

DSTILL: Trajectory Consistency Distillation for Accelerated One-Step Drafting in Block Diffusion Speculative Decoding

Shreyas Kulkarni^{†*}, Uttam Gurram*, Madhav Kanda, Krishna Teja Chitty Venkata, Shichu Chen, Kartik Balasubramaniam, Supriyo Chakraborty, Sambit Sahu
AI Foundations, Capital One

Abstract

Block-diffusion speculative decoding accelerates large language model inference by drafting multiple tokens in parallel. However, one-step drafting in discrete diffusion is limited by token factorization error: each token is predicted from the same masked block, without conditioning on the tokens selected at other positions in that step. This can produce inconsistent drafts and reduce acceptance length. Iterative refinement mitigates this limitation by conditioning later predictions on previously committed tokens, though additional passes increase drafting cost. To address this trade-off between draft fidelity and inference latency, we introduce DSTILL, a two-stage framework that distills a target-conditioned drafter’s own multi-step refinement into one-step drafting. We first train the drafter to complete partially revealed blocks, then use its refinement trajectories to train a student to predict the final completion in one pass. Using a curriculum that gradually transitions from partial to fully masked inputs, we successfully transfer the multi-step refinement capabilities into one step. Across reasoning, math, code benchmarks on three model families DSTILL improves acceptance length by up to 28% and speedup by up to 25% over one-step DFlash, achieving up to $6.45\times$ speedup over autoregressive decoding.

1 Introduction

Speculative decoding accelerates Large Language Model (LLM) inference by using a small, inexpensive draft model to propose future tokens and verifying those proposals against the target model in a single parallel forward pass (Leviathan et al., 2023; Chen et al., 2023). Under the standard rejection-sampling verifier, this procedure is lossless as it preserves the output distribution of autoregressive sampling from the target model. An effective drafter

must achieve a high *acceptance length* τ , the expected number of draft tokens accepted per verification step, while keeping drafting latency low.

Target-conditioned¹ block-diffusion drafters, such as DFlash (Chen et al., 2026), predict an entire block in parallel. Each block starts in the *root state*: the latest verified token is visible as an anchor, and all future-token positions are masked. A one-pass drafter predicts each masked position without conditioning on the tokens selected at the other masked positions. Independent predictions can combine tokens from incompatible continuations (Figure 2a) (Lee et al., 2026). Refining the block over multiple denoising passes progressively commits high-confidence tokens, fixing their values for subsequent passes. The remaining positions can then condition on these commitments and improve their agreement with the target. However, the additional drafter passes increase latency, and the benefit of longer accepted prefixes may not offset this cost.

We introduce DSTILL, a two-stage trajectory consistency distillation framework that transfers the benefits of multi-step refinement to one-step speculative drafting. In the first stage, we train a target-conditioned drafter to complete partially masked blocks in a single forward pass. After training, we use this drafter as a teacher to collect multi-step denoising trajectories, recording the intermediate partially unmasked blocks and final completions which are fully unmasked which we refer to as *end-points*. In the second stage, a student initialized from the teacher learns to predict each trajectory’s final completion directly from its recorded states. At inference time, the student drafts a block in one forward pass, followed by standard target-model verification (Figure 1).

*Co-first authors. [†]Corresponding author. Email: shreyas.kulkarni2@capitalone.com.

¹A target-conditioned drafter uses hidden states computed by the target LLM as additional context for predicting draft tokens, rather than relying solely on the preceding token sequence. The target model remains frozen.

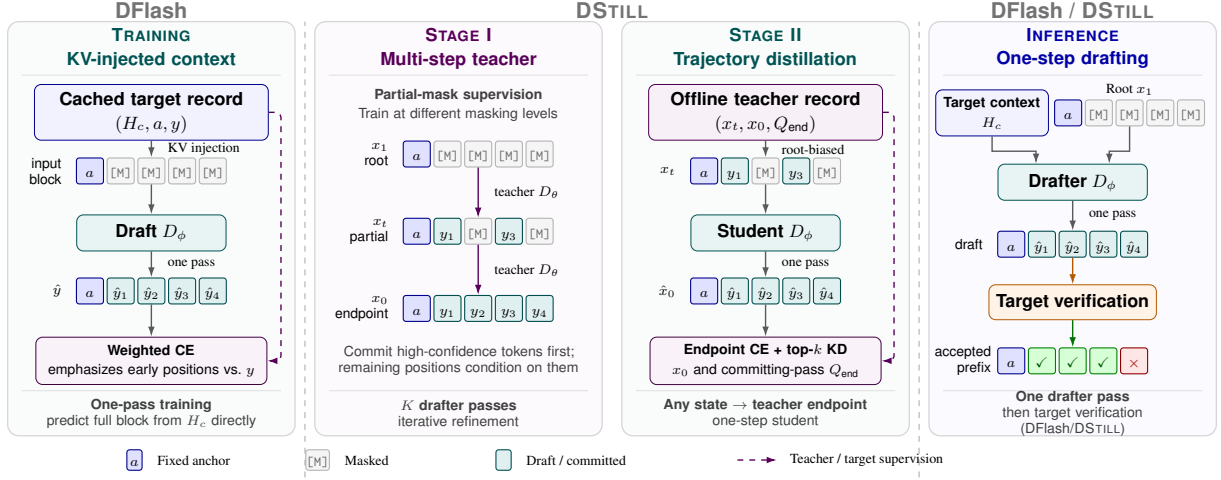


Figure 1: **DFlash** (first panel) trains a one-pass drafter conditioned on target features H_c to predict a masked block from an anchor token, using position-weighted cross-entropy (CE). **DSTILL** (middle two panels) first trains a partial-mask teacher and records its K -pass refinement trajectories. A student then learns to predict trajectory endpoints from root or intermediate states using endpoint CE and knowledge distillation from committing-pass distributions Q_{end} , with sampling increasingly favoring the anchor-only root. **Inference** (last panel) is identical for both methods: one forward pass proposes a block, followed by standard target verification.

During distillation, we sample a state from the teacher’s recorded trajectory and use it as input to the student. We resample this state across epochs while keeping the trajectory’s final completion fixed as the training target. This exposes the student to different partial commitments for the same block, with varying amounts of context for predicting the remaining tokens. To supervise one-step drafting, we use a curriculum that gradually increases *root-bias*, in which we increase the probability of sampling the root state through a schedule as we progress through training.

Our procedure adapts the theoretical foundation of Consistency Models (Song et al., 2023) to discrete diffusion by learning to map any intermediate point on a masking trajectory directly to the same endpoint. Just as these continuous models enforce a self-consistency property where outputs remain invariant for arbitrary points along a given probability flow trajectory, we train the student to map partially revealed intermediate blocks to a constant, fully refined target sequence. By internalizing this discrete refinement trajectory into its hidden states, the student learns to sharpen the blurry averaging of independent marginal probabilities (Figure 2b,c), mirroring how continuous consistency models learn to convert pure noise into coherent data samples using a single step. We make three main contributions:

Transferring Multi-Step Refinement to One-Step Drafting.

We introduce DSTILL, a framework that distills a drafter’s own multi-step refinement process into a one-step speculative drafter. By training the student to predict the teacher’s refined completion from various states along its trajectory, we successfully transfer multi-step capabilities to a single-step process. This approach recovers 35.8% and 28.6% of the multi-step teacher’s acceptance length improvements on GPT-OSS-120B and on Nemotron-3.5-30B-A3B respectively.

Zero-Overhead Efficiency Gains Across Models and Domains.

We evaluate DSTILL across three target-model families on benchmarks spanning reasoning, math, code, and chat. Compared to DFlash, our method achieves up to 28% higher acceptance length and up to 25% wall-clock speedup. Crucially, after offline distillation, these gains are realized using the exact same drafter architecture, avoiding test-time latency penalties associated with additional correction heads or tree-search decoding.

Generalization to Block Diffusion Speculators.

We also apply trajectory distillation to DSpark’s block diffusion backbone (Cheng et al., 2026), showing a 4.5% improvement in acceptance length (Appendix G) demonstrating its general applicability and motivating extensions to other block diffusion speculators.

2 Related Work

Speculative Decoding. Speculative decoding (Leviathan et al., 2023; Chen et al., 2023) accelerates autoregressive LLM generation by having a cheap draft model propose several future tokens, which the target model then verifies in a single forward pass; a rejection-sampling scheme guarantees that the resulting output distribution exactly matches that of the target model. The approach builds on earlier parallel-decoding ideas such as blockwise parallel decoding (Stern et al., 2018). Follow-up work extends verification from a single chain to a candidate tree (Miao et al., 2024; Chen et al., 2024), while draft-free variants such as Jacobi-style lookahead decoding (Fu et al., 2024) remove the draft model entirely at the cost of a weaker acceptance guarantee.

