Trajectory as the Teacher: Few-Step Discrete Flow Matching via Energy-Navigated Distillation

Amin Karimi Monsefi Apple
Dominic Culver Apple
Nikhil Bhendawade Apple
Manuel R. Ciosici Apple

Yizhe Zhang* Meta
Irina Belousova Apple

Abstract

Discrete flow matching generates text by iteratively transforming noise tokens into coherent language, but may require hundreds of forward passes. Distillation uses the multi-step trajectory to train a student to reproduce the process in a few steps. When the student underperforms, the usual explanation is insufficient capacity. We argue the opposite: *the trajectory is the bottleneck, not the student*. Each training trajectory is built through a chain of blind stochastic jumps with no evaluation of sequence quality. A single bad decision at an early midpoint propagates through subsequent steps, yet the student must imitate the result. Trajectory-Shaped Discrete Flow Matching (TS-DFM) replaces these blind jumps with guided navigation: a lightweight energy compass evaluates candidate continuations at each midpoint, selecting the most coherent. All shaping is training-only; inference cost is unchanged. On 170M-parameter language modeling, the shaped student at 8 steps achieves 32% lower perplexity than the unguided $1\,024$-step teacher while being $128\times$ faster, with gains consistent across source distributions and three evaluators of increasing scale. TS-DFM achieves the best perplexity of any discrete-generation baseline we compare against, including methods trained on $6\times$ more data or using $5\times$ larger models.

1 Introduction

Discrete flow matching (DFM) models generate text by iteratively transforming an uninformative source sequence, such as uniform random tokens or mask tokens, into coherent language through hundreds or thousands of refinement steps [Gat et al., 2024, Campbell et al., 2024, Nie et al., 2025, Ye et al., 2025]. While quality is on par with that of other large language models (LLMs), it is orders of magnitude slower than autoregressive LLMs as a full forward pass is required to generate each token.

Distillation compresses this process: a student model learns to reproduce in a few steps what the iterative process achieves in many (e.g., FS-DFM [Monsefi et al., 2026]). The student never sees training data, but rather learns entirely from *trajectories*, the specific paths through token space that the iterative process traces from a source state to finished text. When the distilled student underperforms, the standard explanation is insufficient model capacity. In this paper, we argue that this explanation ignores the trajectory and that *the trajectory is the bottleneck, not the student.*

Trajectory construction requires converting continuous velocity logits into concrete token sequences at each intermediate step. Because tokens are discrete, this conversion demands a hard stochastic decision at every position. No evaluation of the resulting sequence ever occurs: the process commits

*Work done while at Apple.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

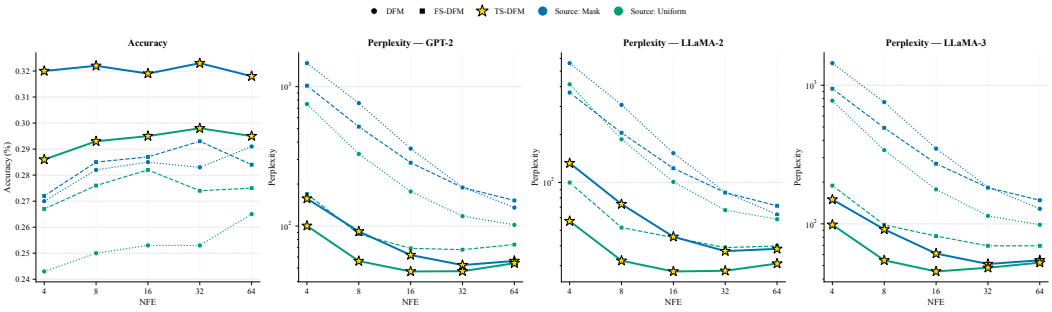


Figure 1: **Generation quality across step budgets, evaluators, and source distributions.** From left to right: next-token prediction accuracy and perplexity under three evaluators. TS-DFM achieves the lowest perplexity, with the largest gains at low NFE (number of function evaluations) where few-step generation matters most.

