VFLASH-FM: TRAINING FOR ONE-PASS DRAFTING IN VIDEO LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative decoding reduces the serial cost of video language model generation, but drafting itself is constrained by the long visual context: autoregressive drafters pay for it at every proposed token, while one-pass parallel drafters lose most of a block to an early error. We introduce VFlash-FM, which improves one-pass drafting through training alone. Our key observation is that a discrete-flow path from a fully masked block to the target continuation defines a family of auxiliary prediction tasks: completing a block from the drafter's own partially revealed predictions, including its mistakes. Training the drafter on these self-generated partial states, with a step-aware flow objective and target-confidence weighting oriented toward accepted prefixes, improves the same network's predictions from all-mask inputs. The extra computation is paid once during training rather than at every decoding step, so deployment remains a single drafter forward over cached target features followed by target verification. Across five video benchmarks, VFlash-FM reaches macro decoding speedups of $3.850\times$ on LLaVA-OneVision-7B and $2.541\times$ on Qwen2.5-VL-7B, exceeding the published speedups of prior video drafting methods we compare against; on LLaVA, it improves over a non-flow drafting variant in both committed progress and speed across all tested verification configurations.

Anonymous Project Page

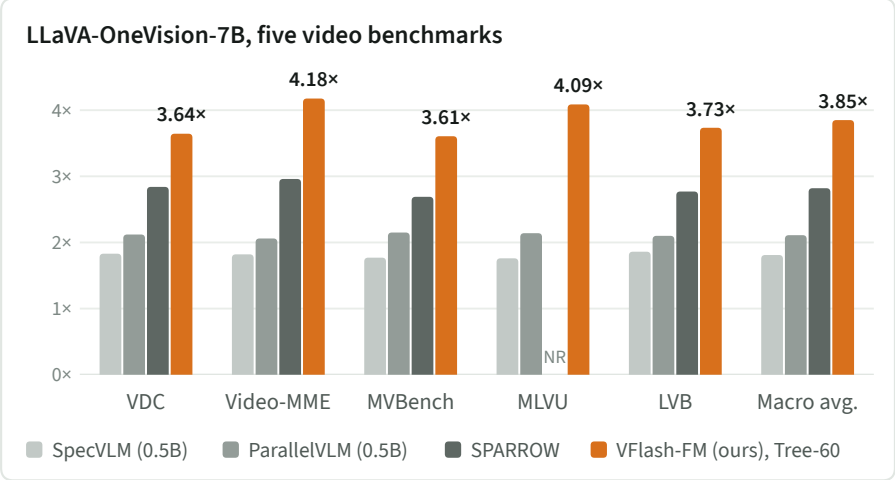


Figure 1: Decoding speedups on LLaVA-OneVision-7B, each method at its fastest listed variant (0.5B drafters for SpecVLM and ParallelVLM; VFlash-FM with Tree-60). Averages cover the benchmarks each study reports. VDC: VideoDetailCaption; LVB: LongVideoBench; NR: not reported.

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.

1 INTRODUCTION

Video language models generate responses autoregressively, so every output token requires a forward pass through a large model whose context contains thousands of visual tokens, and decoding time is dominated by this serial loop. Speculative decoding reduces the number of such passes by letting a cheaper drafter propose several tokens at once, which the target model then verifies in a single forward (Leviathan et al., 2023; Chen et al., 2023). Verification accepts draft tokens only up to the first disagreement with the target, so what a proposal actually buys is its longest accepted prefix. This makes drafting a balance between two costs: a fast drafter that errs on an early token wastes the rest of its block however cheaply it was produced, while a more accurate drafter loses its advantage if each proposal takes too long to generate.

One-pass block drafting sits at one extreme of this balance: the drafter predicts every position of a fixed-length block in a single forward pass, instead of proposing tokens sequentially. DFlash instantiates this design, predicting a fully masked block in parallel while conditioning on the target's cached hidden states (Chen et al., 2026). Because deployment always starts from a fully masked block, its training examples do too. This leaves a training question: can supervision at other stages of block completion, on blocks that are already partially and perhaps incorrectly filled in, improve the same network's prediction from masks? For video language models the question has a practical motivation. Refining a draft at inference time would attend to the long visual context again on every block, whereas additional training computation is paid once per trained drafter.

We introduce VFlash-FM, which trains a target-conditioned predictor on partial responses produced by the drafter itself. Starting from a masked block, a sequence of no-gradient forwards reveals its most confident predictions. Correct reveals supply useful context for neighboring predictions; wrong reveals require the network to recover the target continuation despite its own mistakes. The target's reference continuation remains the supervision in both cases. Training also includes fully masked inputs, while deployment always uses one forward from masks.

Discrete flow matching provides the structure for this training recipe. A path from masks to complete blocks defines prediction tasks at different completion stages, and step-size conditioning specifies how much of the path a prediction should cover (Gat et al., 2024; Monsefi et al., 2026a). We adapt the step-aware path objective of FS-DFM to the drafter's self-generated partial states and combine it with direct target supervision. CAT weighting (An et al., 2026b) emphasizes positions following confident reference prefixes. Together, these choices train the predictor for consecutive agreement with the target under a one-pass inference budget.

The architecture makes this training recipe practical for video. Four decoder layers are initialized from the target, and a linear projector exposes its cached hidden states to the drafter. Reusing these features avoids a separate transformer prefill over the video. This complements approaches that reduce visual tokens or overlap draft and target execution (Ji et al., 2025; Kong et al., 2026). Sparrow also reuses target features but drafts autoregressively (Zhang et al., 2026). At inference, VFlash-FM predicts a full block in parallel; verification then uses either a sequence or a candidate tree constructed from those same predictions (Ringel & Romano, 2026).

Our results connect the training change to both useful progress and latency. On LLaVA-OneVision-7B, chain decoding commits 5.50 tokens per cycle compared with 4.76 for the non-flow DFlash variant. Speed improves across all nine tested verification configurations, and the method reaches $3.850\times$ macro decoding speedup. A separately trained Qwen2.5-VL-7B drafter reaches $2.541\times$ across the same five benchmarks. The existing multi-step configurations commit more tokens but deliver little additional speed, supporting the choice to improve the predictor during training while retaining one-pass deployment.

Our contributions are:

(i) **Partial-state training for one-pass drafting.** We train a video drafter on its own partially generated responses, including errors, while preserving one forward per block at deployment.

(ii) **A target-conditioned flow-training recipe.** We combine a step-aware discrete-flow objective, direct target supervision, and prefix-oriented CAT weighting, and formulate their connection to endpoint prediction.

(iii) **Consistent gains across targets and verification choices.** We evaluate two video language models on five benchmarks. On LLaVA, the recipe improves macro speedup by 6.8%–10.9% over the non-flow variant across all nine decoding configurations, with higher committed progress in every configuration.

2 Related work

Speculative decoding and parallel proposals. Speculative decoding verifies a cheap model's proposals with the target, preserving the target's distribution under the corresponding rejection rule (Leviathan et al., 2023; Chen et al., 2023). DFlash uses block diffusion for target-conditioned parallel drafting (Arriola et al., 2025; Chen et al., 2026); we retain its one-pass inference backbone. PARD-2 introduces CAT optimization (An et al., 2026b), weighting each token's loss by the target's confidence in the preceding reference tokens. This reduces later-token penalties after uncertain prefixes. Tree methods verify multiple candidate continuations in one target pass (Miao et al., 2024; Li et al., 2024c; Chen et al., 2024b). We use DDTree's construction from a block drafter's position-wise predictions (Ringel & Romano, 2026).

Video and multimodal acceleration. Video SpecVLM reduces draft prefill by pruning visual tokens (Ji et al., 2025); ParallelVLM overlaps draft and target prefill (Kong et al., 2026). Both use separate autoregressive drafters in their main setting and also study self-drafting over reduced visual contexts. Sparrow trains a single-layer autoregressive drafter on target hidden states (Zhang et al., 2026). SpecFLASH combines visual-token compression with semi-autoregressive multi-token drafting (Wang et al., 2025). VFlash-FM uses target features to draft a whole block in one pass. Throughout, SpecVLM denotes the video method, distinct from the image method of Huang et al. (2025).

Discrete flow matching. Discrete flow matching models token sequences through continuous-time jump processes (Campbell et al., 2024; Gat et al., 2024). FS-DFM conditions the denoiser on step size and develops finite-step objectives (Monsefi et al., 2026a); TS-DFM studies trajectory shaping (Monsefi et al., 2026b). We use FS-DFM's step-aware path loss, with a clipped coefficient, to train a target-conditioned video drafter on its own partial predictions. A frozen target supplies the reference continuations and token probabilities. Our focus is improved drafting under a fixed one-pass inference budget. Appendix E provides further context.

3 Method

VFlash-FM predicts a block of response tokens from the target's cached hidden states. During training, the same predictor sees both all-mask blocks and partial blocks generated by its own predictions. At deployment, it predicts from masks in one forward, followed by target verification. Algorithm 1 summarizes the procedure.