Feature-based and Multi-head Drafting. EA-GLUE (Li et al., 2024b) and its successors (Li et al., 2024a, 2025) draft tokens autoregressively using target-model features and embeddings of preceding tokens. Medusa (Cai et al., 2024) instead attaches multiple prediction heads to the target backbone, predicting future tokens independently in parallel. Hydra (Ankner et al., 2024) extends this design by conditioning each draft head on preceding draft tokens, introducing sequential dependencies within the proposed continuation while retaining lightweight prediction heads.

Parallel and Diffusion Drafters. DFlash (Chen et al., 2026) replaces sequential autoregressive drafting with a lightweight block-diffusion model that predicts an entire block in one pass while conditioning on hidden features from the target model. DFlare (Zhang et al., 2026a) scales this design by equipping each draft layer with a learned mixture over a broader set of target-layer features, increasing draft capacity. Other work restores intra-block dependencies through lightweight auxiliary modules: Domino (Huang et al., 2026) applies a causal head to the preliminary distributions produced by a parallel backbone, and DSpark (Cheng et al., 2026) couples a parallel backbone with a small sequential markov head while adapting the verification length based on confidence estimates. DDTree (Ringel and Romano, 2026) keeps predictions parallel but uses their per-position distributions to construct a draft tree for joint verification by the target model.

Masked and discrete diffusion language models. Absorbing-state (masked) diffusion formulates gen-

eration as iterative unmasking with weights derived from an evidence lower bound (ELBO) (Austin et al., 2021a; Lou et al., 2024; Sahoo et al., 2024), and large masked-diffusion language models such as LLaDA (Nie et al., 2025) and Dream (Ye et al., 2025) trade sampling steps for generation quality via confidence-based remasking.

Consistency and few-step distillation. Consistency models learn a map from any point on a diffusion trajectory to its endpoint, collapsing many-step sampling into one or a few forward passes (Song et al., 2023), later generalized to arbitrary source-target pairs along the trajectory (Kim et al., 2024). In the discrete/masked setting, self-distillation through time (SDTT) transfers multi-step unmasking distributions to fewer steps (Deschenaux and Gulcehre, 2025); Zhang et al. (2026b) distill a many-step masked-diffusion model through generative-trajectory matching, and Kim et al. (2026) combine consistency-guided distillation with cache-friendly block causality to accelerate masked-diffusion inference. Consistency LLMs fine-tune autoregressive models to predict fixed points from states along Jacobi-decoding trajectories, with an auxiliary ground-truth loss (Kou et al., 2024). These methods accelerate generation by the language model being distilled.

3 DSTILL

Figure 1 describes DSTILL. In Stage I, we train a target-conditioned drafter to complete partially revealed *reference blocks* from sequences generated by the autoregressive target model. The trained drafter then serves as the teacher. We freeze its parameters and record its multi-step refinement trajectories offline. In Stage II, a student initialized from the teacher learns to predict the teacher’s final completions from states along these trajectories. At inference, the student drafts a block in a single forward pass, and the target LLM verifies the proposed tokens.

3.1 Setup and notation

Let a be the latest verified token, and let H_c be the target model’s cached hidden states for the tokens before a . The teacher and student operate on blocks of B positions, with the fixed anchor a at position 0 and future tokens at positions $1, \dots, B-1$. In the *root state*, $x_1 = (a, [\text{MASK}], \dots, [\text{MASK}])$, all future-token positions are masked. Following DFlash (Chen et al., 2026), the drafter D_θ attends

bidirectionally within each block x and to target context H_c strictly before the anchor. It predicts a vocabulary distribution $p_{\theta,i}(\cdot \mid x, H_c)$ at each masked position i . Independently sampling the future tokens from the root predictions gives the proposal distribution

$$q_{\theta}(y_{1:B-1} \mid a, H_c) = \prod_{i=1}^{B-1} p_{\theta,i}(y_i \mid x_1, H_c). \quad (1)$$

Greedy drafting instead takes each position’s top-1 prediction. Figure 2a illustrates how factorized sampling can combine incompatible continuations.

Multi-step refinement allows later predictions to condition on tokens selected earlier. The drafter completes a block over K passes, selecting the highest-confidence masked positions and committing their predicted tokens at each pass. We measure confidence by the largest token probability at each position. Once committed, a token remains fixed for the remaining passes.

We denote the inputs to these passes by $x_1, \dots, x_K$, where x_1 is the root state. The completed block after the final pass is denoted by x_0 and called the *endpoint*. Appendix A gives the reveal schedule and verification procedure.

3.2 Stage I: Partial-mask teacher training

In Stage I, we train the drafter to complete partially revealed blocks in a single forward pass. This prepares the drafter for multi-step refinement, where later passes use previously committed tokens to predict the remaining tokens.

For a reference block $z = (a, z_1, \dots, z_{B-1})$, we sample a reveal probability r uniformly from a discrete set. We choose these probabilities so that training blocks contain, on average, as many visible tokens as the teacher sees at each refinement pass.

Given r , we reveal each future-token position independently with probability r . Revealed positions contain their reference tokens, and the remaining positions contain [MASK], producing the input block x^r . The expected fraction of masked future positions, or *corruption level*, is therefore $1 - r$. The anchor remains visible, and $r = 0$ gives the root state. We resample the reveal probability and positions across epochs.

Given this partially revealed block, we train the drafter to recover the reference tokens at the remaining masked positions. Let M denote these positions. For $|M| > 0$, the unweighted training

objective is

$$\mathcal{L}_{\text{teacher}} = -\frac{1}{|M|} \sum_{i \in M} \log p_{\theta,i}(z_i \mid x^r, H_c). \quad (2)$$

The reveal probabilities and optional loss weights are specified in Appendix B.1. After this training, we use the drafter’s own predictions to collect the refinement trajectories needed for student distillation.

3.3 Trajectory collection

We freeze the trained drafter as the teacher and run its confidence-guided refinement offline, once per training block. Each rollout starts from the root state and progressively reveals the teacher’s predictions. These predictions may differ from the reference tokens used in Stage I. We use greedy token selection for all models: at each selected position, the teacher commits its highest-probability token (Appendix E).

We record each trajectory so that its states and predictions can be reused during student training. Each record contains the final block x_0 and the pass at which each token was committed. At each pass, we also store the teacher’s top- k token IDs and probabilities at the masked positions.

Because committed tokens are never revised, the endpoint and commit passes are sufficient to reconstruct every input state. To reconstruct a state, we reveal endpoint tokens committed before that pass and mask the remaining future positions. The anchor remains visible throughout.

The reconstructed states serve as inputs to the student. We supervise the remaining masked positions using the teacher’s committed tokens and the probability distributions recorded when those tokens were committed. Record formats, reconstruction, and the collection algorithm are presented in Appendix B.2.

3.4 Stage II: One-step trajectory consistency distillation

The student D_{ϕ} shares the teacher’s architecture and is initialized from its weights. During distillation, we sample a state from the teacher’s recorded trajectory and use it as input to the student. The student is trained to predict the trajectory’s final completion at the positions that remain masked.

Different input states from the same trajectory share the same final completion. For each position that remains masked, the target token and teacher

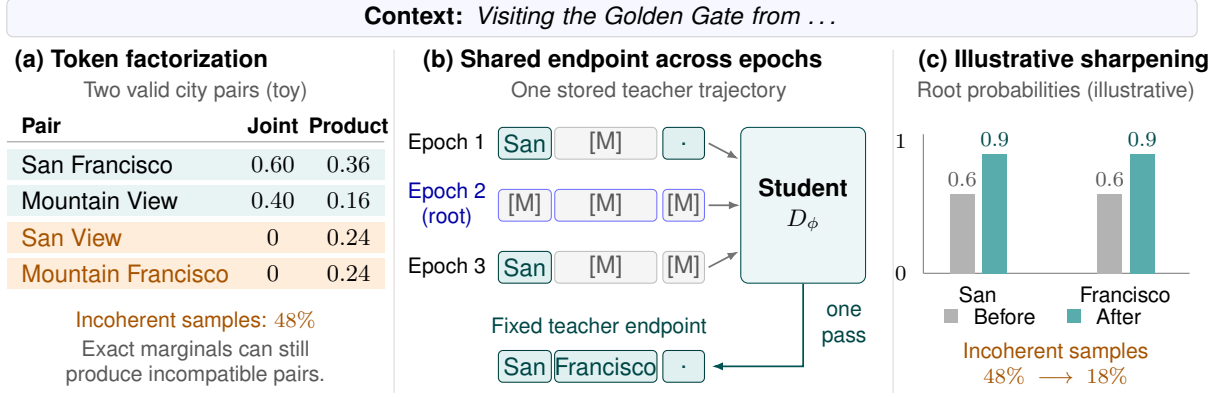


Figure 2: **Factorization and distillation.** (a) *Joint* shows a toy autoregressive target’s token-pair probabilities. *Product* multiplies its per-position marginals, corresponding to independent sampling and assigning 48% probability to incoherent pairs. (b) Recorded states resampled across epochs share one endpoint. (c) Toy probabilities illustrate concentration toward that endpoint; the student remains factorized. [M] denotes masking. The anchor is omitted in this figure.

distribution stay fixed across these states. We refer to this shared supervision as *trajectory consistency*. Figure 2b,c illustrates how this supervision can increase the probabilities of tokens from the teacher’s selected completion. Incompatible combinations can then become less likely while the student’s output distribution remains factorized.