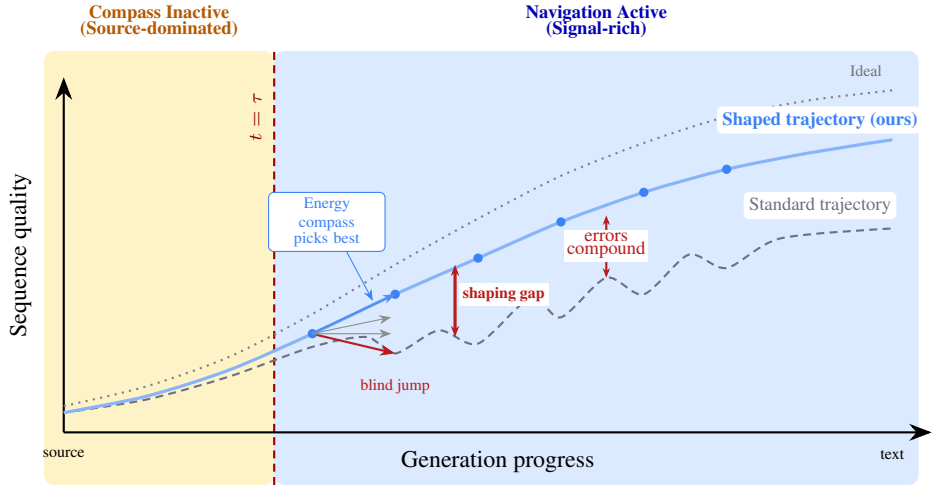


Figure 2: During training, standard trajectory construction compounds errors through blind stochastic jumps. Navigation shaping selects among candidates via an energy compass, activating at $t \geq \tau$.

to a single random realization and moves on. Each subsequent step then conditions on this potentially degraded state, so a single poor sample propagates forward and corrupts every downstream computation. In FS-DFM's RK-4 estimator [Monsefi et al., 2026], three such blind jumps are chained; error introduced at the first midpoint shifts the velocity evaluation at the second, which degrades the third, and the final target absorbs the accumulated drift. The student faithfully imitates this drifted target because it is the only supervision it receives.

We introduce Trajectory-Shaped Discrete Flow Matching (TS-DFM), a framework that shapes training trajectories to maximize their teaching value via a mechanism we call *navigation shaping*. At each midpoint where the standard method makes a single blind stochastic jump, we generate multiple candidate continuations and use a lightweight *energy compass*—a scalar energy model trained on generation-aware negatives—to select the highest-quality one. Empirically, the compass achieves 98.5% accuracy at distinguishing real flow states from generation-relevant corruptions. A *Sequence-to-Token* policy first picks the best candidate globally, then refines it at the token level using velocity confidence as a free proxy. A time threshold τ completes the design: at early flow times the sequence is too uninformative for the compass to discriminate among candidates, so navigation activates only once sufficient linguistic structure has emerged (Figure 2).

On 170M-parameter language modeling, TS-DFM at 8 steps achieves 56.1 GPT-2 perplexity with a uniform source—a 36% reduction over FS-DFM and 32% below the full, unguided 1 024-step teacher, while being $128\times$ faster (Figure 1). For the mask source, improvements are even larger ($5.6\times$ over FS-DFM at 8 steps). TS-DFM outperforms Duo [Sahoo et al., 2025] and SDTT [Deschenaux and

Gulcehre, 2025], achieving state-of-the-art perplexity among few-step discrete generation methods. Improvements hold across three evaluators of increasing scale (GPT-2, LLaMA-2, LLaMA-3). All shaping is training-only, adding 2.0–$2.4\times$ one-time overhead; inference is unchanged.

To our knowledge, this is the first work to treat the intermediate states of a discrete flow trajectory—not just the endpoints—as a target for quality optimization during distillation. Navigation shaping is orthogonal to the distillation mechanism itself. We show it within the FS-DFM distiller, and in principle it can be applied on top of other trajectory-based distillation frameworks. Our contributions are:

- **Trajectory quality as the distillation bottleneck**: We show that the performance ceiling in few-step DFM is set by trajectory quality, not student capacity. Blind stochastic jumps at RK-4 midpoints introduce compounding errors that degrade supervision; optimizing these midpoint states breaks through the ceiling.
- **Sequence-to-Token navigation shaping**: A two-phase policy that selects the best among K candidate continuations at the sequence level, then refines it at the token level, gated by a time threshold τ.
- **Generation-aware energy compass**: A lightweight energy model trained with velocity near-miss negatives that achieves 98.5% corruption-detection accuracy on partially-revealed flow states.
- **No inference cost**: All shaping is training-only; inference remains unchanged.

2 Related work

Discrete diffusion [Austin et al., 2021, He et al., 2023, Xie et al., 2025, Song et al., 2025, Arriola et al., 2025, Yang et al., 2025, Ye et al., 2024] has been explored for text generation in models like LLaDA [Nie et al., 2025] and Dream [Ye et al., 2025]. These generate text via iterative refinement from a source distribution, but quality degrades sharply when the number of sampling steps is reduced.

A natural remedy is distillation: in the continuous setting, progressive distillation [Salimans and Ho, 2022], consistency models [Song et al., 2023], and single-step methods [Meng et al., 2023, Frans et al., 2025] have all been used to compress many-step samplers into few-step students.