3.1 Target conditioning and accepted progress

Let V be a video, u an instruction, and r the committed response prefix. At the start of a decoding cycle, the frozen target selects an anchor token a. With $\|$ denoting concatenation, define

$$c = (V, u, r, a, H_\phi(V, u, r)), \tag{1}$$

$$p_i(v) = p_\phi(v \mid V, u, r \,\|\, a \,\|\, y_{<i}), \qquad y_i = \arg\max_{v \in \mathcal{V}} p_i(v). \tag{2}$$

Here H_ϕ contains target hidden states strictly before the anchor. The anchor enters the drafter as a token embedding; future tokens and their hidden states provide supervision only. The block contains the anchor and $L = 15$ draft positions, giving block size $B = L + 1 = 16$.

For a proposal $\hat{y}$, causal target verification accepts its longest matching prefix,

$$A(\hat{y}, y) = \sum_{k=1}^{L} \prod_{i=1}^{k} \mathbf{1}\{\hat{y}_i = y_i\}. \tag{3}$$

Algorithm 1 VFlash-FM: partial-state training and one-pass inference

Require: Frozen target p_ϕ; drafter q_θ; block length L.

Training: one microbatch of blocks with cached contexts c

Obtain greedy target continuations y and distributions p; compute CAT weights w.
Sample h and $t = jh$ from Equation 9; set $x \leftarrow \mathtt{M}^L$.
for $s = 0, \ldots, j-1$ **do** ▷ Construct states without gradients
    Predict $q_\theta(\cdot \mid x, c, sh, h)$; rank unrevealed positions by argmax confidence.
    Reveal the most confident predictions until $\lceil (s+1)hL \rceil$ positions are revealed.
    Retain all earlier reveals, including mistakes and selected mask tokens.
end for
$q \leftarrow q_\theta(\cdot \mid \mathrm{sg}(x), c, t, h)$ ▷ One forward with gradients
Compute $\mathcal{L}_{\mathrm{FM}} = \langle d \rangle_w + \langle \ell^{\mathrm{path}} \rangle_w$ from Equation 13.
Update θ; keep the target, shared embeddings, and output head frozen.

Inference: one decoding cycle with cached context c **and anchor** a

$q \leftarrow q_\theta(\cdot \mid \mathtt{M}^L, c, 0, 1)$ ▷ One drafter forward
Form a chain of argmax tokens, or construct a candidate tree from q.
Verify the candidates in one target forward using causal or ancestor-only attention.
Commit the longest target-matching prefix/path and its correction or bonus token.
Update caches along the accepted path; use the target token as the next anchor.

The target supplies a correction at the first disagreement, or a bonus token if all draft tokens match. Thus an ordinary cycle commits $M = A + 1$ tokens; a stop token or output cap can truncate the last cycle. Over evaluation contexts,

$$\mathbb{E}[A] = \sum_{k=1}^{L} \Pr(\hat{y}_{1:k} = y_{1:k}). \tag{4}$$

This identity explains the training goal: useful progress depends on consecutively correct prefixes, so improving isolated late positions is insufficient.

3.2 A TARGET-DERIVED ENDPOINT PREDICTOR

The drafter has four decoder layers with the target's geometry, initialized from the target's own weights. For LLaVA-OneVision-7B, we copy layers $\{1, 9, 17, 25\}$, numbered from zero, and use their hidden states as context features. A learned linear projector and RMS norm map the concatenated features to the target hidden width. Token embeddings and the output head are shared with the target and frozen. The copied decoder layers, projector, final normalization, and step conditioner are trained for drafting; Appendix A gives the architecture and initialization details.

Following DFlash, each draft layer takes queries from the block and keys and values from the projected target features and block tokens. Attention is bidirectional within the block and restricted to pre-anchor positions in the context. For a partial block $x_t \in \mathcal{V}^L$, the drafter predicts endpoint distributions

$$q_{\theta,i}(v \mid x_t, c, t, h) = \mathrm{softmax}_v \, f_{\theta,i}(x_t, c, t, h). \tag{5}$$

Here $f_{\theta,i}$ denotes the drafter's logits, computed with the target's full-vocabulary output head; θ denotes drafter parameters. The mask $\mathtt{M}$ is the tokenizer's padding token, distinct from the chat stop token. Each draft layer receives (t, h) embeddings through zero-initialized FiLM modulation, initially preserving the unconditioned backbone.

3.3 DISCRETE FLOW MATCHING ON A DRAFT BLOCK

A discrete path from masks to the target continuation defines supervision across completion stages. At each position,

$$\pi_{t,i}(v \mid y_i) = (1-t)\delta_{\mathtt{M}}(v) + t\delta_{y_i}(v), \qquad 0 \le t \le 1, \tag{6}$$

where δ_v is a point mass. One network predicts the endpoint posterior $p_{1|t}(v \mid x_t, c)$, conditioned on time t and step size h following FS-DFM (Monsefi et al., 2026a). Unlike reference-path samples, our training states include the drafter's potentially incorrect predictions (Section 3.4).

With the endpoint denoiser held fixed over $[t, t+h]$, the associated jump process has transition

$$\Pr(x_{t+h,i} = v \mid x_t, c) = r\delta_{x_{t,i}}(v) + (1-r)p_{1|t}(v \mid x_t, c), \quad r = \frac{1-t-h}{1-t}. \tag{7}$$

At $(t, h) = (0, 1)$, $r = 0$: this auxiliary process ends with the all-mask endpoint prediction, connecting it to one-pass drafting. Prediction quality depends on the learned network. Appendix B derives the kernel and its assumptions.

The path objective. We adapt FS-DFM's path objective (Monsefi et al., 2026a, Section 4.3), clipping its interval-averaged rate at 10. Suppressing conditioning arguments and writing $I_i = \mathbf{1}\{x_{t,i} = y_i\}$,

$$\ell_i^{\text{path}} = -\bar{g}(t,h)\left[q_{\theta,i}(x_{t,i}) - I_i + (1-I_i)\log q_{\theta,i}(y_i)\right]. \tag{8}$$

For a masked or incorrectly revealed token, the loss directs probability toward the target token with a bounded term for the current token. For a correctly revealed token, it rewards retaining that token. The coefficient grows toward the end of an interval, emphasizing predictions that must complete the block.

3.4 Training on the drafter's own partial states

Training states are constructed by rolling out the drafter, so revealed tokens can be correct or incorrect. We sample

$$\begin{aligned} &h_k = 2^{-k}, \quad k = 0, \dots, 4, \qquad \Pr(h = h_k) = \omega_k/16, \quad \omega = (2, 1, 1, 10, 2), \\ &t = jh, \qquad j \sim \text{Uniform}\{0, \dots, 1/h - 1\}. \end{aligned} \tag{9}$$

Starting from masks, run j no-gradient drafter forwards. After forward s, retain the existing reveals and add the most confident remaining argmax predictions until $\lceil (s+1)hL \rceil$ positions have been revealed. Reference tokens never replace the drafter's predictions. One gradient-bearing forward then predicts the target continuation from this partial state, conditioned on (t, h). For example, $h = 1/4, t = 1/2$ uses two preparatory forwards; $h = 1, t = 0$ uses none.

Correct reveals provide information about neighboring tokens, while erroneous reveals require the predictor to recover the target continuation. The resulting states provide additional supervision for the same weights used at deployment. The schedule assigns 25.8% of microbatches to rollout-free all-mask states and the exact deployment condition $(0, 1)$ in 12.5%. Model-generated errors and confidence-ordered reveals make this state distribution different from the reference marginal in Equation 6; the path loss is therefore used as a surrogate on these states. The experiments evaluate the resulting training recipe as a whole.

3.5 The training objective

Direct supervision combines cross-entropy toward the target's greedy token and KL divergence from its top-64-renormalized distribution,

$$d_i = -\log q_{\theta,i}(y_i) + D_{\text{KL}}\big(p_i^{[64]} \,\|\, q_{\theta,i}\big). \tag{10}$$

The drafter retains a full-vocabulary normalizer. Confidence-Adaptive Token (CAT) optimization (An et al., 2026b) weights each position by the target's confidence in the preceding reference tokens:

$$w_i = m_i \prod_{r=1}^{i-1} p_r(y_r), \qquad \langle z \rangle_w = \frac{\sum_{b,i} w_{b,i} z_{b,i}}{\max(\sum_{b,i} w_{b,i}, 10^{-6})}. \tag{11}$$

Here m_i selects valid response positions, excludes the anchor, and b indexes blocks. CAT uses the target's full-vocabulary probabilities. Low-confidence reference prefixes downweight later positions, focusing supervision on tokens that can contribute to an accepted prefix.

Both loss terms use the same gradient-bearing forward:

$$\bar{g}(t,h) = \min\left\{10, \frac{1}{h}\log\frac{\max(1-t,\epsilon)}{\max(1-t-h,\epsilon)}\right\}, \qquad \epsilon = 10^{-9}, \tag{12}$$

$$\mathcal{L}_{\mathrm{FM}} = \langle d \rangle_w + \langle \ell^{\mathrm{path}} \rangle_w. \tag{13}$$

Clipping keeps the coefficient finite when $t + h = 1$. At the deployment condition, the per-position loss is $11\,\mathrm{CE}_i + \mathrm{KL}_i - 10 q_{\theta,i}(\mathtt{M})$, so the recipe changes the relative strength of hard-label supervision as well as the training states. Appendix B gives the full decomposition and gradients.