Root-biased state sampling. To expose the student to different amounts of visible context, we train on both root and intermediate states. Intermediate states provide committed tokens that help predict the remaining positions, while root states match the input used for one-step drafting. With probability p_{root} , we sample the root state x_1 ; otherwise, we uniformly sample a recorded intermediate state. Blocks without intermediate states use root.

As training progresses, we increase p_{root} from 0.5 to 0.85 using a cosine schedule, establishing a curriculum that places greater emphasis on the root state later in training (Appendix B.3). The input state is resampled across epochs, while trajectory’s final completion and teacher distributions remain fixed.

Endpoint supervision. After selecting an input state, we supervise the student’s predictions at the positions that remain masked. The training loss mask restricts supervision to response-token positions, and the anchor is excluded.

Each supervised position receives two training signals: the token committed by the teacher and the teacher’s probability distribution from the pass that committed it. Cross-entropy (CE) trains the student to predict the committed token, while knowledge

distillation (KD) transfers the teacher’s uncertainty.

To compute the KD term, we use the teacher’s retained top- k tokens and group the remaining probability mass into a single category. Let $Q_{\text{end},i}$ denote this distribution for position i . Let $P_{\phi,i}$ denote the student’s distribution given (x_t, H_c) , using the same teacher-selected token IDs and the same aggregate category.

For each supervised position i , we combine CE and KD with weight $\alpha \in [0, 1]$ on the KD term:

$$\mathcal{L}_i = (1 - \alpha) \underbrace{[-\log p_{\phi,i}(x_{0,i} | x_t, H_c)]}_{\text{endpoint CE}} + \alpha \underbrace{D_{\text{KL}}(Q_{\text{end},i} \| P_{\phi,i})}_{\text{top-}k \text{ KD}}. \quad (3)$$

We use $\alpha = 0.2$ and $k = 16$, and average over the positions included in the loss.

We optimize this objective while keeping the teacher fixed. Both drafters reuse the target’s token embeddings and language-model (LM) head, which projects hidden states to vocabulary logits. These shared components also remain frozen, so Stage II updates only the remaining student parameters. Appendix B.4 gives the exact probability construction, optional trajectory weights, batch aggregation, and training algorithm.

3.5 Inference

At inference, the student receives the target’s cached hidden states H_c and the root state x_1 . A single forward pass produces the proposed block $\hat{x}_0 = (a, \hat{y}_1, \dots, \hat{y}_{B-1})$.

Table 1: Generation speedup S and mean acceptance length τ across target models and benchmarks at concurrency $C = 1$, temperature = 0, reasoning = on/medium. Params denotes drafter parameters in Millions.

Model	Method	Params (M)	GSM8K		MATH-500		HumanEval		MBPP		MT-Bench		SWE-Bench		AVERAGE	
			S	τ	S	τ	S	τ	S	τ	S	τ	S	τ	S	τ
GPT-OSS-120B	DFlash	785	4.81×	6.3	4.01×	5.17	3.53×	4.58	3.57×	4.63	2.42×	3.09	2.33×	2.96	3.45×	4.45
	DStill (Ours)	785	5.17×	6.79	4.27×	5.54	3.7×	4.83	3.8×	4.97	2.6×	3.34	2.53×	3.25	3.68×	4.79
Gemma-4-31B	DFlash	1536	5.85×	6.78	5.9×	6.84	5.2×	6.03	4.3×	5.00	3.16×	3.67	2.74×	3.17	4.53×	5.25
	DStill (Ours)	1536	6.45×	7.48	6.28×	7.28	5.59×	6.49	4.76×	5.52	3.47×	4.03	2.92×	3.38	4.91×	5.70
Gemma-4-26B-A4B	DFlash	430	2.86×	4.57	2.98×	4.65	2.58×	4.09	2.23×	3.55	1.78×	2.81	1.62×	2.54	2.34×	3.70
	DStill (Ours)	430	3.42×	5.57	3.53×	5.59	3.11×	5.02	2.79×	4.53	2.08×	3.38	1.75×	2.81	2.78×	4.48
Nemotron-3.5-30B-A3B	DFlash	481	2.62×	4.81	2.66×	4.86	2.31×	4.27	2.08×	3.79	1.77×	3.18	1.62×	2.89	2.18×	3.97
	DStill (Ours)	481	2.98×	5.62	3.02×	5.64	2.57×	4.88	2.33×	4.36	1.94×	3.6	1.66×	3.04	2.42×	4.52

For independently sampled one-pass proposals, the student’s per-position distributions provide the proposal conditionals required by the standard rejection-sampling verifier. This verifier preserves the target model’s sampling distribution. Greedy verification accepts tokens that match the target’s top-1 predictions.

After a rejection, the target supplies a correction token. If all draft tokens are accepted, it supplies a bonus token. This target-generated token becomes the next anchor, starting another drafting and verification step. At fixed one-pass drafting cost, longer accepted prefixes can improve end-to-end speedup; Section 4 evaluates resulting wall-clock trade-off.

4 Experiments

Table 1 presents the main comparison across target models and benchmarks, reporting generation speedup and acceptance length. We also measure how much of the teacher’s eight-pass refinement gain transfers to one-pass drafting and examine the teacher objective, student supervision, and state-sampling curriculum.

4.1 Experimental Setup

Models and benchmarks. We evaluate GPT-OSS-120B (OpenAI, 2025b), Gemma-4-31B and Gemma-4-26B-A4B (Gemma Team, 2026), and Nemotron-3.5-30B-A3B (NVIDIA, 2026), covering dense, transformer mixture-of-experts (MoE), and hybrid Mamba-attention MoE architectures. We run the benchmarks on GSM8K (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2021; Lightman et al., 2024), HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021b), SWE-bench (Jimenez et al., 2024), and MT-Bench (Zheng et al., 2023). We evaluate 128 prompts each from GSM8K, MATH-500, and MBPP; 164 each from

Table 2: Generation speedup S (relative to standard autoregressive decoding) for GPT-OSS-120B across benchmarks and request concurrencies $C \in \{1, 8, 16, 32\}$, at temperature 0 and 1, reasoning_effort = medium. Avg τ is the mean acceptance length averaged over the four concurrencies. For each dataset, temperature, and concurrency, faster method is in **bold**.

Dataset	Method	Temperature = 0					Temperature = 1				
		Avg τ	Speedup S ($\times$)				Avg τ	Speedup S ($\times$)			
			$C=1$	$C=8$	$C=16$	$C=32$		$C=1$	$C=8$	$C=16$	$C=32$
GSM8K	DFlash	6.3	4.81	3.01	3.04	3.59	5.54	4.17	2.05	2.55	3.3
	DStill	6.86	5.17	3.26	3.29	3.92	5.93	4.54	2.21	2.75	3.42
MATH-500	DFlash	5.16	4.01	2.47	2.53	3	4.37	3.29	2.15	2.09	2.49
	DStill	5.51	4.27	2.67	2.62	3.06	4.78	3.69	2.34	2.16	2.71
HumanEval	DFlash	4.54	3.53	2.15	2.19	2.61	4.11	3.12	1.93	1.9	2.39
	DStill	4.88	3.7	2.31	2.38	2.69	4.4	3.33	2.07	2.05	2.46
MBPP	DFlash	4.7	3.57	2.26	2.29	2.65	4.25	3.25	2	1.97	2.4
	DStill	4.98	3.8	2.39	2.42	2.69	4.58	3.48	2.15	2.12	2.56
MT-Bench	DFlash	3.1	2.42	1.53	1.52	1.86	2.92	2.25	1.44	1.37	1.76
	DStill	3.31	2.6	1.64	1.63	1.98	3.05	2.39	1.47	1.42	1.77
SWE-Bench	DFlash	2.9	2.33	1.43	1.41	1.62	2.66	2.06	1.29	1.24	1.5
	DStill	3.19	2.53	1.52	1.55	1.77	2.84	2.2	1.37	1.33	1.55

HumanEval and SWE-bench; and 80 from MT-Bench. Further evaluation details are provided in Appendix E.

Training. Each target regenerates assistant responses in OpenPerfectBlend (mlabonne, n.d.); its hidden-state cache is shared across both training stages and trajectory collection. Our DStill drafters use block size $B = 16$, comprising one anchor and 15 proposed tokens. We collect teacher trajectories using $K = 8$ refinement passes with greedy token selection. Unless otherwise stated, student distillation uses $\alpha = 0.2$, top- $k = 16$, and a cosine root-probability schedule from 0.5 to 0.85. The student is initialized from the teacher, and the target embeddings and LM head remain frozen throughout training. Drafter architectures are listed in Appendix Table 10, with further training details in Appendix E.