In the discrete setting, SDTT [Deschenaux and Gulcehre, 2025] distills masked diffusion models (MDLM [Sahoo et al., 2024], SEDD [Lou et al., 2024]) through iterative self-distillation rounds, achieving strong few-step quality but requiring multiple rounds of training and a large base model. ReMDM [Wang et al., 2025] improves few-step masked diffusion at inference time through remasking strategies (cap, conf, rescale, loop), avoiding retraining but adding per-step overhead. Duo [Sahoo et al., 2025] establishes a duality between Gaussian and uniform-state discrete diffusion, enabling consistency distillation in the discrete setting and achieving competitive few-step generation, though at the cost of substantially more training data. FS-DFM [Monsefi et al., 2026] brought distillation to discrete flow matching with step-size conditioning and RK-4 trajectory estimation. T3D [Zhang et al., 2026] concurrently proposes trajectory self-distillation with a reverse-KL objective that encourages mode-seeking behavior, addressing the few-step gap through changes to the distillation loss rather than the trajectory construction.

This paper builds on FS-DFM's distillation mechanism but addresses an orthogonal question: not *how* to distill, but *what the student learns from*. That the quality of the training signal matters for distillation is well established—teacher capacity and temperature shape student performance in knowledge distillation [Hinton et al., 2015], and trajectory sampling strategies influence outcomes in diffusion distillation [Yin et al., 2024]—yet prior work on discrete sequences has not attempted to optimize the trajectories themselves. Unlike SDTT [Deschenaux and Gulcehre, 2025], which improves the student through multiple distillation rounds on the same trajectories, and ReMDM [Wang et al., 2025], which corrects errors at inference time, TS-DFM intervenes at the source: shaping the trajectories during training so that the student receives higher-quality supervision from the start.

To shape trajectories, we introduce an energy model that scores partially revealed flow states and selects the highest-quality candidate at each step. Energy-based models have a long history in generative modeling [LeCun et al., 2006, Song and Kingma, 2021]: in continuous diffusion they underlie classifier-free and classifier-based guidance [Dhariwal and Nichol, 2021, Du et al., 2023,

Liu et al., 2022, Bansal et al., 2023], while in the text domain they have been used for reranking and controlled generation of complete sequences [Deng et al., 2020, Khalifa et al., 2021]. Our use differs in three respects: (i) the energy model operates on *partial* states—mixtures of meaningful and source tokens—rather than complete sequences or vectors; (ii) it acts as a discrete selection criterion rather than providing gradients; and (iii) it participates exclusively at training time, incurring no inference cost.

3 Background and motivation

Discrete flow matching (DFM) [Gat et al., 2024, Campbell et al., 2024] generates text by traversing a trajectory from a source state to text. A velocity field v_θ iteratively transforms a source sequence x_0 into coherent text x_1 over N steps, where x_0 is drawn from a source distribution (uniform over the vocabulary, or mask tokens). At intermediate time $t \in [0, 1]$, the state x_t is a position-wise mixture of source and target tokens: each position j independently holds either the target or the source token,

$$x_t[j] = \begin{cases} x_1[j] & \text{with probability } \alpha(t), \\ x_0[j] & \text{with probability } 1 - \alpha(t), \end{cases} \tag{1}$$

where $\alpha(t)$ is a monotonically increasing schedule with $\alpha(0) = 0$ and $\alpha(1) = 1$. This interpolation defines both the forward sampling path during training and the partially formed sequences that the energy compass must evaluate.

The **Few-Step Discrete Flow-Matching** framework [Monsefi et al., 2026] compresses long teacher trajectories into few-step student ones using three models. The **teacher** $v_\theta(x_t, t)$ is a frozen pretrained DFM velocity model. The **student** $s_\psi(x_t, t, h)$ shares the teacher's architecture with an additional step-size conditioning mechanism and is initialized from the teacher. The **semi-teacher** $\tilde{v}_\omega(x_t, t, h)$ is an exponential moving average (EMA) of the student's weights. For small step sizes, the teacher's single-step velocity is used directly; for large step sizes, the semi-teacher estimates the velocity via fourth-order Runge–Kutta (RK-4) integration, constructing multi-step targets that the teacher alone cannot provide in a single evaluation:

$$k_1 = \tilde{v}_\omega\big(x_t,\, t,\, \tfrac{h}{2}\big), \qquad k_{i+1} = \tilde{v}_\omega\big(x_{\text{mid},i},\, t_{\text{mid},i},\, \tfrac{h}{2}\big), \;\; i = 1, 2, 3, \tag{2}$$

where $t_{\text{mid},1} = t_{\text{mid},2} = t + \frac{h}{2}$, $t_{\text{mid},3} = t + h$, and each midpoint $x_{\text{mid},i}$ is a concrete token sequence constructed from the previous stage's velocity via a stochastic continuous-time Markov chain (CTMC) jump. The final RK-4 target $v_{\text{RK-4}} = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$ is the standard weighted combination, and the student minimizes $\mathcal{L}_{\text{KL}} = \text{KL}(s_\psi(x_t, t, h) \| v_{\text{RK-4}})$. The student's *entire* training signal comes from these RK-4 targets: the semi-teacher's constructed trajectory at each training step *is* the lesson. The student has no independent access to data, no alternative examples, no error signal beyond "match this target."