How the loss uses a partial prediction. The path objective distinguishes three prediction tasks. At a masked position, it learns the target continuation from the video context and any revealed neighbors. At an incorrectly revealed position, it learns to correct a token proposed by the drafter itself. At a correctly revealed position, the path term becomes $\bar{g}(1 - q_{\theta,i}(y_i))$, rewarding consistency with the correct token. Direct supervision remains active in all three cases, so the training target is always the target model's continuation.

This distinction also appears in the gradients. For a current token $x \neq y$, the path loss is $-\bar{g}(\log q_y + q_x)$ and its gradient with respect to the current-token logit is $\bar{g} q_x^2$, compared with $\bar{g} q_x$ for cross-entropy alone. The bounded current-token term therefore changes how an existing prediction is revised, while the target-token term directs the prediction toward the reference continuation. Time and step conditioning let the same network apply this objective across completion stages. At deployment, all positions are masked; the more strongly weighted hard-label term then favors agreement with the target's greedy choice. Appendix B gives the remaining gradients and the per-example optimum.

Why partial-state training can help one-pass prediction. The deployment input is fully masked, but the parameters that process it are also trained on every partial state. These tasks use the same video features, block attention, and output head to recover the same target continuation. Correct reveals provide additional context; incorrect reveals require recovery from the drafter's own errors. This supplies varied conditional prediction tasks for the shared network, while explicit all-mask examples retain direct supervision of the deployment task.

The flow objective also acts at the all-mask state itself. As Equation 13 shows, it increases the relative weight of the greedy target token, while CAT emphasizes positions whose preceding reference tokens are confident. The recipe therefore combines intermediate-state supervision with a deployment-state objective oriented toward consecutive target agreement. This is the motivation for using flow matching under a one-pass budget: the path structures additional training tasks without requiring traversal of that path at inference. The experiments test the combined recipe.

3.6 One-pass deployment and verification

At deployment, set $x_0 = \mathtt{M}^L$, $t = 0$, and $h = 1$. A single drafter forward gives $\hat{y}_i = \arg\max_v q_{\theta,i}(v \mid \mathtt{M}^L, c, 0, 1)$. The drafter caches projected context features and appends newly committed positions after each cycle. There are no preparatory rollouts or refinement passes. Chain verification follows Equation 3.

The conditioning boundary. The same boundary applies to every training block, even when multiple anchors share a cached response. A block can read target features only from positions preceding its own anchor. Target features computed on the reference continuation are excluded from its conditioning. Partial states instead contain the drafter's predictions at those future positions. At inference, only the verified prefix contributes new context features; rejected proposals never enter the persistent context. The newly selected target token starts the next block as an embedding. This keeps the information available to the drafter consistent between training and deployment.

Tree verification. Our Tree variant builds a candidate tree from the same one-pass distributions using DDTree (Ringel & Romano, 2026). Nodes are added best-first by summed log-probabilities along each path, up to a budget of 60 draft nodes and depth L. The target verifies the nodes in one pass under ancestor-only attention and commits the longest matching path and its correction or bonus token. We call the default variant Tree and use Tree-N when comparing different node budgets.

To reduce verification latency, attention over the shared cached context uses a dense mask-free kernel, while a sparse kernel handles each candidate's own tree path, including itself. Combining the

two outputs through their log-sum-exp normalizers preserves the softmax over permitted keys without constructing a dense tree mask over the long video context. Appendix C gives the construction and cache updates.

With an identical deterministic target decision rule and correct causal verification, accepting only matching prefixes preserves greedy target generation by induction. This statement assumes consistent numerical decisions, attention masks, and cache operations. Our experiments use greedy decoding; the factorized draft readout does not establish a validated stochastic sampler.

4 EXPERIMENTS

Target, data, and training. We evaluate LLaVA-OneVision-7B (Li et al., 2025a) and Qwen2.5-VL-7B (Bai et al., 2025), each with its own four-layer drafter trained for five epochs using the recipe of Section 3.4. Both use the target's greedy captions from LLaVA-Video-178K (Zhang et al., 2025) as supervision. The LLaVA run uses the 176,494-clip caption manifest; the Qwen run uses its leakage-pruned 175,864-clip version. Each drafter follows its target's decoder geometry and is evaluated under the same protocol (Appendix A).

Benchmarks and protocol. Five video benchmarks are used: VideoDetailCaption, MVBench, LongVideoBench, Video-MME, and MLVU (LMMs-Lab, 2024; Li et al., 2024a; Wu et al., 2024; Fu et al., 2025; Zhou et al., 2025). Every clip is prompted with one fixed detailed-description instruction of the kind used to generate the training captions ("Please provide a detailed description of the video, focusing on the main subjects, their actions, and the background scenes"), so the timed workload is long-form caption decoding on the videos of these benchmarks. Each benchmark uses 50 clips for each of three locked seeds (150 clip evaluations). Before timing, each evaluation process decodes its first two clips once with every method, the autoregressive baseline included, and discards those runs: this warm-up absorbs one-time GPU costs (CUDA context creation, kernel autotuning, memory-allocator growth), so that the baseline and the speculative decoder are both timed warm and compared fairly. Decoding uses native grouped-query attention for target and drafter, preallocated key-value caches, lower-right causal verification, block size 16, one drafter forward per block, greedy decoding, a 512-token output cap, and an autoregressive baseline of the target run immediately before the speculative arm on the same clip, on one NVIDIA H100 80GB.

Measurement. Decode timing excludes multimodal prefill and starts after the first target token, with CUDA synchronization at the timing boundaries and no profiler in the timed pass. Per clip, the decoding speedup is autoregressive divided by speculative seconds per output token; the benchmark value is the arithmetic mean of the 150 per-clip ratios and the macro value the mean of the five benchmark values, not a ratio of pooled times. The reported M is the mean over clips of the per-clip mean committed tokens per cycle, including the target's correction or bonus token.

Results. Table 1 reports the chain and tree results and published baselines. FM chain decoding reaches a five-benchmark macro speedup of $3.773\times$ and 5.50 committed tokens per cycle. Tree is the fastest tested FM tree overall, at $3.850\times$ and 7.23 committed tokens per cycle.

Figure 1 compares the benchmark-level speedups of the fastest listed variant of each method on LLaVA-OneVision-7B (0.5B drafters for SpecVLM and ParallelVLM, SPARROW, and FM Tree), one configuration per method across benchmarks; on the four benchmarks shared with SPARROW, FM Tree averages $3.79\times$ against SPARROW's $2.82\times$. On Qwen2.5-VL-7B, with the same one-step, $B = 16$ deployment and a separately trained drafter, FM chain decoding reaches $2.323\times$ macro with 3.17 committed tokens per cycle, and Tree reaches $2.541\times$ with 4.55; both exceed the published self-drafting and learned-drafter results on every benchmark.

Block size and flow steps at inference. Appendix D.2 varies block size and refinement steps for the same checkpoint. A larger block with two steps raises chain progress from 5.50 to 6.65 tokens per cycle, but changes speedup only from $3.773\times$ to $3.831\times$; tree speedup changes from $3.850\times$ to $3.828\times$. Thus, the extra accepted tokens yield little net speed benefit in these configurations. This supports the one-pass design: improving predictions during training can increase useful progress without paying for another refinement pass at deployment. The main results use one step and $B = 16$, the training block size.

Table 1: Video decoding results for two 7B targets. S is decoding speedup over each study's autoregressive baseline; τ is its reported token-length statistic. Bold indicates the highest listed speedup for each target and benchmark.

	VDC		V-MME		MVBench		MLVU		LVB		Avg.	
Method / variant	S	τ	S	τ	S	τ	S	τ	S	τ	S	τ
LLaVA-OneVision-7B												
SpecVLM (0.5B)	1.83	4.89	1.82	4.83	1.77	4.76	1.76	4.71	1.86	4.92	1.81	4.82
SpecVLM (7B self)	1.26	3.98	–	–	1.33	4.06	1.24	3.70	1.30	3.84	1.28	3.90
ParallelVLM (0.5B)	2.12	8.47	2.06	7.86	2.15	8.64	2.14	8.24	2.10	8.35	2.11	8.31
ParallelVLM (7B self)	1.53	14.80	1.55	15.60	1.49	13.20	1.59	15.70	1.57	15.00	1.55	14.86
SPARROW (learned)	2.84	3.88	2.96	3.98	2.69	3.67	–	–	2.77	3.78	2.82	3.83
VFlash-FM (Chain)	3.530	5.19	4.045	6.02	3.460	5.03	4.030	5.90	**3.803**	5.36	3.773	5.50
VFlash-FM (Tree)	**3.644**	6.85	**4.176**	7.89	**3.606**	6.74	**4.087**	7.65	3.734	7.04	**3.850**	7.23
Qwen2.5-VL-7B												
SpecVLM (7B self)	1.50	3.83	–	–	1.37	3.92	–	–	1.59	3.84	1.49	3.86
ParallelVLM (7B self)	1.52	14.40	1.53	14.90	1.47	14.80	1.50	14.60	1.53	15.30	1.51	14.80
SPARROW (learned)	2.04	3.89	–	–	2.06	3.87	–	–	2.02	3.78	2.04	3.85
VFlash-FM (Chain)	2.448	3.36	2.327	3.18	2.224	3.03	2.294	3.12	2.325	3.16	2.323	3.17
VFlash-FM (Tree)	**2.677**	4.80	**2.549**	4.58	**2.453**	4.40	**2.497**	4.48	**2.529**	4.52	**2.541**	4.55

VDC: VideoDetailCaption; V-MME: Video-MME; LVB: LongVideoBench. Sources: ParallelVLM and 0.5B SpecVLM from Kong et al. (2026), Table 1; self-drafting SpecVLM from Ji et al. (2025), Tables 1–2; SPARROW from Zhang et al. (2026), Tables 2–3, temperature zero. Averages cover the available benchmarks (dashes: unreported). For ours, $\tau = M$ includes the target's correction/bonus token; source length conventions are not harmonized.