Table 3: Generation speedup S (relative to standard autoregressive decoding) for the Nemotron-3.5-30B-A3B trained drafters across benchmarks and request concurrencies $C \in \{1, 8, 16, 32\}$, for reasoning on/off and temperature 0/1.

Dataset	Method	Temperature = 0					Temperature = 1				
		Avg τ	Speedup S ($\times$)				Avg τ	Speedup S ($\times$)			
			C=1	C=8	C=16	C=32		C=1	C=8	C=16	C=32
Reasoning = ON											
GSM8K	DFlash	4.81	2.62	1.42	2.48	1.98	4.18	2.42	1.16	2.04	1.7
	DStill	5.71	2.98	1.67	2.79	2.19	4.73	2.64	1.3	2.34	1.9
MATH-500	DFlash	4.77	2.66	1.36	2.35	2.06	4.06	2.3	1.16	1.97	1.78
	DStill	5.52	3.02	1.52	2.69	2.39	4.53	2.47	1.28	2.19	1.95
HumanEval	DFlash	4.29	2.31	1.19	2.05	1.89	4.21	2.35	1.17	1.99	1.81
	DStill	4.89	2.57	1.33	2.33	2.13	4.74	2.63	1.27	2.24	2
MBPP	DFlash	3.78	2.08	1.01	1.77	1.64	3.76	2.11	1.02	1.7	1.57
	DStill	4.37	2.33	1.14	2.03	1.9	4.29	2.33	1.15	1.94	1.79
MT-Bench	DFlash	3.15	1.77	0.88	1.51	1.47	3.1	1.78	0.86	1.47	1.43
	DStill	3.59	1.94	0.96	1.7	1.68	3.45	1.93	0.95	1.62	1.58
SWE-Bench	DFlash	2.9	1.62	0.79	1.35	1.3	2.86	1.64	0.79	1.3	1.23
	DStill	3.03	1.66	0.82	1.4	1.35	2.97	1.67	0.81	1.33	1.3
Reasoning = OFF											
GSM8K	DFlash	5.65	3.1	2.27	3.33	2.22	4.88	2.71	1.85	2.69	2.01
	DStill	7.02	3.75	2.66	3.83	2.47	5.9	3.37	2.13	3.04	2.24
MATH-500	DFlash	4.95	2.69	1.4	2.47	2.09	4.43	2.54	1.27	2.23	1.96
	DStill	5.51	2.91	1.56	2.77	2.31	4.92	2.68	1.44	2.42	2.19
HumanEval	DFlash	5.99	3.2	2.15	3.07	2	5.32	2.88	1.89	2.74	1.83
	DStill	7.46	3.88	2.64	3.43	2.28	6.52	3.46	2.26	3.22	1.95
MBPP	DFlash	4.42	2.42	1.21	2.02	1.65	4.2	2.35	1.18	1.92	1.56
	DStill	5.33	2.82	1.49	2.4	1.95	5.05	2.77	1.41	2.24	1.85
MT-Bench	DFlash	2.91	1.7	0.89	1.45	1.42	2.76	1.61	0.84	1.36	1.21
	DStill	3.3	1.91	0.98	1.67	1.5	3.13	1.78	0.94	1.52	1.4
SWE-Bench	DFlash	2.98	1.68	0.88	1.46	1.35	2.83	1.65	0.84	1.36	1.24
	DStill	3.22	1.77	0.93	1.54	1.45	3.11	1.77	0.87	1.51	1.32

Comparisons. Our primary baseline is a draft model trained with the DFlash objective (Chen et al., 2026). Within each target, this baseline and DStill share the drafter architecture, training corpus, block size, and one-pass drafting budget. For GPT-OSS-120B and Nemotron-3.5-30B-A3B, we evaluate the anchor-only baseline and partial-mask teacher with one and eight passes to measure the effect of iterative refinement and its transfer to the student (Table 4). We also apply trajectory distillation to a DSpark drafter (Appendix Table 11). Autoregressive target decoding provides the speedup reference.

Inference setup. Table 1 reports results at request concurrency $C = 1$ and temperature 0, with reasoning enabled and reasoning effort set to medium for GPT-OSS-120B. For GPT-OSS-120B and Nemotron-3.5-30B-A3B, we additionally evaluate $C \in \{1, 8, 16, 32\}$ at temperatures 0 and 1; Nemotron is evaluated with reasoning both enabled and disabled (Tables 2 and 3). Prompts, generation settings, and hardware are matched across methods within each comparison.

Evaluation metrics. Mean acceptance length τ is the average number of tokens in the accepted block prefix, including the anchor token. We report speculative decoding speedup over autoregressive target decoding as $S = t_{AR}/t_{spec}$, where t_{AR} and t_{spec} denote generation time per output token under autoregressive and speculative decoding, respectively. Averages reported in the tables are unweighted means across benchmarks or concurrency levels, as indicated.

4.2 Main Results

Benchmark comparison. We compare DStill with DFlash on reasoning, mathematics, code generation, and conversation benchmarks for three model families across temperature and reasoning settings in Tables 1, 2, and 3. We train iso-parameter models for both the methods using the OpenPerfectBlend dataset.

Performance across reasoning modes. With reasoning enabled, DStill consistently outperforms DFlash by 6.7%-18.8% on average speedups over autoregressive decoding and 7.6%-21.1% on acceptance length as shown in Table 1. On GPT-OSS-120B, DStill shows speed up improvements over DFlash across temperature and concurrency settings as in Table 2. In non-thinking mode, Nemotron achieves speedups of up to 3.88 $\times$ and 3.46 $\times$ at temperatures 0 and 1, respectively, at concurrency 1 (Table 3).

Transfer of refinement gains. Table 4 compares DFlash, the teacher, and DStill across refinement steps. Eight teacher passes improve acceptance length by 18.0% on GPT-OSS-120B and 17.9% on Nemotron. One-pass DStill recovers 35.8% and 28.6% of these gains, outperforming DFlash by 7.6% and 19.4%, respectively. The largest teacher-student gap is on Nemotron HumanEval (8.57 vs. 7.46; 1.11 tokens). Eight-pass DFlash even underperforms the one-pass teacher, likely due to a mismatch between its root-only training and partially revealed refinement states, motivating Stage I.

Serving performance. DStill improves throughput across all reported settings, achieving speedups of up to 5.2 $\times$ for GPT-OSS-120B and 3.0 $\times$ for Nemotron-3.5-30B-A3B over autoregressive decoding. At concurrency 32, speedups remain 2.7-3.9 $\times$ and 2.1-2.4 $\times$, respectively, showing that the decoding gains persist under concurrent serving.

Table 4: Draft acceptance across target models and benchmarks. Denoising steps denote sequential drafter forward passes per block. Average is the unweighted mean across the six benchmarks. Recovery is the percentage of the teacher’s one-to-eight-pass gain in average acceptance length recovered by one-pass DSTILL. Thinking mode is *disabled* in the measurements with Nemotron-3.5-30B-A3B so the numbers will be different from Table 1

Target Model	Drafter	Denoising Steps	GSM8K	MATH-500	HumanEval	MBPP	MT-Bench	SWE-Bench	Average	Recovery (%)
			τ	τ	τ	τ	τ	τ	τ	
GPT-OSS-120B	DFlash	1	6.30	5.17	4.58	4.63	3.09	2.96	4.45	–
	DFlash	8	4.97	4.06	3.75	3.76	2.88	2.79	3.70	–
	Teacher	1	6.44	5.10	4.63	4.65	3.19	2.97	4.50	–
	Teacher	8	7.58	6.14	5.48	5.62	3.60	3.41	5.31	–
	DSTILL	1	6.79	5.54	4.83	4.97	3.34	3.25	4.79	35.80
Nemotron-3.5-30B-A3B	DFlash	1	5.84	4.84	5.98	4.44	2.90	2.92	4.49	–
	DFlash	8	4.94	4.19	4.70	3.90	2.84	2.78	3.89	–
	Teacher	1	6.77	5.43	6.85	5.02	3.18	3.31	5.10	–
	Teacher	8	7.80	6.48	8.57	6.17	3.51	3.52	6.01	–
	DSTILL	1	6.94	5.65	7.46	5.40	3.32	3.35	5.36	28.57

4.3 Ablation Studies

We examine the effects of continued baseline training and student state sampling. Evaluations use mean acceptance length with greedy decoding and one drafting pass on $8 \times \text{H100}$ GPUs. Unless otherwise stated, student distillation runs for 3 epochs with $\alpha = 0.2$ and a cosine p_{root} schedule from 0.5 to 0.85. Additional ablations are reported in Appendix C.

Continued one-step training. Table 5 compares DFlash trained for 10 epochs with DSTILL in its default setting on Gemma-4-31B. DSTILL achieves higher acceptance length across all six benchmarks, with improvements ranging from 0.21 tokens on SWE-bench to 0.7 tokens on GSM8K. These results suggest that trajectory distillation provides gains beyond longer DFlash training.