Why the trajectory is a poor teacher. To construct each midpoint $x_{\text{mid},i}$ in the RK-4 chain, velocity logits must be converted into a concrete token sequence. Since tokens are discrete, this requires a stochastic jump: a Poisson-rate process determines whether each position j transitions, and a categorical sample chooses the new token. Concretely, given the current state x and velocity logits k at time t with sub-step size h':

$$x_{\text{mid}}[j] = \begin{cases} \hat{x}_1[j] \sim \text{Cat}\big(k[j]\big) & \text{with prob. } p_{\text{jump}}(j), \\ x[j] & \text{otherwise}, \end{cases} \tag{3}$$

where $\hat{x}_1[j]$ is sampled from the velocity's categorical distribution at position j, and $p_{\text{jump}}(j)$ is the transition probability derived from the CTMC rate and sub-step size h'. This procedure is applied at each of the three internal RK-4 stages. Each application produces one specific realization *without any quality evaluation*, and the problem compounds through the chain. A poor first sample affects the velocity evaluation for subsequent steps, each compounding the degradation:

$$k_1 \xrightarrow{\text{sample}} \underbrace{x_{\text{mid},1}}_{\text{degraded}} \rightarrow k_2 \xrightarrow{\text{sample}} \underbrace{x_{\text{mid},2}}_{\text{worse}} \rightarrow k_3 \xrightarrow{\text{sample}} \underbrace{x_{\text{mid},3}}_{\text{worst}} \rightarrow k_4. \tag{4}$$

The final target $v_{\text{RK-4}} = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$ absorbs compounded errors from all midpoints, leading to trajectory drift (Figure 2). Because the student learns to imitate the drifted trajectory, it cannot recover the quality lost during trajectory construction, and learns the compounded errors.

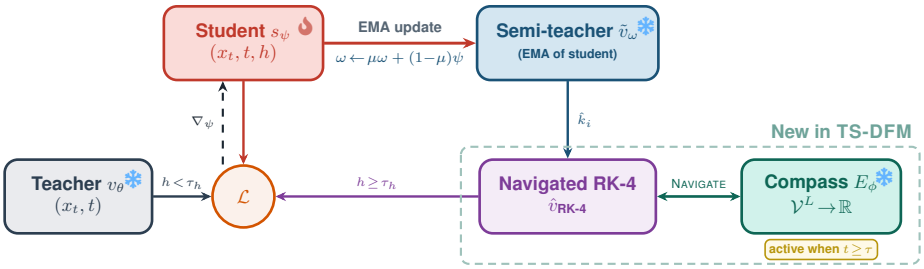


Figure 3: **TS-DFM training overview.** For small steps, the frozen teacher provides the target directly. For large steps, the semi-teacher constructs targets via RK-4 with each midpoint guided by **energy compass** E_ϕ. TS-DFM-specific components are in the dashed box; the rest inherits from FS-DFM.

4 Method: navigation shaping

We propose Trajectory-Shaped Discrete Flow Matching (TS-DFM), a framework that builds on FS-DFM but replaces the blind stochastic jumps with guided navigation. Figure 3 illustrates TS-DFM's training pipeline and the three new components: an energy compass, a Sequence-to-Token navigation procedure, and a time threshold that controls when guidance activates.

4.1 The energy compass

The core tool for navigation is a scalar energy model $E_\phi(x) : \mathcal{V}^L \to \mathbb{R}$ that evaluates the quality of a token sequence without time conditioning—it scores the raw token sequence only. Sequence quality is an intrinsic property of the content. A partially revealed sequence with coherent tokens should score well regardless of whether it appears at t=0.3 or t=0.7.

The compass must distinguish subtle quality differences between partially revealed flow states rather than scoring clean text. The standard diffusion corruption strategy of random token replacement produces trivial negatives that do not represent the decisions that appear during trajectory navigation. To obtain an effective energy compass E_ϕ, we train with a noise-contrastive objective over *generation-aware negatives* to assign lower energy to real flow states.