Effect of flow-based training. Table 2 compares VFlash-FM with a DFlash variant trained without flow matching, denoted DFlash+CAT in the appendix. Both use four layers initialized from the same target layers, a feature projector, the shared output head, and CAT-weighted cross-entropy plus top-64 KL supervision. Both train for five epochs with the same learning rate, warmup, and anchor sampling. The baseline uses the leakage-pruned 175,864-clip manifest; the FM run uses 176,494 clips. Adding the step conditioner increases trainable parameters from 983.6M to 1,006.1M. The FM recipe also introduces partial-state rollouts and the path loss. This comparison evaluates these flow-training ingredients together. Both drafters decode the same evaluation clips with the same seeds and protocol.

With chain verification, committed tokens per cycle increase from 4.76 to 5.50 and macro speedup from $3.472\times$ to $3.773\times$, a gain of 8.7%. The best-tree comparison improves speedup by 10.1% and committed progress from 6.32 to 7.23. Each method uses one 60-node configuration selected by macro speedup and held fixed across benchmarks: sibling decay 0.9 for the baseline and no decay for VFlash-FM. This factor lowers the priority of lower-ranked sibling candidates during tree construction (Appendix D.1).

Consistency across benchmarks and verification settings. Table 3 breaks down the training comparison by benchmark. The gain appears on all five benchmarks with both chain verification and each method's best tree. Thus, the aggregate improvement reflects consistently higher committed progress and faster decoding across the evaluated video collections. Appendix D reports the complete configuration sweep.

Table 2: Ablation of flow training on LLaVA-OneVision-7B: DFlash variant, trained without flow matching, versus VFlash-FM, averaged over the five benchmarks. M counts committed tokens per cycle, including the target token. Best tree uses each method's highest-macro Tree configuration, held fixed across benchmarks.

	DFlash variant		VFlash-FM		Relative
Decoding	Speedup	M	Speedup	M	speedup gain
Chain	$3.472\times$	4.76	$\mathbf{3.773}\times$	**5.50**	$+8.7\%$
Best tree	$3.497\times$	6.32	$\mathbf{3.850}\times$	**7.23**	$+10.1\%$

Table 3: Flow training versus DFlash variant on each benchmark, LLaVA-OneVision-7B. S is decoding speedup; M is committed tokens per cycle including the target token. Best tree uses 60 draft nodes, with sibling decay 0.9 for the DFlash variant and no decay for VFlash-FM. Shaded rows are VFlash-FM, in bold: every entry is higher than the DFlash variant entry for the same decoding.

	VDC		V-MME		MVBench		MLVU		LVB		Avg.	
Drafter, decoding	S	M	S	M	S	M	S	M	S	M	S	M
DFlash variant, Chain	3.348	4.50	3.704	5.19	3.308	4.44	3.694	5.09	3.306	4.59	3.472	4.76
VFlash-FM, Chain	**3.530**	**5.19**	**4.045**	**6.02**	**3.460**	**5.03**	**4.030**	**5.90**	**3.803**	**5.36**	**3.773**	**5.50**
DFlash variant, best tree	3.415	6.08	3.713	6.86	3.460	6.01	3.595	6.64	3.301	5.99	3.497	6.32
VFlash-FM, best tree	**3.644**	**6.85**	**4.176**	**7.89**	**3.606**	**6.74**	**4.087**	**7.65**	**3.734**	**7.04**	**3.850**	**7.23**

Implications for one-pass drafting. The chain comparison shows that the training gain does not require tree expansion. The fixed-tree comparisons show that it persists when the candidate budget and construction rule are held constant. Together, these results locate the benefit of the combined training recipe in more useful draft predictions across verification choices. The block-size and refinement study addresses the complementary deployment decision: accepting more tokens is valuable only when the extra proposal and verification work can be amortized. For the long-context setting studied here, improving the shared predictor during training offers a way to obtain that progress while retaining one-pass deployment.

5 Conclusion

VFlash-FM improves a video drafter's one-pass predictions by expanding its training tasks to self-generated partial responses. A discrete path from masked blocks to the target continuation defines prediction tasks at every stage of block completion, and the drafter's own rollouts, with their correct and incorrect reveals alike, populate those tasks; step-aware flow supervision and target-confidence weighting then train the shared predictor to recover the target continuation from each of these states. Deployment is unchanged: one drafter forward over cached target features, followed by chain or tree verification. Across five video benchmarks, the method reaches macro decoding speedups of $3.850\times$ on LLaVA-OneVision-7B and $2.541\times$ on Qwen2.5-VL-7B. On LLaVA, relative to the non-flow variant, chain verification commits 5.50 tokens per cycle instead of 4.76, and speed gains of 6.8%–10.9% accompany higher committed progress across all nine tested verification configurations. Spending more computation at inference instead, through refinement steps or wider trees, commits more tokens but returns little additional speed, whereas the training-time improvement carries no deployment cost at all. Together, the results demonstrate a practical use of additional training computation to improve drafting under a fixed one-pass inference budget.

Limitations. Partial-state construction adds an average of 3.25 no-gradient drafter forwards per gradient-bearing forward during training. The method requires access to the target's weights and hidden states, and each target needs its own drafter training. Copying target layers retains their width, and the drafter's context cache grows with video length. Additional implementation details appear in Appendix A.

AI USE STATEMENT

A large language model was used to polish writing and improve clarity.

REPRODUCIBILITY STATEMENT

We will release a public Git repository upon acceptance containing the training and inference code, experiment configurations, and evaluation scripts. Section 3 specifies the method, Section 4 the evaluation protocol, and Appendices A and F the implementation and run details.

REFERENCES

Zihao An, Huajun Bai, Ziqiong Liu, Dong Li, and Emad Barsoum. PARD: Accelerating LLM Inference with Low-Cost PARallel Draft Model Adaptation. In *The Fourteenth International Conference on Learning Representations*, 2026a. doi: 10.48550/arXiv.2504.18583. URL https://arxiv.org/abs/2504.18583.

Zihao An, Taichi Liu, Ziqiong Liu, Dong Li, Ruofeng Liu, and Emad Barsoum. PARD-2: Target-Aligned Parallel Draft Model for Dual-Mode Speculative Decoding, 2026b. URL https://arxiv.org/abs/2605.08632.

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block Diffusion: Interpolating Between Autoregressive and Diffusion Language Models. In *The Thirteenth International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2503.09573. URL https://arxiv.org/abs/2503.09573.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured Denoising Diffusion Models in Discrete State-Spaces. In *Advances in Neural Information Processing Systems*, 2021. doi: 10.48550/arXiv.2107.03006. URL https://arxiv.org/abs/2107.03006.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL Technical Report, 2025. URL https://arxiv.org/abs/2502.13923.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM Inference Acceleration Framework with Multiple Decoding Heads. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. doi: 10.48550/arXiv.2401.10774. URL https://arxiv.org/abs/2401.10774.

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative Flows on Discrete State-Spaces: Enabling Multimodal Flows with Applications to Protein Co-Design. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. doi: 10.48550/arXiv.2402.04997. URL https://arxiv.org/abs/2402.04997.

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 *Proceedings of the 43rd International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2026. doi: 10.48550/arXiv.2602.06036. URL https://arxiv.org/abs/2602.06036.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An Image is Worth 1/2 Tokens After Layer 2: Plug-and-Play Inference Acceleration for Large Vision-Language Models. In *European Conference on Computer Vision (ECCV)*, 2024a. doi: 10.48550/arXiv.2403.06764. URL https://arxiv.org/abs/2403.06764.

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable and Robust Speculative Decoding. In *Advances in Neural Information Processing Systems*, 2024b. doi: 10.48550/arXiv.2402.12374. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/ea1f5f0878d43ff4fb8bf64ef4a2326c-Abstract-Conference.html.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One Step Diffusion via Shortcut Models. In *The Thirteenth International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2410.12557. URL https://arxiv.org/abs/2410.12557.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, Peixian Chen, Yanwei Li, Shaohui Lin, Sirui Zhao, Ke Li, Tong Xu, Xiawu Zheng, Enhong Chen, Caifeng Shan, Ran He, and Xing Sun. Video-MME: The First-Ever Comprehensive Evaluation Benchmark of Multi-modal LLMs in Video Analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24108–24118, 2025. URL https://arxiv.org/abs/2405.21075.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the Sequential Dependency of LLM Inference Using Lookahead Decoding. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. doi: 10.48550/arXiv.2402.02057. URL https://arxiv.org/abs/2402.02057.