Table 5: Acceptance length of DSTILL and DFlash trained for 10 epochs on Gemma-4-31B.

Dataset	DFlash (5 epochs) $\uparrow$	DFlash (10 epochs) $\uparrow$	DSTILL (default) $\uparrow$
GSM8K	6.78	6.54	7.48
MATH500	6.84	6.67	7.28
HumanEval	6.03	5.96	6.49
MBPP	5.00	5.00	5.52
MT-Bench	3.67	3.57	4.03
SWE-Bench	3.17	3.13	3.38

State sampling. We compare root-only training, which samples exclusively from the fully masked block state ($p_{\text{root}} = 1$), with our cosine schedule, which progressively increases the probability of sampling this state from 0.5 to 0.85. We trained the drafters for 16 epochs, and conduct this ablation on one-eighth of the training dataset, with all other training settings held fixed. Table 6 shows the

cosine schedule improves τ across all six benchmarks, with the largest gains on GSM8K (+0.22) and MBPP (+0.10). These results suggest that progressively increasing the probability of sampling the fully masked state, while retaining exposure to intermediate trajectory states, improves student acceptance length.

Table 6: Mean acceptance length τ for root-only and cosine state sampling on Gemma-4-26B-A4B at concurrency 16.

Dataset	$p_{\text{root}} = 1\uparrow$	Cosine $\uparrow$	Δ
GSM8K	4.77	4.99	+0.22
HumanEval	4.58	4.60	+0.02
MATH500	4.90	4.97	+0.07
MBPP	4.11	4.21	+0.10
MT-Bench	3.16	3.19	+0.03
SWE-Bench	2.57	2.58	+0.01
Average	4.01	4.09	+0.08

5 Conclusion

We presented a trajectory consistency distillation procedure for block-diffusion speculative decoding: a two-stage training pipeline that trains a multi-step masked-diffusion drafter with partial-mask supervision and then distills its confidence-guided denoising trajectory into a one-step student via an offline trajectory-replay pipeline. Distillation trains the student to map any state along the teacher’s trajectory, including the root state used at inference time, directly to the teacher’s endpoint, under a curriculum that shifts weight toward the root state as training progresses. We show that this procedure recovers 28.6%–35.8% of the acceptance length improvements of a multi-step teacher in a single step student with zero test-time overheads. Across models and domains, we demonstrate improvements

of 28% in acceptance length and 25% in speedup. Finally, we demonstrate our procedure’s general applicability to diffusion speculators by improving DSpark acceptance lengths by 4.5%.

Limitations

DSTILL has a higher one-time training cost than direct DFlash training, comprising partial-mask teacher training, offline trajectory collection and storage, and student distillation. These additional costs can be amortized over subsequent generation, but only at a large enough inference workload. The student learns the teacher’s selected continuations and inherits errors in those endpoints, and the student’s one-pass outputs also remain factorized. Our experiments only cover the target models, drafter configurations, and benchmarks reported here. Transfer to other block sizes, architectures beyond DFlash and DSpark, and target-feature layouts requires further evaluation. Acceptance-length gains must be considered together with measured latency under the inference workload. Actual latency reductions depend on the specific inference workload, hardware serving kernels, and concurrency limits.

References

- Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. 2024. Hydra: Sequentially-dependent draft heads for Medusa decoding. In *Conference on Language Modeling (COLM)*.
- Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. 2021a. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 17981–17993.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021b. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.
- Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *International Conference on Machine Learning (ICML)*.
- Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.
- Jian Chen, Yesheng Liang, and Zhijian Liu. 2026. DFlash: Block diffusion for flash speculative decoding. *arXiv preprint arXiv:2602.06036*.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. 2024. Sequoia: Scalable, robust, and hardware-aware speculative decoding. *arXiv preprint arXiv:2402.12374*.
- Xin Cheng, Xingkai Yu, Chenze Shao, Jiashi Li, Yunfan Xiong, Yi Qian, Jiaqi Zhu, Shirong Ma, Xiaokang Zhang, Jiasheng Ye, Qinyu Chen, Chengqi Deng, Jiping Yu, Damai Dai, Zhengyan Zhang, Yixuan Wei, Yixuan Tan, Wenkai Yang, Runxin Xu, and 14 others. 2026. DSpark: Confidence-scheduled speculative decoding with semi-autoregressive generation. *arXiv preprint arXiv:2607.05147*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Justin Deschenaux and Caglar Gulcehre. 2025. Beyond autoregression: Fast LLMs via self-distillation through time. In *International Conference on Learning Representations (ICLR)*.
- Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. 2024. Break the sequential dependency of LLM inference using lookahead decoding. In *International Conference on Machine Learning (ICML)*.
- Gemma Team. 2026. Gemma 4 Technical Report. *arXiv preprint arXiv:2607.02770*.
- Google. 2026a. gemma-4-26B-A4B-it. <https://huggingface.co/google/gemma-4-26B-A4B-it>.
- Google. 2026b. gemma-4-31B-it. <https://huggingface.co/google/gemma-4-31B-it>.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*.
- Jianuo Huang, Yaojie Zhang, Qituan Zhang, Hao Lin, Hanlin Xu, and Linfeng Zhang. 2026. Domino: Decoupling causal modeling from autoregressive drafting in speculative decoding. *arXiv preprint arXiv:2605.29707*.
- Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. 2024. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*.

- Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. 2024. Consistency trajectory models: Learning probability flow ODE trajectory of diffusion. In *International Conference on Learning Representations (ICLR)*.
- Minseo Kim, Chenfeng Xu, Coleman Hooper, Harman Singh, Ben Athiwaratkun, Ce Zhang, Kurt Keutzer, and Amir Gholami. 2026. CDLM: Consistency diffusion language models for faster sampling. In *Proceedings of the 9th MLSys Conference*.
- Siqi Kou, Lanxiang Hu, Zhezhi He, Zhijie Deng, and Hao Zhang. 2024. CLLMs: Consistency large language models. In *International Conference on Machine Learning (ICML)*.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.
- Chanhyuk Lee, Jaehoon Yoo, Manan Agarwal, Sheel Shah, Jerry Huang, Aditi Raghunathan, Seunghoon Hong, Nicholas M. Boffi, and Jinwoo Kim. 2026. Flow map language models: One-step language modeling via continuous denoising. *arXiv preprint arXiv:2602.16813*.
- Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning (ICML)*, pages 19274–19286. PMLR.
- Shenggui Li, Chao Wang, Yikai Zhu, Yubo Wang, Fan Yin, Shuai Shi, Yefei Chen, Xiaomin Dong, Qiaoling Chen, Jin Pan, Ji Li, Yineng Zhang, Lei Yu, Yonggang Wen, Ivor Tsang, and Tianwei Zhang. 2026. SpecForge: A flexible and efficient open-source training framework for speculative decoding. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 71081–71095. PMLR.
- Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024a. EAGLE-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024b. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *International Conference on Machine Learning (ICML)*.
- Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2025. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. *arXiv preprint arXiv:2503.01840*.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let’s verify step by step. In *International Conference on Learning Representations (ICLR)*.
- Aaron Lou, Chenlin Meng, and Stefano Ermon. 2024. Discrete diffusion modeling by estimating the ratios of the data distribution. In *International Conference on Machine Learning (ICML)*.
- 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. 2024. 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 (ASPLOS)*.
- mlabonne. n.d. OpenPerfectBlend. Hugging Face dataset card. Accessed October 2, 2026.
- Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025. Large language diffusion models. *arXiv preprint arXiv:2502.09992*.
- NVIDIA. 2026. NVIDIA-Nemotron-3.5-Lightning-30B-A3B-BF16 Model Card. Hugging Face model card. Associated released model configuration; accessed October 2, 2026.
- OpenAI. 2025a. gpt-oss-120b. <https://huggingface.co/openai/gpt-oss-120b>. Accessed October 12, 2026.
- OpenAI. 2025b. gpt-oss-120b & gpt-oss-20b Model Card. *arXiv preprint arXiv:2508.10925*.
- Liran Ringel and Yaniv Romano. 2026. Accelerating speculative decoding with block diffusion draft trees. *arXiv preprint arXiv:2604.12989*.
- Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. 2023. Consistency models. In *International Conference on Machine Learning (ICML)*.
- Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31.
- Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Dream 7B: Diffusion large language models. *arXiv preprint arXiv:2508.15487*.

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

Tunyu Zhang, Xinxi Zhang, Ligong Han, Haizhou Shi, Xiaoxiao He, Zhuowei Li, Hao Wang, Kai Xu, Akash Srivastava, Chengzhi Mao, Hao Wang, Vladimir Pavlovic, and Dimitris N. Metaxas. 2026b. Few-step diffusion language models via trajectory self-distillation. *arXiv preprint arXiv:2602.12262*.

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. 2023. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 46595–46623.