We use *velocity near-miss* to construct negative examples. Given a positive x_t, we step backward along the flow to create x_{t-h}, run the frozen teacher velocity model forward to reconstruct $\hat{x}_t$, and use $\hat{x}_t$ as a negative. The result competes directly with the positive—both are plausible sequences at time t—but $\hat{x}_t$ contains the teacher's reconstruction errors at the unrevealed positions. These are the kind of subtle errors the compass must detect when comparing CTMC jump candidates during trajectory construction. The total training loss is:

$$\mathcal{L}_{\text{energy}} = \mathcal{L}_{\text{NCE}} + \lambda_{\text{reg}} \cdot \mathcal{L}_{\text{reg}} + \lambda_{\text{order}} \cdot \mathcal{L}_{\text{order}}, \qquad \lambda_{\text{reg}} = \lambda_{\text{order}} = 1.0 \tag{5}$$

where $\mathcal{L}_{\text{NCE}}$ trains the compass to assign lower energy to real flow states than corruptions, $\mathcal{L}_{\text{reg}}$ penalizes energy-scale drift, and $\mathcal{L}_{\text{order}}$ enforces the time-ordering constraint. Appendix A contains the complete training details, additional negative strategies, and a time-ordering loss that enforces monotonically decreasing energy as t increases for each sample x_t. On held-out FineWeb-Edu, the trained compass distinguishes real flow states from generation-relevant corruptions with 98.5%–99.8% accuracy and exhibits perfectly monotone bin-level energy as t increases, for both source distributions (Appendix B).

4.2 Sequence-to-Token navigation

The compass provides *sequence-level* scores, but trajectory quality depends on *token-level* decisions. Scoring entire candidates is affordable but coarse—a sequence can score well while containing poor tokens. Factorizing guidance to individual positions would require $|\mathcal{V}| \times L$ energy evaluations per midpoint, which is intractable. We address this with NAVIGATE, a guided two-phase approach:

1. **Sequence phase** (sequence-level). Generate K candidate next states via temperature-diverse CTMC transitions from the current state. Evaluate them in a single batched forward pass

through E_ϕ and select the lowest-energy candidate. This ensures the trajectory enters a globally low-energy region of sequence space.

2. **Token phase** (token-level). Refine the selected candidate at individual positions using the velocity network's own confidence as a *free* proxy for per-token quality—the logits are already computed from the RK-4 velocity evaluation. Positions where the selected token disagrees with a high-confidence velocity prediction are *candidates* for replacement, modulated by an adaptive guidance coefficient that decays with t. A single energy evaluation on the refined result serves as a safeguard: the refinement is accepted only if it does not increase energy beyond a tolerance ϵ_{safe}.

The sequence phase ensures global trajectory coherence. The token phase ensures local token-level accuracy. Together they cover both granularities without requiring per-token energy evaluation. One cannot do without the other: sequence selection cannot fix individual bad tokens hidden in a good sequence, and token refinement cannot recover from selecting the wrong candidate in the first place. Candidate diversity is sufficient for selection to matter: at $t{=}\tau{=}0.2$, K independent CTMC jumps from a shared midpoint differ at $\sim$147 of 1 024 positions (14%) on average (Appendix E.2). The safeguard accepts $\geq 94.7\%$ of refinements, and rejections would on average have raised energy (Appendix D.2). An inference-time ablation confirms complementarity: at 64 steps the full policy (38.0 PPL) beats sequence-only (52.0) and token-only (47.6) by $\geq 20\%$ (Appendix D.1). Appendices C and D contain complete details and ablations.

Replacing each blind jump with NAVIGATE transforms the standard RK-4 estimation into a navigated one. The semi-teacher $\tilde{v}_\omega$ computes velocity logits as before; the only change is how midpoint states are constructed from those logits. We denote navigated velocities with hats ($\hat{k}_i$) and quality-checked midpoints with asterisks ($x^*_{\text{mid},i}$) to distinguish them from the blind equivalents:

$$\hat{k}_1 = \tilde{v}_\omega(x_t,\, t,\, \tfrac{h}{2}), \qquad x^*_{\text{mid},1} = \textsc{Navigate}(x_t,\, \hat{k}_1,\, t,\, \tfrac{h}{2},\, E_\phi), \tag{6}$$

$$\hat{k}_2 = \tilde{v}_\omega(x^*_{\text{mid},1},\, t{+}\tfrac{h}{2},\, \tfrac{h}{2}), \qquad x^*_{\text{mid},2} = \textsc{Navigate}(x^*_{\text{mid},1},\, \hat{k}_2,\, t{+}\tfrac{h}{2},\, \tfrac{h}{2},\, E_\phi), \tag{7}$$