Mukul Gagrani, Raghavv Goel, Wonseok Jeon, Junyoung Park, Mingu Lee, and Christopher Lott. On Speculative Decoding for Multimodal Large Language Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2024. doi: 10.48550/arXiv.2404.08856. URL https://arxiv.org/abs/2404.08856.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete Flow Matching. In *Advances in Neural Information Processing Systems*, 2024. doi: 10.48550/arXiv.2407.15595. URL https://arxiv.org/abs/2407.15595.

Haiduo Huang, Fuwei Yang, Zhenhua Liu, Xuanwu Yin, Dong Li, Pengju Ren, and Emad Barsoum. SpecVLM: Fast Speculative Decoding in Vision-Language Models, 2025. URL https://arxiv.org/abs/2509.11815.

Yicheng Ji, Jun Zhang, Heming Xia, Jinpeng Chen, Lidan Shou, Gang Chen, and Huan Li. SpecVLM: Enhancing Speculative Decoding of Video LLMs via Verifier-Guided Token Pruning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 7216–7230, 2025. doi: 10.18653/v1/2025.emnlp-main.366. URL https://aclanthology.org/2025.emnlp-main.366/.

Quan Kong, Yuhao Shen, Yicheng Ji, Huan Li, and Cong Wang. ParallelVLM: Lossless Video-LLM Acceleration with Visual Alignment Aware Parallel Speculative Decoding, 2026. URL https://arxiv.org/abs/2603.19610.

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.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. LLaVA-OneVision: Easy Visual Task Transfer. *Transactions on Machine Learning Research*, 2025a. doi: 10.48550/arXiv.2408.03326. URL https://arxiv.org/abs/2408.03326.

Guanghao Li, Zhihui Fu, Min Fang, Qibin Zhao, Ming Tang, Chun Yuan, and Jun Wang. DiffuSpec: Unlocking Diffusion Language Models for Speculative Decoding, 2025b. URL https://arxiv.org/abs/2510.02358.

Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, Limin Wang, and Yu Qiao. MVBench: A Comprehensive Multi-modal Video Understanding Benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22195–22206, 2024a. URL https://arxiv.org/abs/2311.17005.

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*, 2024b. doi: 10.48550/arXiv.2401.15077. URL https://arxiv.org/abs/2401.15077.

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, 2024c. doi: 10.18653/v1/2024.emnlp-main.422. URL https://aclanthology.org/2024.emnlp-main.422/.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up Inference Acceleration of Large Language Models via Training-Time Test. In *Advances in Neural Information Processing Systems*, 2025c. doi: 10.48550/arXiv.2503.01840. URL https://arxiv.org/abs/2503.01840.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling. In *The Eleventh International Conference on Learning Representations*, 2023. doi: 10.48550/arXiv.2210.02747. URL https://arxiv.org/abs/2210.02747.

Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky T. Q. Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow Matching Guide and Code, 2024. URL https://arxiv.org/abs/2412.06264.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual Instruction Tuning. In *Advances in Neural Information Processing Systems*, 2023. doi: 10.48550/arXiv.2304.08485. URL https://arxiv.org/abs/2304.08485.

LMMs-Lab. VideoDetailCaption. Hugging Face dataset, 2024. URL https://huggingface.co/datasets/lmms-lab/VideoDetailCaption.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. SpecInfer: Accelerating Large Language Model Serving with Tree-based Speculative Inference and Verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pp. 932–949, 2024. doi: 10.1145/3620666.3651335. URL https://arxiv.org/abs/2305.09781.

Amin Karimi Monsefi, Nikhil Bhendawade, Manuel Rafael Ciosici, Dominic Culver, Yizhe Zhang, and Irina Belousova. FS-DFM: Fast and Accurate Long Text Generation with Few-Step Diffusion Language Models. In *The Fourteenth International Conference on Learning Representations*, 2026a. doi: 10.48550/arXiv.2509.20624. URL https://arxiv.org/abs/2509.20624.

Amin Karimi Monsefi, Dominic Culver, Nikhil Bhendawade, Manuel R. Ciosici, Yizhe Zhang, and Irina Belousova. Trajectory as the Teacher: Few-Step Discrete Flow Matching via Energy-Navigated Distillation, 2026b. URL https://arxiv.org/abs/2605.07924.

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. In *Advances in Neural Information Processing Systems*, 2025. doi: 10.48550/arXiv.2502.09992. URL https://arxiv.org/abs/2502.09992.

Liran Ringel and Yaniv Romano. Accelerating Speculative Decoding with Block Diffusion Draft Trees, 2026. URL https://arxiv.org/abs/2604.12989.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and Effective Masked Diffusion Language Models. In *Advances in Neural Information Processing Systems*, 2024. doi: 10.48550/arXiv.2406.07524. URL https://arxiv.org/abs/2406.07524.

Jameson Sandler, Jacob K. Christopher, Thomas Hartvigsen, and Ferdinando Fioretto. SpecDiff-2: Scaling Diffusion Drafter Alignment For Faster Speculative Decoding. In *Conference on Machine Learning and Systems (MLSys)*, 2026. doi: 10.48550/arXiv.2511.00606. URL https://arxiv.org/abs/2511.00606.

Neta Shaul, Itai Gat, Marton Havasi, Daniel Severo, Anuroop Sriram, Peter Holderrieth, Brian Karrer, Yaron Lipman, and Ricky T. Q. Chen. Flow Matching with General Discrete Paths: A Kinetic-Optimal Perspective. In *The Thirteenth International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2412.03487. URL https://arxiv.org/abs/2412.03487.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency Models. In *Proceedings of the 40th International Conference on Machine Learning*, 2023. doi: 10.48550/arXiv.2303.01469. URL https://arxiv.org/abs/2303.01469.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise Parallel Decoding for Deep Autoregressive Models. In *Advances in Neural Information Processing Systems*, 2018. doi: 10.48550/arXiv.1811.03115. URL https://arxiv.org/abs/1811.03115.

Zihua Wang, Ruibo Li, Haozhe Du, Joey Tianyi Zhou, Yu Zhang, and Xu Yang. SpecFLASH: A Latent-Guided Semi-autoregressive Speculative Decoding Framework for Efficient Multimodal Generation, 2025. URL https://arxiv.org/abs/2505.12728.

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. Fast-dLLM v2: Efficient Block-Diffusion LLM. In *The Fourteenth International Conference on Learning Representations*, 2026. doi: 10.48550/arXiv.2509.26328. URL https://arxiv.org/abs/2509.26328.

Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. LongVideoBench: A Benchmark for Long-context Interleaved Video-Language Understanding. In *Advances in Neural Information Processing Systems*, volume 37, pp. 28828–28857, 2024. URL https://arxiv.org/abs/2407.15754.

Libo Zhang, Zhaoning Zhang, Wangyang Hong, Peng Qiao, and Dongsheng Li. Sparrow: Text-Anchored Window Attention with Visual-Semantic Glimpsing for Speculative Decoding in Video LLMs. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9928–9942, San Diego, California, USA, 2026. doi: 10.18653/v1/2026.acl-long.450. URL https://aclanthology.org/2026.acl-long.450/.

Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. LLaVA-Video: Video Instruction Tuning With Synthetic Data. *Transactions on Machine Learning Research*, 2025. doi: 10.48550/arXiv.2410.02713. URL https://arxiv.org/abs/2410.02713.

Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Zhengyang Liang, Shitao Xiao, Minghao Qin, Xi Yang, Yongping Xiong, Bo Zhang, Tiejun Huang, and Zheng Liu. MLVU: Benchmarking Multi-task Long Video Understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13691–13701, 2025. URL https://arxiv.org/abs/2406.04264.

Appendix

Appendix Contents

A Implementation record **15**
A.1 Architecture and initialization 15
A.2 Training-state construction and loss normalization 16
A.3 Positioning and cache updates 16

B Derivations and loss analysis **16**
B.1 The frozen-denoiser transition kernel 16
B.2 The path loss and its gradients 17
B.3 Decomposition and the all-mask optimum 18
B.4 Reveal quotas and sampled training states 18

C Tree-60 verification **19**

D Ablation studies **19**
D.1 Decoding configurations 19
D.2 Block size and flow steps at inference 20

E Additional related work **21**

F Run provenance **22**

A IMPLEMENTATION RECORD

This appendix records the training and evaluation configuration.