A Drafting and Verification Details

A.1 Drafting Setup

Let $\mathcal{T}$ denote the target language model and $(u_{1:S}, a)$ the current verified prefix, where a is its most recent token. We cache the target’s hidden states H_c at a fixed set of layers over $u_{1:S}$. A draft block contains B positions: position 0 holds the fixed anchor a , and positions $1, \dots, B-1$ are the future tokens to be proposed. Uncommitted positions contain [MASK], so the initial block is $x_1 = (a, [\text{MASK}], \dots, [\text{MASK}])$.

The drafter D_θ receives embeddings of the current block state x together with H_c , attends bidirectionally within the block and to context strictly before its anchor, and produces a vocabulary distribution $p_{\theta,i}(\cdot | x, H_c)$ at each masked position i . Independently sampling all non-anchor positions from the root predictions gives the factorized proposal in Equation 1. Greedy drafting selects the most probable token at each position. In both cases, all positions are predicted from the same root state, yielding the one-pass factorization discussed in Section 1.

The target scores the proposal in one causal forward pass. Standard lossless speculative sampling uses the target’s prefix-conditional probabilities and the proposal’s actual left-to-right conditionals, together with rejection correction (Leviathan et al., 2023; Chen et al., 2023). Under greedy verification, acceptance is determined by matching the target’s top-1 predictions. Let $L \in \{0, \dots, B-1\}$ be the length of the accepted draft prefix. We denote mean acceptance length by $\tau = 1 + \mathbb{E}[L]$.

A.2 Denoising Trajectories

A multi-step drafter resolves the block over K sequential passes. Let $x_t \equiv x^{(t)}$ denote the state before pass $t \in \{1, \dots, K\}$, starting from the anchor-only root x_1 . The set of positions already committed is $c^{(t)} \subseteq \{0, \dots, B-1\}$, with $c^{(1)} = \{0\}$ and cardinality $|c^{(t)}|$. Thus the root contains the given anchor but zero committed future tokens. The remaining masked positions are $M_t = \{1, \dots, B-1\} \setminus c^{(t)}$; we write M when the pass index is unnecessary.

At pass t , the drafter predicts from (x_t, H_c) and assigns each masked position confidence $\text{conf}_i = \max_v p_{\theta,i}(v | x_t, H_c)$. It commits the

$$n_t = \left\lceil \frac{B - |c^{(t)}|}{K - t + 1} \right\rceil \quad (4)$$

highest-confidence positions, writing their greedy or sampled tokens into the block and leaving earlier commitments unchanged. The updated state $x^{(t+1)}$ is the input to the next pass. The schedule allocates the remaining masked positions across the remaining passes, committing all positions after K passes. The rollout starts from the root predictions in Equation 1; later passes condition on tokens committed earlier.

We reserve $x_0 \equiv x^{(K+1)} = (a, y_1, \dots, y_{B-1})$ for the fully committed teacher endpoint. Here, t indexes forward passes and the subscript 0 identifies the completed block. The student’s proposed block is $\hat{x}_0 = (a, \hat{y}_1, \dots, \hat{y}_{B-1})$. Figure 1 uses these state conventions. Lossless verification of a sampled multi-step proposal requires its actual left-to-right conditionals, which can differ from the distributions recorded at commitment time.

This rollout defines the intermediate states x_t and endpoint x_0 used in Section 3: Stage I trains the teacher across intermediate corruption levels, and Stage II uses recorded teacher trajectories to supervise a one-pass student.

B Training and Trajectory Details

B.1 Reveal schedule and teacher objective

During training, we cache H_c from teacher-forced sequences and construct blocks around sampled anchor positions. Each block’s attention is restricted to its own positions and target context strictly before its anchor, with bidirectional attention within the block. This attention mask prevents cross-block leakage and allows multiple anchored blocks to be

scored in parallel. Let $z = (a, z_1, \dots, z_{B-1})$ denote a reference training block.

We derive corruption levels from the number of positions committed before each pass of Equation 4. Let m_t include the anchor. The corresponding reveal probabilities for *non-anchor* positions are

$$\begin{aligned} m_1 &= 1, \\ m_{t+1} &= m_t + \left\lceil \frac{B - m_t}{K - t + 1} \right\rceil, \\ \mathcal{R} &= \left\{ \frac{m_t - 1}{B - 1} : 1 \leq t \leq K, m_t < B \right\}. \end{aligned} \quad (5)$$

For each block, we sample r uniformly from $\mathcal{R}$ and independently reveal each non-anchor position with probability r ; the anchor is always revealed. Revealed positions use their reference-token embeddings, and the others use [MASK], yielding the corrupted block x^r . The reveal level and Bernoulli mask are sampled anew when a block is revisited across epochs. The root level is $r = 0$. Each level matches the expected number of revealed positions to a rollout state. Bernoulli sampling varies the revealed counts and position subsets. Fully committed levels are excluded from $\mathcal{R}$. Any Bernoulli outcome with no masked positions contributes no loss and is excluded from the training average.

For a nonempty masked set M , the per-block objective is

$$\begin{aligned} \ell_i^{\text{CE}} &= -\log p_{\theta,i}(z_i | x^r, H_c), \\ \mathcal{L}_{\text{teacher}} &= w_{\text{block}} \frac{\sum_{i \in M} w_{\text{pos}}(i) \ell_i^{\text{CE}}}{\sum_{i \in M} w_{\text{pos}}(i)}. \end{aligned} \quad (6)$$

Training minimizes the arithmetic mean of this objective over nonempty blocks; batches with no nonempty blocks are skipped. The block weight scales each block’s normalized position loss and controls its contribution to the training average. Both weights are optional and default to 1.

Mask-count weighting. To emphasize heavily masked blocks relative to nearly complete ones, we use

$$w_{\text{block}} = \frac{(|M|/(B-1))^{\alpha_m}}{\mathbb{E}_{j \sim \text{Unif}\{1, \dots, B-1\}}[(j/(B-1))^{\alpha_m}]}. \quad (7)$$

Setting $\alpha_m = 0$ gives $w_{\text{block}} = 1$, while $\alpha_m > 0$ increases the relative weight of heavily masked blocks. The denominator normalizes against a uniform *reference* distribution over mask counts. The mean weight under the training sampler depends on its mask-count distribution.

Position decay. Following DFlash (Chen et al., 2026), we optionally give greater weight to early positions because acceptance depends on a prefix. We use $w_{\text{pos}}(i) = \exp(-(i-1)/\gamma)$ for $\gamma > 0$. With position decay disabled, $w_{\text{pos}}(i) = 1$.

The discrete levels emphasize corruption depths associated with the teacher’s refinement schedule, while random reveal subsets broaden the training distribution. This aligns expected reveal counts; the positions and reference tokens differ from the confidence-selected predictions exposed during a teacher rollout. Stage-I examples reveal *reference tokens* independently; they are not connected rollouts through the drafter’s own predictions. They therefore do not supply the trajectory endpoints needed for distillation. We obtain these by running the frozen Stage-I checkpoint through the full refinement schedule. Equivalent trajectories already recorded from that checkpoint under the same rollout policy could be reused; no additional teacher-training stage is needed.

B.2 Trajectory records and reconstruction

We run the frozen teacher’s confidence-guided rollout offline, once per training block, and record the tokens it commits. Each record contains

- $x_0 \in \{1, \dots, V\}^B$: the fully committed endpoint, where V is the vocabulary size;
- $\kappa \in \{0, \dots, K\}^B$: each position’s commit pass, with $\kappa_0 = 0$;
- $(\text{ids}_i^{(t)}, \log p_i^{(t)})$: the top- k token ids and their full-softmax log-probabilities at each position while it remains masked.

Because commitments are never revised, x_0 and κ reconstruct every intermediate block without storing a separate $K \times B$ sequence of block states:

$$x_{t,i} = \begin{cases} x_{0,i}, & \kappa_i < t, \\ [\text{MASK}], & \kappa_i \geq t. \end{cases} \quad (8)$$

For example, a token committed at pass 2 is masked in the input to that pass and visible in the next state. The revealed token is the *teacher-committed* token $x_{0,i}$, which may differ from the reference token z_i .

