational Linguistics, 2024b. doi: 10.18653/v1/2024.emnlp-main.422.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *Advances in Neural Information Processing Systems*, 2025b.

Zikun Li, Zhuofu Chen, Remi Delacourt, Gabriele Oliaro, Zeyu Wang, Qinghan Chen, Shuhuai Lin, April Yang, Zhihao Zhang, Zhuoming Chen, et al. Adaserve: Accelerating multi-slo llm serving with slo-customized speculative decoding. In *Proceedings of the 21st European Conference on Computer Systems*, pp. 36–54, 2026.

Zuning Liang, Zhiyi Yao, Qi Chen, Yuedong Xu, Hao Dai, Zhiqiang Ding, Tongkai Yang, Jinlong Hou, and Yuan Cheng. Dualdecoder: Accelerate long context llm inference by predictive prefetch. *arXiv preprint arXiv:2607.26475*, 2026.

Fuliang Liu, Xue Li, Ketai Zhao, Yinxi Gao, Ziyan Zhou, Zhonghui Zhang, Zhibin Wang, Wanchun Dou, Sheng Zhong, and Chen Tian. Dart: Diffusion-inspired speculative decoding for fast llm inference. *arXiv preprint arXiv:2601.19278*, 2026a.

Hong Liu, Rui Cen, Junhan Shi, Guangshuo Qin, Jiebin Zhang, Tianyu Liu, Runzhi Fan, Guoliang Zhao, Ruobing Xie, Kai Zhang, et al. Angelspec: Towards real-world high performance inference with speculative decoding. *arXiv preprint arXiv:2607.25852*, 2026b.

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 generative large language model serving with tree-based speculative inference and verification. *arXiv preprint arXiv:2305.09781*, 2023.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Hongyi Jin, Tianqi Chen, and Zhihao Jia. Towards efficient generative large language model serving: A survey from algorithms to systems. *ACM Computing Surveys*, 58(1):1–37, 2025.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.

Benjamin Rozonoyer, Jacopo Minniti, Dhruvesh Patel, Neil Band, Avishek Joey Bose, Tim GJ Rudner, and Andrew McCallum. Learned relay representations for forward-thinking discrete diffusion models. *arXiv preprint arXiv:2605.22967*, 2026.

Jameson Sandler, Jacob K Christopher, Thomas Hartvigsen, and Ferdinando Fioretto. Specdiff-2: Scaling diffusion drafter alignment for faster speculative decoding. *arXiv preprint arXiv:2511.00606*, 2025.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models*, 2023. https://crfm.stanford.edu/2023/03/13/alpaca.html.

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.

Tianjian Yang and Meng Li. Spec-auf: Accept-until-fail training under train-inference misalignment for masked block drafters. *arXiv preprint arXiv:2607.01893*, 2026.

Jiebin Zhang, Zhenghan Yu, Song Liu, Eugene J Yu, Zheng Li, Dawei Zhu, Jiangshan Duo, Weimin Xiong, Yifan Song, Guanghua Yu, et al. Dflare: Scaling up draft capacity for block diffusion speculative decoding. *arXiv preprint arXiv:2606.02091*, 2026a.

Yaojie Zhang, Jianuo Huang, Junlong Ke, Yuhang Han, Yongji Long, Tianchen Zhao, Biqing Qi, and Linfeng Zhang. Flexdraft: Flexible speculative decoding via attention tuning and bonus-guided calibration. *arXiv preprint arXiv:2605.20022*, 2026b.

Yilong Zhao, Jiaming Tang, Kan Zhu, Zihao Ye, Chi-Chih Chang, Chaofan Lin, Jongseok Park, Guangxuan Xiao, Mohamed S Abdelfattah, Mingyu Gao, et al. Accelerating large-scale reasoning model inference with sparse self-speculative decoding. *Proceedings of Machine Learning and Systems*, 8:251–265, 2026.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, 2023.

A Appendix

A.1 Implementation Details

Baselines. We compare DFlow with two representative speculative decoding methods, EAGLE-3 and DFlash, which respectively represent autoregressive drafting and parallel block drafting. Recent extensions of DFlash, including DFlare, Domino, DSpark, and DFlash 2, primarily improve the within-round drafting process from complementary dimensions, whereas DFlow focuses on reusing verifier information across consecutive drafting rounds. To isolate the contribution of verifier information reuse, we therefore use DFlash as the controlled base architecture in our main experiments. DFlow can also be combined with stronger drafting mechanisms rather than serving as a mutually exclusive alternative. To empirically examine this complementarity, we further integrate DFlow with Domino as a representative extension and evaluate their combined performance.

Training Setup. To eliminate performance differences caused by training data, all speculative decoding methods are trained on the same ShareGPT dataset for 6 epochs. We follow the default optimization settings of each method, using a learning rate of 6×10^{-4} for DFlash and DFlow and 1×10^{-4} for EAGLE-3.

Inference Setup. For evaluation, we set the maximum number of newly generated tokens to 2048 for all methods. All experiments use FlashAttention for attention computation. All speculative decoding methods use single-path drafting during inference.

A.2 Semantic Correction Ablation

To examine the effect of semantic correction in the relay module, we compare the DFlow with a variant without reject-conditioned modulation. Across six benchmarks, direct verifier relay achieves an average acceptance length of 3.82, while DFlow with semantic correction improves it to 3.93, showing that adapting the relayed representation to the context change across drafting rounds provides additional benefit.

Table 5: Ablation of semantic correction in DFlow.

Method	**Avg.** τ
DFlow	**3.93**
w/o Semantic Correction	3.82