Setting	Value
Target	LLaVA-OneVision-Qwen2-7B (`llava-hf/llava-onevision-qwen2-7b-ov-hf`)
Target geometry	28 layers, hidden 3584, 28 Q / 4 KV heads, vocab 152128
Draft layers / block size	4 / 16 (anchor + 15 proposals)
Warm-start source layers	target layers $\{1, 9, 17, 25\}$; all 932.2M layer parameters copied
New parameters	projector 51.4M, FM conditioner 22.5M (zero-init FiLM), final norm
Trainable parameters	1006.1M (target embedding and output head frozen, shared)
Inference FM steps	1
State construction / source	Rollout / mask
Finest step size	$1/16$
Step-size weights ω	$[2, 1, 1, 10, 2]$ over $h \in \{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}\}$
Mask ID	151643 (<|endoftext|>, tokenizer pad token)
Path / direct loss weights	1 / 1; $\bar{g}$ clip 10, $\epsilon = 10^{-9}$
CE / KL weights, KL support, temperature	1 / 1, top 64, 1
CAT weighting	On (full-vocabulary target confidence)
Attention	SDPA; context positions $<$ anchor; bidirectional within block
Training clips / held-out clips	176,494 / 2,000 (32 used per periodic eval)
Frames / visual tokens / sequence limit	128 / 25,089 per clip / 32768
Anchors per clip / microbatch	up to 256 / at most 64 (one (t, h) draw per microbatch)
Batch	1 clip per rank per step
Epochs	5
Optimizer	AdamW, lr 3×10^{-4}, 2% linear warmup, cosine decay to 0
Regularization / precision	grad-norm clip 1.0, weight decay 0.01 (default), bf16 drafter, seed 0
Evaluation decode	greedy, up to 512 new tokens

A.1 ARCHITECTURE AND INITIALIZATION

The drafter uses $D = 4$ decoder layers with the same geometry as the target, initialized by copying the target's own weights. For LLaVA-OneVision-7B, the target is a 28-layer Qwen2-7B decoder with hidden size 3584, 28 query heads, and four key-value heads. We copy layers $\{1, 9, 17, 25\}$, numbered from zero. These are also the layers whose hidden states condition the drafter, selected using DFlash's rule of spacing D feature taps evenly from layer 1 to layer $n - 3$ of an n-layer target. Three components are new and start from scratch: a projector (a bias-free linear map from the concatenated hidden states of the same D target layers to the hidden size, followed by an RMS norm), the step conditioner described below, and the final RMS norm. The drafter has $1{,}006.1$M trainable parameters (Appendix A); the target's token embedding and output head are shared with the drafter and frozen.

Both scalars are embedded by 256-dimensional sinusoidal features (128 frequencies, cosine and sine) followed by a two-layer MLP; the two embeddings are concatenated, projected to a 256-dimensional code e, and passed through a SiLU. Each draft layer applies FiLM modulation from e to its attention and MLP branches: with per-branch shift, scale, and gate vectors (β, γ, α), each a linear function of e, a branch F with input x computes $x + (1 + \alpha) \odot F\big((1 + \gamma) \odot \mathrm{norm}(x) + \beta\big)$. All FiLM projections are zero-initialized, so at initialization the conditioning is an exact identity and the drafter coincides with a plain DFlash-architecture drafter.

Mask and output vocabulary. The mask token is the tokenizer padding ID, which has a row in the shared embedding and output head and is distinct from the chat stop token. The drafter uses a full-vocabulary softmax; the mask probability is not suppressed during proposal generation. The geometry and parameter counts in the table above describe the LLaVA drafter. The Qwen drafter follows the same design using the Qwen target's geometry.

A.2 Training-state construction and loss normalization

Training processes one clip per rank and step, with up to 256 distinct anchor positions drawn from its response (fewer for the 27% of clips whose response is shorter than 271 tokens), split into microbatches of at most 64 blocks. One (t, h) pair is drawn per microbatch and shared by its blocks. Starting with masks, execute $j = t/h$ no-gradient drafter evaluations; the s-th evaluation ($s = 0, \ldots, j-1$) is conditioned on time sh and step size h. After it, retain previously revealed tokens and reveal the most confident predictions among the remaining positions until the reveal count reaches $\lceil (s+1)h\,n \rceil$, where n is the number of valid response positions in the block; with the anchor sampler used here, which places anchors at least B positions before the end of the sequence, $n = L = 15$ for every block. The revealed tokens are the drafter's own argmax predictions, not substituted reference tokens, and they are never edited later. Denote the resulting state by $x_t = R_{\mathrm{sg}(\theta)}(c; t, h)$, where sg means that gradients do not pass through the rollout. The same drafter then performs one gradient-bearing forward on x_t, conditioned on (t, h).

The expected number of preparatory forwards per microbatch is 3.25 under Equation 9. Rollout-free all-mask states account for 25.8% of microbatches, and the exact deployment condition for 12.5%. Each microbatch has its own CAT-weighted loss normalizer. The clip loss is the anchor-count-weighted mean of its microbatch losses, and gradients are averaged across ranks. CAT uses full-vocabulary target probabilities; the KL term uses the target's top-64-renormalized distribution and the drafter's full-vocabulary normalizer. CAT weights are a target-confidence surrogate, rather than the measured probability that a draft prefix survives.

A.3 Positioning and cache updates

The anchor and proposed tokens occupy consecutive absolute sequence positions in the rotary embedding. Training blocks attend only to context positions strictly before their anchor. After verification, the draft suffix is discarded and the target cache is truncated through the last accepted draft token. Hidden states of newly committed positions through that token are appended to the drafter context. The target's correction or bonus token becomes the next anchor and is re-fed in the next verification pass. Generation stops when the chat stop token `<|im_end|>` is committed or the output cap is reached.

For multi-step inference with S forwards, the deterministic reveal procedure uses $h = 1/S$, retains revealed tokens between forwards, and returns the argmax of the final forward at every position. Its reveal quota rounds up the reference path's expected reveal count; it does not sample the transition distribution. With greedy target labels, the exact endpoint posterior is a point mass on the reference continuation at each context. Training on partial states supplies auxiliary prediction tasks for the shared network; improved all-mask accuracy is an empirical property of the learned predictor.

B Derivations and loss analysis

B.1 The frozen-denoiser transition kernel

Reference path and domain. Fix the context c and a target continuation y. The conditional reference path is $\pi_t(x \mid y) = \prod_i \pi_{t,i}(x_i \mid y_i)$, with $\pi_{t,i}$ defined in Equation 6. For $0 \le t < 1$, its posterior-parameterized coordinate rates are

$$u_t^i(v \mid x_t, c) = g(t)\big[p_{1|t}(v \mid x_t, c) - \delta_{x_{t,i}}(v)\big], \qquad g(t) = \frac{1}{1-t}. \tag{14}$$

For $v \neq x_{t,i}$ these are nonnegative jump rates; the diagonal term is minus their sum. An exact endpoint posterior is defined on the support of the reference path. On model-generated states outside that support, the learned predictor supplies an extension, and the training loss is used as the surrogate specified in Section 3.4.

Exact integration with a frozen prediction. Fix a coordinate and hold its probability vector p constant throughout $[t, t+h]$, including after any intermediate jumps. This defines an auxiliary

process with row-generator $Q(\tau) = g(\tau)(P - I)$, where $P = \mathbf{1}p^\top$. For $0 < h < 1 - t$,

$$G_{t,h} = \int_t^{t+h} g(\tau)\, d\tau = \log \frac{1-t}{1-t-h}, \qquad P^2 = \mathbf{1}(p^\top \mathbf{1})p^\top = P. \tag{15}$$

All generators are scalar multiples of the same matrix, so they commute. Since $I - P$ is also a projection,

$$K_{t,h} = \exp\big(G_{t,h}(P - I)\big) = P + e^{-G_{t,h}}(I - P) = rI + (1 - r)P, \qquad r = \frac{1-t-h}{1-t}. \tag{16}$$

Reading the row indexed by the current token gives Equation 7. Its entries are nonnegative and its rows sum to one. At $h = 1 - t$, the expression is understood as the limit $G_{t,h} \to \infty$: $r \to 0$ and every row tends to p. The kernel is exact for the auxiliary frozen-prediction process. It does not claim exact integration when the denoiser is recomputed after intermediate state changes. The clipped, numerically regularized coefficient used in the training loss is separate from this unclipped transition-kernel derivation.

Relation to a single-jump update. For a current token x and fixed p, the exit rate is $g(\tau)(1 - p_x)$. FS-DFM's single-jump update (Monsefi et al., 2026a, Appendix B.1) uses jump probability $1 - e^{-G_{t,h}(1-p_x)}$ and draws from p renormalized to exclude x. Its diagonal transition probability is therefore $e^{-G_{t,h}(1-p_x)}$, whereas the exact frozen-prediction kernel has diagonal $p_x + (1-p_x)e^{-G_{t,h}}$. Both equal $1 - G_{t,h}(1-p_x) + O(G_{t,h}^2)$, and their off-diagonal entries both equal $G_{t,h}p_v + O(G_{t,h}^2)$. They agree to first order, but differ for finite intervals because the exact kernel allows multiple jumps and returns to x. For fixed $p_x < 1$, the single-jump update leaves x in the terminal limit; if $p_x = 1$, the exit rate is zero and the position stays unchanged. Our greedy, confidence-ordered reveal procedure is distinct from either stochastic sampler.

What the tower identity establishes. For a stochastic endpoint Y and any fixed token v, the law of total expectation gives

$$\Pr(Y_i = v \mid c) = \mathbb{E}_{X_t \sim \pi_t(\cdot|c)}\big[\Pr(Y_i = v \mid X_t, c)\big]. \tag{17}$$

This identity concerns exact posteriors under the reference joint distribution. It does not bound a learned predictor's error or prove transfer from rollout states to all-mask inputs. For the deterministic greedy continuation used here, $Y = y$ given c, and the identity is immediate on the reference support.