For each non-anchor position i , let $S_i = \text{ids}_i^{(\kappa_i)}$ be the retained ids at its committing pass. For $v \in S_i$, recover the original teacher probability as $q_{i,v} = \exp(\log p_i^{(\kappa_i)}(v))$, where the stored log-probability corresponds to id v . These probabilities

Algorithm 1 Offline Trajectory Collection

Input: Frozen teacher D_θ , context H_c , anchor a , block size B , steps K , top- k

Output: Endpoint x_0 , commit passes κ , top- k records

- 1: $x^{(1)} \leftarrow (a, [\text{MASK}], \dots, [\text{MASK}])$
- 2: $c^{(1)} \leftarrow \{0\}; \kappa_0 \leftarrow 0$
- 3: **for** $t = 1$ **to** K **do**
- 4: $M_t \leftarrow \{1, \dots, B-1\} \setminus c^{(t)}$
- 5: logits $\leftarrow D_\theta(x^{(t)}, H_c)$
- 6: $P_i^{(t)} \leftarrow \text{softmax}(\text{logits}_i), i \in M_t$
- 7: $\text{conf}_i \leftarrow \max_v P_{i,v}^{(t)}, i \in M_t$
- 8: Store top- k ids in $\text{ids}^{(t)}$
- 9: Store their log $P_{i,v}^{(t)}$ in $\text{logp}^{(t)}$
- 10: $n_t \leftarrow \lceil (B - |c^{(t)}|) / (K - t + 1) \rceil$
- 11: $J_t \leftarrow$ top n_t indices in M_t by confidence
- 12: $x^{(t+1)} \leftarrow x^{(t)}$
- 13: Fill J_t with greedy or sampled teacher tokens
- 14: $\kappa_i \leftarrow t, i \in J_t$
- 15: $c^{(t+1)} \leftarrow c^{(t)} \cup J_t$
- 16: **end for**
- 17: $x_0 \leftarrow x^{(K+1)}$
- 18: **return** $x_0, \kappa, \{(\text{ids}^{(t)}, \text{logp}^{(t)})\}_{t=1}^K$

retain their full-vocabulary normalization. Define $q_{i,\text{other}} = 1 - \sum_{v \in S_i} q_{i,v}$ and let $Q_{\text{end},i}$ denote the resulting distribution over $S_i \cup \{\text{other}\}$. We write $Q_{\text{end}} = (Q_{\text{end},i})_{i=1}^{B-1}$. Thus, the record supports both exact state reconstruction and committing-pass distribution supervision.

B.3 State-sampling schedule

With probability p_{root} , we sample the anchor-only root x_1 ($t = 1$). Otherwise, we sample t uniformly from $\{2, \dots, \kappa_{\text{max}}\}$, where $\kappa_{\text{max}} = \max_i \kappa_i$ is the last commit pass. Blocks with no intermediate state fall back to $t = 1$. We resample t on each visit to a trajectory across epochs, reconstructing the corresponding recorded state each time. The endpoint x_0 and committing-pass distributions remain fixed. Thus, different partial commitments provide supervision toward the same completed block; the student learns to predict all remaining scored positions in one pass. This combines the root state with intermediate states that provide additional endpoint supervision. Teacher-committed tokens in intermediate states provide additional context for predicting the remaining masked positions early in training. We ramp the root probability with training

progress $\rho \in [0, 1]$:

$$\begin{aligned}
 u(\rho) &= \frac{\rho - w}{1 - c - w}, \\
 s(\rho) &= \frac{1 - \cos(\pi u(\rho))}{2}, \\
 \Delta p &= p_{\text{end}} - p_{\text{start}}, \\
 p_{\text{root}}(\rho) &= \begin{cases} p_{\text{start}}, & \rho \leq w, \\ p_{\text{start}} + \Delta p s(\rho), & w < \rho < 1 - c, \\ p_{\text{end}}, & \rho \geq 1 - c. \end{cases} \quad (9)
 \end{aligned}$$

Here $w, c \geq 0$ are warmup and cooldown fractions with $w + c < 1$. We use $p_{\text{start}} = 0.5$, $p_{\text{end}} = 0.85$, $w = 0.10$, and $c = 0.30$. The schedule gives intermediate states more weight early and increasingly emphasizes the root state. In particular, the root receives direct supervision throughout training, with sampling probability 0.5 initially and 0.85 late in training. At the root, every scored non-anchor position is masked and receives endpoint supervision. Root behavior is therefore trained explicitly rather than inferred only from partially committed inputs.

B.4 Distillation distributions and loss aggregation

We reconstruct x_t using Equation 8 and restrict M_t to still-masked, scored response positions, excluding the anchor. Scored positions are response-token positions selected for supervision by the training loss mask. For $v \in S_i$, let $p_{i,v} = p_{\phi,i}(v \mid x_t, H_c)$ and $p_{i,\text{other}} = 1 - \sum_{v \in S_i} p_{i,v}$. Define $P_{\phi,i}$ over the same $k + 1$ categories as $Q_{\text{end},i}$. The per-position objective is Equation 3. The CE term targets the teacher’s committed token, while KD transfers its committing-pass probability mass, including the omitted tail:

$$\begin{aligned}
 D_{\text{KL}}(Q_{\text{end},i} \parallel P_{\phi,i}) &= \sum_{v \in S_i} q_{i,v} \log \frac{q_{i,v}}{p_{i,v}} \\
 &\quad + q_{i,\text{other}} \log \frac{q_{i,\text{other}}}{p_{i,\text{other}}}. \quad (10)
 \end{aligned}$$

The KL divergence is computed over the retained token ids and a single aggregate tail category. Storing the teacher’s top- k probabilities is sufficient to evaluate this coarsened loss. We use $\alpha = 0.2$; Section 4 reports its ablation.

Trajectory-quality weighting. Optionally, we favor positions where the teacher’s endpoint agrees with the reference training block z . Let d be the first scored non-anchor position with $x_{0,d} \neq z_d$, or

Algorithm 2 Trajectory Consistency Distillation

Input: Student D_ϕ , training batch with stored trajectories, contexts H_c , progress ρ
Output: Updated ϕ , unless the batch has zero weight

```

1:  $p_{\text{root}} \leftarrow$  Equation 9 at  $\rho$ 
2:  $\text{lossSum} \leftarrow 0$ ;  $\text{weightSum} \leftarrow 0$ 
3: for each block in the batch do
4:   Resample  $t$  from the root-biased schedule
5:   Reconstruct  $x_t$  using Equation 8
6:    $M_t \leftarrow \{i : \kappa_i \geq t, i \text{ scored}, i \neq 0\}$ 
7:   if  $M_t \neq \emptyset$  then
8:      $\text{logits} \leftarrow D_\phi(x_t, H_c)$ 
9:     Recover  $Q_{\text{end},i}$  at pass  $\kappa_i, i \in M_t$ 
10:    Compute  $\mathcal{L}_i$  using Equation 3,  $i \in M_t$ 
11:    Set  $w_i$  using Equation 11, or 1 if disabled
12:     $\text{lossSum} \leftarrow \text{lossSum} + \sum_{i \in M_t} w_i \mathcal{L}_i$ 
13:     $\text{weightSum} \leftarrow \text{weightSum} + \sum_{i \in M_t} w_i$ 
14:   end if
15: end for
16: if  $\text{weightSum} > 0$  then
17:   Backpropagate  $\text{lossSum}/\text{weightSum}$ 
18:   Update student parameters  $\phi$ 
19: end if

```

$d = B$ if there is no disagreement. We use

$$w_j = \begin{cases} 1, & j < d, \\ w_{\text{first}}, & j = d, \\ w_{\text{first}} \beta^{j-d}, & j > d, \end{cases} \quad (11)$$

with $0 < w_{\text{first}} \leq 1$ and $0 < \beta < 1$; a flat tail weight applies if β is unset. The weights provide a *reference-agreement heuristic* that reduces supervision after a mismatch while retaining the trajectory. A teacher token that differs from a sampled reference continuation can still be accepted under lossless speculative sampling.

For a batch of blocks indexed by b , we aggregate over scored masked positions using

$$\mathcal{L}_{\text{student}} = \frac{\sum_b \sum_{i \in M_{t_b}} w_{b,i} \mathcal{L}_{b,i}}{\sum_b \sum_{i \in M_{t_b}} w_{b,i}}. \quad (12)$$

When trajectory weighting is disabled, $w_{b,i} = 1$, recovering the ordinary mean token loss. Only blocks with scored masked positions enter the batch loss; batches with no such positions are skipped. Algorithm 2 summarizes one update.

C Ablation Results

Student supervision. To investigate the effect of the supervision objective, we compare endpoint CE ($\alpha = 0$), KD ($\alpha = 1$), and their combination, CE+KD ($\alpha = 0.2$), in Equation 3. As shown in Table 7, the three objectives achieve broadly comparable acceptance lengths across benchmarks. CE and KD provide complementary supervision

signals, combining endpoint-level targets with distributional information from the teacher.

Table 7: Student supervision ablation at concurrency 16. Higher τ is better. $\alpha = 0$ denotes endpoint CE, $\alpha = 0.2$ denotes CE+KD (ours), and $\alpha = 1$ denotes KD.

Dataset	$\alpha = 0$	$\alpha = 0.2$	$\alpha = 1$
GSM8K	5.6	5.4	5.5
HumanEval	5.0	5.1	5.1
MATH500	5.7	5.6	5.6
MBPP	4.5	4.4	4.5
MT-Bench	3.3	3.4	3.3
SWE-Bench	2.8	2.8	2.8

Trajectory-quality weighting. The distillation loss can optionally down-weight supervision at positions after the teacher’s recorded endpoint first diverges from the reference continuation, following the reference-agreement heuristic in Equation 11. We compare a student distilled with uniform position weights against one that enables this weighting with $w_{\text{first}} = 0.4$ and tail decay $\beta = 0.5$. Table 8 reports student acceptance length. Weighting increases τ on all six benchmarks, by 0.04 to 0.12 tokens, raising the average from 4.72 to 4.80 (+1.7%). Concentrating supervision on the agreed prefix, rather than on lower-agreement suffix positions that are less likely to be accepted, improves one-step drafting at no additional inference cost.