$$\hat{k}_3 = \tilde{v}_\omega(x^*_{\text{mid},2},\, t{+}\tfrac{h}{2},\, \tfrac{h}{2}), \qquad x^*_{\text{mid},3} = \textsc{Navigate}(x^*_{\text{mid},2},\, \hat{k}_3,\, t{+}\tfrac{h}{2},\, \tfrac{h}{2},\, E_\phi), \tag{8}$$

$$\hat{k}_4 = \tilde{v}_\omega(x^*_{\text{mid},3},\, t{+}h,\, \tfrac{h}{2}), \qquad \hat{v}_{\text{RK-4}} = \tfrac{1}{6}(\hat{k}_1 + 2\hat{k}_2 + 2\hat{k}_3 + \hat{k}_4). \tag{9}$$

Compared with the standard chain where blind jumps produced midpoints with compounding errors, navigated jumps produce midpoints with quality-checked transitions. The advantage compounds multiplicatively: if each guided jump reduces error by a factor $\rho < 1$ relative to a blind jump, the overall target improves by $\sim\rho^3$ across the three midpoints.

4.3 Time-threshold activation

The compass needs signal to navigate. At low flow time t, only a small fraction $\alpha(t)$ of positions carry meaningful tokens—the sequence lies far from the manifold of natural language, and all candidates appear similarly uninformative. Activating the compass here would add computational cost without benefit, and could introduce bias by selecting on noise. We therefore activate navigation only above a time threshold τ:

$$x'_{\text{mid}} = \begin{cases} \textsc{Jump}(x_t, k, t, h) & \text{if } t < \tau \quad \text{(standard blind jump)}, \\ \textsc{Navigate}(x_t, k, t, h, E_\phi) & \text{if } t \geq \tau \quad \text{(compass-guided selection)}. \end{cases} \tag{10}$$

In the source-dominated regime ($t < \tau$), the trajectory explores freely. In the signal-rich regime ($t \geq \tau$), linguistic structure—local coherence, grammatical dependencies, topical consistency—has emerged sufficiently for the compass to make informed selections, and navigation takes over.

Within a single RK-4 estimation, different midpoints may cross the threshold naturally. For example, with $t{=}0.3$, $h{=}0.25$, $\tau{=}0.5$: the first midpoint at $t{=}0.3$ is unguided, while midpoints at $t_{\text{mid}}{=}0.55$ receive navigation. The trajectory automatically transitions from exploration to guided navigation as the sequence approaches the data manifold. As shown in Figure 3, the student's objective is unchanged from FS-DFM distillation—the same KL divergence, but against the navigated target: $\mathcal{L} = \text{KL}(s_\psi(x_t, t, h) \| \hat{v}_{\text{RK-4}})$, where $\hat{v}_{\text{RK-4}}$ comes from Equation (9). The student model, training loop, and inference procedure are identical to FS-DFM; the only modification is how midpoint states are constructed during RK-4 target estimation. At inference, the compass is not used—all quality improvements are encoded in the student's weights.

5 Experiments

Table 1: **Main results across source distributions** for 1 024-token unconditional generation. **Green** cells surpass the 1 024-step DFM teacher. Sampling variance at fixed checkpoints (5 decoding seeds) is in Appendix E.6.

	DFM (Teacher)				FS-DFM				**TS-DFM (DFM init)**				**TS-DFM (FS-DFM init)**			
Steps	Ent	$G^2\downarrow$	$L^2\downarrow$	$L^3\downarrow$	Ent	$G^2\downarrow$	$L^2\downarrow$	$L^3\downarrow$	Ent	$G^2\downarrow$	$L^2\downarrow$	$L^3\downarrow$	Ent	$G^2\downarrow$	$L^2\downarrow$	$L^3\downarrow$
Uniform source																
4	8.1	749.9	413.3	772.9	7.0	169.7	99.8	188.9	7.1	121.2	73.4	120.6	7.0	**100.3**	**57.1**	**98.5**
8	8.1	328.9	186.7	340.4	7.4	87.6	52.0	98.5	7.3	79.0	47.7	79.9	7.2	**56.1**	**32.2**	**54.6**
16	8.0	176.9	100.8	177.2	7.4	73.2	45.1	81.6	7.6	70.1	41.5	69.5	7.4	**47.2**	**27.1**	**46.4**
32	8.0	117.7	67.0	114.1	7.7	67.8	39.0	69.5	7.8	71.0	41.0	69.7	7.5	**46.7**	**28.5**	**47.4**
1 024	7.8	82.8	47.8	78.9		—				—				—		
Mask source																
4	8.5	1473.3	561.2	1441.0	8.2	1015.7	367.1	943.5	7.1	**157.4**	**132.2**	**149.8**	7.7	386.7	201.2	363.4
8	8.4	761.3	307.0	754.9	8.2	516.6	204.9	492.4	7.4	**91.4**	**73.1**	**91.3**	7.9	188.7	94.8	177.4
16	8.4	359.3	152.7	348.9	8.2	284.7	122.7	271.6	7.6	**61.9**	**45.6**	**61.1**	8.0	109.9	55.2	101.9
32	8.3	190.3	87.0	184.2	8.2	189.0	86.3	182.5	7.7	**52.4**	**37.0**	**51.3**	8.0	91.4	45.4	84.5
1 024	8.1	93.6	45.7	87.9		—				—				—		