B.2 The path loss and its gradients

Let $x = x_{t,i}$, $y = y_i$, $q = q_{\theta,i} = \mathrm{softmax}(z)$, and treat the clipped coefficient $\bar{g}$ as fixed with respect to the logits. To see the sign of the stay term, consider the off-diagonal reference and predicted rates

$$a_v = \bar{g}\,\mathbf{1}\{v = y\}, \qquad b_v = \bar{g}q_v, \qquad v \neq x. \tag{18}$$

Their generalized KL divergence is

$$\sum_{v \neq x}\left[a_v \log \frac{a_v}{b_v} - a_v + b_v\right] = \begin{cases} -\bar{g}(\log q_y + q_x), & x \neq y, \\ \bar{g}(1 - q_y), & x = y, \end{cases} \tag{19}$$

using $0 \log(0/b) = 0$. This is Equation 8; in particular, the loss is nonnegative despite the negative bounded term in the first case.

For $x \neq y$, differentiating with $\partial q_u / \partial z_v = q_u(\mathbf{1}\{u = v\} - q_v)$ yields

$$\frac{\partial \ell^{\mathrm{path}}}{\partial z_v} = \begin{cases} \bar{g}q_x^2, & v = x, \\ \bar{g}\,[q_y(1 + q_x) - 1], & v = y, \\ \bar{g}q_v(1 + q_x), & v \notin \{x, y\}. \end{cases} \tag{20}$$

The gradients sum to zero, as required by invariance to a common logit shift. Relative to $\bar{g}$-weighted cross-entropy, the current-token gradient is reduced from $\bar{g}q_x$ to $\bar{g}q_x^2$, while gradients on other wrong tokens are larger. Since $q_y \leq 1 - q_x$, the target-token gradient remains nonpositive.

For $x = y$, the complete gradient is

$$\frac{\partial \ell^{\text{path}}}{\partial z_v} = \begin{cases} -\bar{g} q_y (1 - q_y), & v = y, \\ \bar{g} q_y q_v, & v \neq y. \end{cases} \tag{21}$$

These gradients vanish as $q_y \to 0$. The direct loss remains active and supplies cross-entropy and soft-target supervision at that position. The formulas above are per-token path-loss gradients; CAT weighting and the fixed microbatch normalizer scale their contributions to the implemented objective.

B.3 Decomposition and the all-mask optimum

With $I_i = \mathbf{1}\{x_{t,i} = y_i\}$, $\mathrm{CE}_i = -\log q_{\theta,i}(y_i)$, and $\mathrm{KL}_i = D_{\mathrm{KL}}(p_i^{[64]} \| q_{\theta,i})$, the direct loss and path loss satisfy

$$d_i + \ell_i^{\text{path}} = [1 + \bar{g}(1 - I_i)]\,\mathrm{CE}_i + \mathrm{KL}_i + \bar{g}\,[I_i - q_{\theta,i}(x_{t,i})]. \tag{22}$$

At $(t, h) = (0, 1)$, the implemented coefficient is $\bar{g} = 10$. For $x_{t,i} = \mathtt{M} \neq y_i$, the loss is therefore $11\,\mathrm{CE}_i + \mathrm{KL}_i - 10 q_{\theta,i}(\mathtt{M})$.

A per-example optimum. Write $\mu = p_i^{[64]}$ and first assume $\mu_{\mathtt{M}} = 0$. For mask mass $m = q_{\mathtt{M}} < 1$, write $q_v = (1 - m) r_v$ on the non-mask vocabulary. Up to a constant independent of q, the objective becomes

$$-\sum_{v \neq \mathtt{M}} (11 \delta_{y_i}(v) + \mu_v) \log r_v - 12 \log(1 - m) - 10m. \tag{23}$$

The derivative of the last two terms is $12/(1-m) - 10 > 0$, so the minimum has $m = 0$. Minimizing the remaining cross-entropy gives

$$q^* = \frac{11 \delta_{y_i} + \mu}{12}, \qquad q^*_{\text{direct}} = \frac{\delta_{y_i} + \mu}{2}. \tag{24}$$

Thus the path term increases the per-example optimum's mass on the greedy token by $\frac{5}{12}(1 - \mu_{y_i}) \geq 0$. This is optimization over the closed probability simplex; zero probabilities are limiting values of a finite-logit softmax. It is not a statement that shared network parameters attain all per-example optima simultaneously.

If $\mu_{\mathtt{M}} > 0$, Equation 24 does not apply. The constrained optimum instead has

$$q_v^* = \frac{11 \delta_{y_i}(v) + \mu_v}{\lambda} \quad (v \neq \mathtt{M}), \qquad q_{\mathtt{M}}^* = \frac{\mu_{\mathtt{M}}}{\lambda - 10}, \qquad \lambda = 11 + \sqrt{1 + 10 \mu_{\mathtt{M}}}. \tag{25}$$

The multiplier is the root greater than 10 of the normalization equation. This case makes explicit why the zero-mask-mass assumption is needed for the simpler mixture formula.

Loss normalization. Both terms use the same CAT weights and normalizer at all valid response positions. Correctly revealed positions therefore remain in the objective and in its denominator. Their inclusion changes the relative weight assigned to the other positions; it does not by itself determine the direction of the total parameter gradient.

B.4 Reveal quotas and sampled training states

Assume $y_i \neq \mathtt{M}$ for each valid position. Under the reference path, the expected number of revealed positions at time t is Lt. Under its monotone mask-to-target coupling, a masked position reveals during $[t, t+h]$ with probability $h/(1-t)$, giving marginal expected progress $L(1-t)h/(1-t) = Lh$.

The implemented rollout instead selects exactly $K(t) = \lceil Lt \rceil$ positions at a grid time. Thus $0 \leq K(t) - Lt < 1$, and the step increment $K(t + h) - K(t)$ need not equal Lh. Selection follows the model's confidence and uses its own token predictions, so neither the positions nor their contents follow the reference distribution. With $L = 15$, $h = 1/16$, and $t = 15/16$, the quota is already $\lceil 225/16 \rceil = 15$: all positions have been selected before the final prediction. A selected argmax may itself equal $\mathtt{M}$; the implementation still marks that position as revealed and retains it.

The schedule's exact rollout statistics are

$$\mathbb{E}[j] = \sum_{k=0}^{4} \frac{\omega_k}{16} \frac{2^k - 1}{2} = 3.25, \qquad \Pr(j = 0) = \sum_{k=0}^{4} \frac{\omega_k}{16} 2^{-k} = \frac{33}{128}, \qquad \Pr(t = 0, h = 1) = \frac{1}{8}. \tag{26}$$

Accordingly, 25.8% of microbatches start from masks with no preparatory rollout, and 12.5% use the exact deployment condition. The first percentage describes $j = 0$ samples; it does not exclude a positive-length rollout that happens to predict only mask tokens.

C TREE-60 VERIFICATION

Tree-60 replaces the single chain by a draft tree built from the same one-pass distributions, following DDTree (Ringel & Romano, 2026). Each position's candidates are ranked by $q_{\theta,i}$, and nodes are added best-first by the sum of per-position log-probabilities along their root path (a product of parallel marginals, used as a score rather than as a calibrated joint probability): the tree starts from the top-1 token at depth 1, and each time a node is added, its next sibling (the next-ranked candidate at the same depth under the same parent) and its best child (the top-1 candidate at the next depth) enter the priority queue, until the budget of 60 draft nodes below the anchor is spent, with depth at most L. The target scores all nodes in one forward under an attention mask that lets each node see the context, the anchor, and its ancestors only, with a node's position equal to the anchor position plus its depth. The accepted length is the longest root path whose every token is the target's greedy choice at its parent, $A_{\mathcal{T}} = \max\{k : \ y_{1:k} \text{ is a root path of } \mathcal{T}\}$, so that $\mathbb{E}[A_{\mathcal{T}}] = \sum_k \Pr(y_{1:k} \in \mathcal{T})$; the target's choice at the last accepted node is committed as in the chain case, and only the accepted path enters the target and drafter caches. Tree construction adds a small per-cycle cost and the wider verification forward a larger one, which is why Tree-60's 31% more committed tokens per cycle in Table 1 translate into a 2% higher speedup.

D ABLATION STUDIES

This appendix explains the decoding configurations compared in Table 4 and examines block size and the number of flow steps at inference (Appendix D.2). The per-benchmark training comparison appears in Table 3.

D.1 DECODING CONFIGURATIONS

Table 4 compares the two LLaVA-OneVision-7B drafters using chains and trees with 32, 60, or 128 nodes. Each configuration is fixed across the five benchmarks. With sibling decay $\gamma \in (0, 1]$, a node of zero-based sibling rank r receives queue priority $s + r \log \gamma$, where s is its unpenalized path log-probability. Equivalently, its path-probability score is multiplied by γ^r. The penalty is local to the candidate: a child's priority uses the unpenalized parent path score. Decay delays lower-ranked siblings; it changes the chosen nodes, not the drafter, node budget, or verification rule. Setting $\gamma = 1$ recovers the plain tree.

Table 4: LLaVA-OneVision-7B: all nine configurations of the paired drafter comparison. S is macro decoding speedup, M is committed tokens per cycle including the target token, and Gain is VFlash-FM's relative speedup over the DFlash variant. Tree-N uses N draft nodes. Bold marks each drafter's fastest configuration.