Table 8: Trajectory-quality weighting ablation on GPT-OSS-120B. The weighted run additionally enables the reference-agreement weighting of Equation 11 with $w_{\text{first}}=0.4$ and tail decay $\beta=0.5$.

Dataset	Unweighted $\uparrow$	Weighted $\uparrow$	Δ
GSM8K	6.96	7.01	+0.05
MATH-500	5.35	5.47	+0.12
HumanEval	4.77	4.88	+0.11
MBPP	4.95	4.99	+0.04
MT-Bench	3.26	3.32	+0.06
SWE-Bench	3.06	3.15	+0.09
Average	4.72	4.80	+0.08

D Additional Experimental Results

Table 4 compares drafting cost and acceptance for the anchor-only baseline, the teacher evaluated with one or eight passes, and the distilled one-pass student. The teacher comparison isolates the refinement gain available for distillation.

E Experimental Details

Verifier models. Table 9 lists the target-model specifications. The A suffix denotes nominal active

parameters for the Gemma and Nemotron MoE models.

Training data. OpenPerfectBlend (mlabonne, n.d.) supplies an instruction mixture spanning general conversation, mathematics, and code. For GPT-OSS-120B, assistant turns are regenerated through SGLang with reasoning effort set to medium and temperature 0.7. The regenerated conversations are chat-templated and converted to Arrow. A teacher-forced pass through the completed sequences caches input ids, the training loss mask, and target hidden states at `target_layer_ids`, using sequence length 24,576. The cache is reused across teacher training, trajectory collection, and student distillation.

Drafter architecture. The GPT-OSS-120B drafter is an injection-conditioned Qwen3-style transformer with $B = 16$, mask token id 200,000, and `target_layer_ids = [1, 9, 17, 25, 33]`. The backbone has 8 decoder layers, hidden size 2880, intermediate size 7680, and grouped-query attention with 64 query heads and 8 key/value heads. At each layer, a linear projection of the concatenated target features (input dimension 5×2880), followed by RMSNorm, supplies additional attention keys and values. The query stream carries the current block’s token embeddings, including mask embeddings at unrevealed positions. Target embeddings and the LM head are frozen and shared; only draft-specific parameters are optimized. The configuration estimate is approximately 0.8B drafter parameters; the trainable parameter count is 785M. Drafter hidden dimensions and selected target layers for Gemma-4-31B, Gemma-4-26B-A4B, and Nemotron-3.5-30B-A3B are respectively 5376 ([1, 12, 23, 35, 46, 57]), 2816 ([1, 6, 11, 17, 22, 27]), and 2688 ([1, 11, 20, 30, 39, 49]).

Training configuration. Our training implementation builds on SpecForge (Li et al., 2026). For GPT-OSS-120B, Stage I trains the teacher for 10 epochs on 16 A100 nodes, and Stage II distills the student for 3 epochs on 4 H100 nodes. Both stages use a learning-rate warmup ratio of 0.03 and gradient-norm clipping at 1.0. Trajectory collection uses $K = 8$ passes with greedy token selection and retains the top-16 probabilities. Trajectory-quality weighting is disabled except in its dedicated ablation.

Benchmarks and generation settings. We evaluate 128 prompts each from GSM8K, MATH-500, and MBPP; 164 each from HumanEval and SWE-bench; and 80 from MT-Bench. Generation uses a maximum of 4096 new tokens, with temperatures and reasoning settings specified in the corresponding tables. GPT-OSS-120B uses reasoning effort medium. Autoregressive and speculative runs use the same requests and generation settings. Benchmark averages are unweighted arithmetic means across the six datasets, computed separately for speedup and acceptance length. In the concurrency tables, Avg τ averages acceptance length over $C \in \{1, 8, 16, 32\}$.

Inference implementation. We use 8 H100 GPUs with vLLM version 0.30.0 (Kwon et al., 2023), tensor-parallel size 8, and GPU memory utilization 0.8. One-pass DFlash and DStill evaluations use `num_denoising_steps=1` and `num_speculative_tokens=15`. The refinement comparison additionally evaluates drafters with eight denoising passes. DFlash and DSpark use the corresponding dflash and dspark speculative configurations.

Measurement. Each inference configuration is evaluated once after four warmup runs, each allowing up to 128 generated tokens. Generation time per output token is computed by dividing the measured generation time by the actual number of generated tokens.

Teacher-gain recovery and ablation controls. We compute recovery from acceptance lengths averaged across the six benchmarks:

$$\text{Recovery (\%)} = 100 \frac{\tau_{\text{student}} - \tau_{\text{teacher},1}}{\tau_{\text{teacher},8} - \tau_{\text{teacher},1}}.$$

Recovery is reported when the denominator is positive. The state-sampling ablation uses Gemma-4-26B-A4B, with both variants trained for 16 epochs on one-eighth of the training corpus and evaluated on all six benchmarks at concurrency 16. All other training settings are held fixed. Root-only sampling uses $p_{\text{root}} = 1$. The cosine schedule increases this probability from 0.5 to 0.85, with warmup and cooldown fractions of 0.10 and 0.30, respectively. Its training-averaged root probability is approximately 0.71.

F Reproducibility

Appendices A and B detail the drafting, verification, and training procedures. Section 4.1 and Ap-

Table 9: Target specifications from the model reports (OpenAI, 2025b; Gemma Team, 2026; NVIDIA, 2026) and released configurations.

Target Model	Size	Architecture	Precision	Context	# Layers
GPT-OSS-120B (OpenAI, 2025a)	120B	MoE	BF16	131,072 (128K)	36
Gemma-4-31B (Google, 2026b)	31B	Dense	BF16	262,144 (256K)	60
Gemma-4-26B-A4B (Google, 2026a)	26B-A4B	MoE	BF16	262,144 (256K)	30
Nemotron-3.5-30B-A3B (NVIDIA, 2026)	30B-A3B	Hybrid MoE	BF16	1,048,576 (1M)	52

Table 10: Architectural configuration of every draft model used in this paper, extracted directly from each model’s config.json. *Target Hidden Layers* lists the indices of the target model’s layers whose hidden states are injected into the drafter; *Attention* denotes sliding-window (SWA, with window size as subscript) vs. full attention over the drafted block. “(Ours)” rows are checkpoints we trained for DFlash and DSpark these are our own training runs of the published method on a backbone not released by its original authors. DStill is our proposed distillation method.

Model	Method	Params (M)	Hidden	Layers	Target Hidden Layers	Attention
GPT-OSS-120B	DFlash	785	2880	8	1, 9, 17, 25, 33	Full
	DStill (Ours)	785	2880	8	1, 9, 17, 25, 33	Full
Gemma-4-31B	DFlash	1536	5376	5	1, 12, 23, 35, 46, 57	SWA ₂₀₄₈ +1 Full
	DStill (Ours)	1536	5376	5	1, 12, 23, 35, 46, 57	SWA ₂₀₄₈ +1 Full
Gemma-4-26B-A4B	DFlash	430	2816	5	1, 6, 11, 17, 22, 27	SWA ₂₀₄₈
	DStill (Ours)	430	2816	5	1, 6, 11, 17, 22, 27	SWA ₂₀₄₈
Nemotron-3.5-30B-A3B	DFlash	481	2688	6	1, 11, 20, 30, 39, 49	SWA ₂₀₄₈
	DStill (Ours)	481	2688	6	1, 11, 20, 30, 39, 49	SWA ₂₀₄₈

pendix E describe the datasets, model configurations, hyperparameters, hardware, and evaluation settings.

G Application to DSpark

We apply trajectory consistency distillation to DSpark (Cheng et al., 2026), jointly training its backbone and auxiliary heads. We collect $K = 8$ teacher refinement trajectories and distill a one-pass student, retaining the Markov and confidence losses in both training stages. Table 11 reports the per-benchmark acceptance lengths and generation speedups.

Table 11: Mean acceptance length τ and generation speedup S over autoregressive decoding for DSpark and DSpark+DSTILL. Average is the unweighted mean across the six benchmarks.

Dataset	Acceptance length τ		Speedup S	
	DSpark	+DSTILL	DSpark	+DSTILL
GSM8K	6.74	6.89	4.23	4.29
HumanEval	5.28	5.68	3.42	3.66
MATH-500	4.84	4.97	3.14	3.22
MBPP	5.01	5.11	3.25	3.29
MT-Bench	3.31	3.51	2.19	2.33
SWE-Bench	3.03	3.28	2.02	2.19
Average	4.70	4.91	3.04	3.16