Metrics: Ent = generation entropy; G^2 = GPT-2 Large (124M), L^2 = LLaMA-2 (7B), L^3 = LLaMA-3 (8B). All few-step methods use the same 170M-parameter student at inference. Shaded 1 024-step row shows the DFM teacher reference.

5.1 Setup

The DFM teacher and FS-DFM/TS-DFM students are 170M-parameter DDiT transformers [Gat et al., 2024]. The energy compass E_ϕ is a 6-block transformer pretrained on FineWeb-Edu [Penedo et al., 2024] with noise-contrastive estimation and generation-aware negatives, then frozen throughout student training. Details of the energy compass training and evaluation are in Appendices A and B. We evaluate across two source distributions: (i) **Uniform**: tokens drawn uniformly over the GPT-2 vocabulary ($|\mathcal{V}|$=50 257); (ii) **Mask**: all positions initialized with a dedicated `[MASK]` token.

Table 2: **Effect of time threshold τ.** Lower τ activates navigation earlier, improving perplexity but risking diversity loss (*entropy below 6.5). Green rows : default τ=0.2.

	DFM init				**FS-DFM init**			
	8 steps		16 steps		8 steps		16 steps	
	Ent	$G^2\downarrow$	Ent	$G^2\downarrow$	Ent	$G^2\downarrow$	Ent	$G^2\downarrow$
Uniform source								
$\tau = 0.02$	6.4*	54.7	7.2	56.3	6.4*	43.5	6.8	**35.8**
$\tau = 0.2$	7.3	79.0	7.6	70.1	7.2	**56.1**	7.4	47.2
$\tau = 0.4$	7.5	87.9	7.6	69.6	7.2	71.5	7.4	53.9
Mask source								
$\tau = 0.02$	6.2*	92.9	7.6	89.5	7.4	137.0	7.6	83.1
$\tau = 0.2$	7.4	**91.4**	7.6	**61.9**	7.9	188.7	8.0	109.9
$\tau = 0.4$	7.8	139.4	7.8	78.8	8.0	254.2	8.1	118.9

All models are trained on FineWeb-Edu [Penedo et al., 2024]. Each source distribution has its own DFM teacher (65.5B tokens) and FS-DFM student (13.2B additional fine-tuning tokens), both serving as potential initializations for TS-DFM (6.4B additional shaping tokens). Default configuration: K=5 candidates per midpoint, time threshold τ=0.2. Appendix C.7 discusses the training-time overhead of navigation shaping relative to standard FS-DFM.

We report generation entropy (Ent) and perplexity (PPL, lower is better) evaluated by three reference language models: GPT-2 Large (124M, denoted G^2) [Radford et al., 2019], LLaMA-2 7B (denoted L^2) [Touvron et al., 2023], and LLaMA-3 8B (denoted L^3) [Grattafiori et al., 2024]. All evaluations use 1 024-token unconditional generation on WikiText-103 [Merity et al., 2017] at 4, 8, 16, and 32 sampling steps, with the 1 024-step DFM teacher as a quality reference.

5.2 Main results

Table 1 presents the central result: across both source distributions, trajectory shaping produces a student that substantially outperforms the FS-DFM baseline and surpasses the unguided 1 024-step DFM teacher at just 8 inference steps.

Table 3: **Compass at inference vs. at training** (uniform source, 170M, GPT-2 PPL ↓). Latency is per $1\,024$-token generation at the lowest step count shown. Teacher rows are a separate evaluation with 64 samples per cell. TS-DFM rows are 5-seed means. Values therefore differ slightly from Table 1 (details in Appendices E.7 and E.8).