	DFlash variant		VFlash-FM		
Decoding	S	M	S	M	Gain
Chain	3.472	4.76	3.773	5.50	+8.7%
Tree-32	3.331	5.98	3.693	6.91	+10.9%
Tree-32, decay 0.8	3.319	6.00	3.675	6.86	+10.7%
Tree-60	3.486	6.28	**3.850**	7.23	+10.4%
Tree-60, decay 0.7	3.485	6.32	3.826	7.20	+9.8%
Tree-60, decay 0.8	3.448	6.28	3.824	7.18	+10.9%
Tree-60, decay 0.9	**3.497**	6.32	3.831	7.20	+9.6%
Tree-128	3.371	6.68	3.602	7.61	+6.8%
Tree-128, decay 0.8	3.306	6.62	3.566	7.53	+7.8%

The improvement persists when tree construction is held fixed. With the same plain 60-node tree, speedup rises from 3.486× to 3.850×, a 10.4% gain. Across all nine decoding configurations, VFlash-FM improves macro speedup by 6.8%–10.9% and commits more tokens per cycle in every configuration and on every benchmark (Tables 3 and 4). The advantage therefore accompanies increased useful progress and persists across verification choices.

Accepted progress and verification cost. More accepted tokens do not automatically produce a proportional speedup. For the LLaVA drafter, switching from Chain to Tree increases committed progress from 5.50 to 7.23 tokens per cycle, while speedup increases from 3.773× to 3.850×. Constructing and verifying a wider tree consumes part of the benefit. A 128-node tree commits still more tokens, but is slower than the 60-node tree on LLaVA. Qwen exhibits a different tradeoff: its 128-node tree is the fastest tested configuration at 2.569×. The main comparison retains the 60-node choice selected on LLaVA for both targets. These results motivate evaluating proposals by accepted progress together with latency, rather than maximizing either block size or tree width alone.

Table 5: Qwen2.5-VL-7B: deployment configurations for VFlash-FM. These results describe the trained FM drafter's inference tradeoff; the paired training ablation is reported separately for LLaVA in Table 4. Bold marks the fastest Qwen configuration.

Decoding	Speedup	Committed tokens per cycle
Chain	2.323	3.17
Tree-32	2.403	4.29
Tree-32, decay 0.8	2.404	4.28
Tree-60	2.541	4.55
Tree-60, decay 0.7	2.513	4.49
Tree-60, decay 0.8	2.517	4.50
Tree-60, decay 0.9	2.532	4.53
Tree-128	**2.569**	4.82
Tree-128, decay 0.8	2.547	4.77

D.2 Block size and flow steps at inference

Table 6 varies the two inference-time settings of the same checkpoint, the block size B and the number of flow steps per block, under the protocol of Table 1. Block sizes $B \in \{8, 16, 24, 32\}$ were evaluated at $\{1, 2, 4\}$ flow steps: every one-step configuration was run on all five benchmarks, and the two- and four-step configurations were screened on VideoDetailCaption, whose two fastest cells, $B = 24$ and $B = 32$ at two steps, were carried to all five benchmarks. The one-step $B = 8$ rows were measured in a later batch that shared the machine with other jobs, in which the autoregressive baseline ran between 3% and 21% slower than in the main batch, depending on the benchmark (11% on average), so their speedups are not directly comparable with the other rows; their committed-

token counts do not depend on machine load. At $B = 8$ the tree trails the chain by $0.17\times$, the widest margin in the table, because the tree's construction and wider verification do not amortize over so short a block; at $B = 32$ with two steps the tree also trails the chain, by $0.003\times$. Of the rows in the table, the two-step $B = 32$ chain is the fastest on Video-MME and MLVU, and the one-step $B = 16$ rows on the other three benchmarks. The fastest chain macro is the two-step $B = 32$ configuration ($3.831\times$); the fastest tree macro is the one-step $B = 16$ configuration ($3.850\times$).

Table 6: Block size B and number of flow steps at inference on LLaVA-OneVision-7B, same checkpoint, hardware, and protocol as Table 1. S is decoding speedup; M is committed tokens per cycle including the target token. The $B = 16$, one-step rows repeat Table 1. Bold marks the highest speedup in each column. The $B = 8$ rows were measured under machine load (see text).

	VDC		V-MME		MVBench		MLVU		LVB		Avg.	
Configuration	S	M	S	M	S	M	S	M	S	M	S	M
B=8, 1 step, Chain	3.222	4.44	3.394	4.81	3.005	4.26	3.465	4.75	3.208	4.48	3.259	4.55
B=8, 1 step, Tree-60	3.081	5.73	3.229	6.01	3.014	5.65	3.200	5.93	2.935	5.53	3.092	5.77
B=16, 1 step, Chain	3.530	5.19	4.045	6.02	3.460	5.03	4.030	5.90	**3.803**	5.36	3.773	5.50
B=16, 1 step, Tree-60	**3.644**	6.85	4.176	7.89	**3.606**	6.74	4.087	7.65	3.734	7.04	**3.850**	7.23
B=24, 2 steps, Chain	3.584	6.23	4.049	7.07	3.407	6.00	4.075	7.03	3.720	6.34	3.767	6.54
B=24, 2 steps, Tree-60	3.580	7.85	4.102	9.01	3.566	7.77	4.091	9.00	3.711	8.17	3.810	8.36
B=32, 2 steps, Chain	3.626	6.10	**4.246**	7.48	3.359	5.91	**4.207**	7.25	3.716	6.51	3.831	6.65
B=32, 2 steps, Tree-60	3.630	8.00	4.187	9.38	3.465	7.72	4.137	9.14	3.722	8.25	3.828	8.50

E ADDITIONAL RELATED WORK

Drafters and verification. Blockwise parallel decoding drafts a block with extra prediction heads and keeps the longest prefix that the base model verifies (Stern et al., 2018), the scheme that one-pass block drafters, including ours, carry out with a learned drafter. Medusa drafts several future tokens from one forward with multiple decoding heads (Cai et al., 2024). EAGLE conditions a lightweight drafter on the target's hidden features (Li et al., 2024b), the conditioning route that DFlash and our drafter follow; EAGLE-2 builds dynamic draft trees from drafter confidences (Li et al., 2024c), and EAGLE-3 trains the drafter on its own multi-step predictions through training-time test (Li et al., 2025c). SpecInfer verifies a token tree in one target forward (Miao et al., 2024), and Sequoia chooses tree topologies for a given node budget (Chen et al., 2024b); our Tree-60 follows DDTree's construction (Ringel & Romano, 2026). Lookahead decoding drafts in parallel without a draft model through Jacobi iterations (Fu et al., 2024). PARD adapts a small parallel draft model that predicts a masked block in one pass (An et al., 2026a), and PARD-2 adds the target-confidence weighting that we adopt (An et al., 2026b). Diffusion language models have also served as drafters, training-free (Li et al., 2025b) or aligned to the verifier (Sandler et al., 2026).

Diffusion language models. Discrete diffusion with absorbing (mask) states (Austin et al., 2021) underlies masked diffusion language models trained with a weighted cross-entropy (Sahoo et al., 2024), which scale to large models (Nie et al., 2025). Block diffusion generates autoregressively across blocks and by diffusion within a block (Arriola et al., 2025), the formulation that DFlash's drafter inherits, and autoregressive models can be adapted into block-diffusion models that decode each block in parallel (Wu et al., 2026).

Flow matching. Flow matching regresses a velocity field that generates a prescribed probability path (Lipman et al., 2023). Discrete flow matching carries the construction to sequences of discrete tokens with continuous-time Markov chains (Campbell et al., 2024; Gat et al., 2024), including general discrete paths with kinetic-optimal velocities (Shaul et al., 2025); Lipman et al. (2024) give a unified treatment with code, including mixture paths such as Equation 6. Shortcut models condition the network on the step size so that one large step agrees with several small ones (Frans et al., 2025), and consistency models map points on a probability-flow ODE trajectory to its data endpoint

(Song et al., 2023); FS-DFM brings step-size conditioning to discrete flow matching (Monsefi et al., 2026a), and our (t, h) conditioning follows it.

Multimodal models. Visual instruction tuning connects a vision encoder to a language model (Liu et al., 2023), the family of the LLaVA-OneVision target (Li et al., 2025a); Qwen2.5-VL (Bai et al., 2025) is our second target. Speculative decoding for multimodal large language models has been studied on LLaVA (Gagrani et al., 2024). Visual tokens can also be pruned inside the model by attention (Chen et al., 2024a), the direction that SpecVLM applies to its draft model (Ji et al., 2025), whereas our drafter reads the target's hidden states and prunes nothing.

F RUN PROVENANCE

The chain and Tree-60 values in Table 1 come from our five-benchmark sweep. Tree-60 is selected by its macro speedup on LLaVA-OneVision-7B, not independently for each benchmark, and is used unchanged for Qwen2.5-VL-7B. Macros are computed from unrounded benchmark values; averaging the rounded displayed values can differ in the last digit. Table 6 uses the same checkpoint, clip manifests, seeds, and timing configuration. The DFlash+CAT and FM results in Tables 2, 3, and 4 come from the paired ablation run, in which both drafters decode the same clips. The DFlash+CAT tree row uses the single Tree-60 configuration with reported macro speedup $3.497\times$ across all benchmarks. Each benchmark's three seeds of 50 clips are fixed in advance and contain no clip that appears in the training data. Timing uses PyTorch 2.11 with CUDA 12.8 and Transformers 5.15.1 on one NVIDIA H100 80GB.