Sampler	**Compass**	**8 steps**	**16 steps**	$1\,024$ **steps**	**Lat. (s)**
Teacher, blind	—	—	—	83.98	18.9
Teacher + NAVIGATE (every step)	inference	—	—	49.03	186
FS-DFM, blind	—	89.30	63.24	—	0.18
FS-DFM + best-of-5 rerank	inference	80.41	63.49	—	0.94
FS-DFM + final-step selection	inference	81.36	64.67	—	0.38
FS-DFM + full NAVIGATE	inference	80.18	57.36	—	1.27
TS-DFM (FS-DFM init), blind	training	56.97 ± 1.48	45.73 ± 1.27	—	0.18

Uniform source. With FS-DFM initialization, TS-DFM achieves 56.1 GPT-2 PPL at 8 steps—a 36% reduction over the FS-DFM baseline (87.6) and 32% below the $1\,024$-step teacher (82.8). The improvement is consistent across all three evaluators and persists at 16 and 32 steps, where TS-DFM reaches 47.2 and 46.7, respectively. Even with DFM initialization (no prior distillation), TS-DFM at 8 steps (79.0) already beats the $1\,024$-step teacher, confirming that the gains come from trajectory shaping rather than the initialization alone.

Mask source. For the mask source, DFM initialization proves more effective than FS-DFM initialization. TS-DFM with DFM init achieves 91.4 GPT-2 PPL at 8 steps and 61.9 at 16 steps—both surpassing the $1\,024$-step mask teacher (93.6). The mask source is notably more challenging for few-step generation: the FS-DFM baseline achieves only 516.6 PPL at 8 steps, making TS-DFM's $5.6\times$ improvement especially notable. This directly addresses a key limitation of FS-DFM, which reported strong results only for the uniform source and noted that mask-source distillation remained an open problem. TS-DFM demonstrates that trajectory shaping makes few-step distillation effective for the mask source as well.

Training-time shaping vs. inference-time guidance. To test whether the gain comes from trajectory shaping or from the compass itself, Table 3 gives the same frozen compass to fixed samplers at inference. Reranking, final-step selection and full NAVIGATE all leave the unshaped FS-DFM student near 80 PPL at 8 steps, whereas blind TS-DFM reaches 56.97 at the same inference cost. Guided at every step, the $1\,024$-step teacher improves from 83.98 to 49.03, yet the blind 16-step student is lower still (45.73), at $\sim 0.2\%$ of its inference cost. What separates these rows is when the compass acts, not access to it (Appendices E.7 and E.8).

Effect of time threshold τ. The time threshold τ controls when navigation shaping activates. We train TS-DFM students with $\tau \in \{0.02, 0.2, 0.4\}$ for both source distributions, each with two initializations: from the DFM teacher (DFM init) and from the FS-DFM student (FS-DFM init). Table 2 reports GPT-2 perplexity and generative entropy at 8 and 16 steps. First, τ=0.2 offers the best quality–diversity trade-off for both sources: it achieves strong perplexity while maintaining entropy above 7.0. The lowest threshold (τ=0.02) yields better raw perplexity but drops entropy below 6.5, indicating diversity collapse. Second, the two sources prefer different initializations: uniform benefits from FS-DFM init (56.1 vs. 79.0 at 8 steps with τ=0.2), while mask benefits from DFM init (91.4 vs. 188.7)—suggesting that the mask-source FS-DFM student has converged to velocity patterns that are harder to redirect with trajectory shaping. We adopt these defaults throughout. Third, the training overhead is modest: $2.0\times$ at τ=0.4 to $2.4\times$ at τ=0.02, with our default τ=0.2 at $2.2\times$. Additional analysis is in Appendices C.7 and E.1.

Choice of K. The number of candidates K in the sequence phase controls the quality–cost trade-off. In inference-time ablations (Appendix D.3), introducing any candidate selection (K=2) provides the largest single improvement (23% PPL reduction). Returns diminish quickly: K=5 reaches 43% reduction, while K=8 provides no further benefit. We use K=5 throughout. An inference-time sweep of β_0 and

Table 4: **Training overhead**.

Method	**s/step**	**vs. FS-DFM**
FS-DFM	1.37	$1.0\times$
TS-DFM (τ=0.4)	2.79	$2.0\times$
TS-DFM (τ=0.2)	3.07	$2.2\times$
TS-DFM (τ=0.02)	3.31	$2.4\times$

Table 5: **Few-step generation comparison at 8 and 16 steps.** Methods span four families: masked diffusion (MDLM, SEDD, ReMDM variants), multi-round distillation (SDTT), consistency distillation (Duo), and flow-matching distillation (FS-DFM, TS-DFM). For FS-DFM and TS-DFM, the first 65.5B tokens correspond to DFM pretraining. Additional tokens (‡) denote fine-tuning. †Reproduced using official code. Sampling variance at fixed checkpoints (5 decoding seeds) is in Appendix E.6.

			8 steps		16 steps	
Method	**Size (B)**	**Tokens (B)**	PPL↓	Ent	PPL↓	Ent
ReMDM-loop†	0.17	262	1319.2	8.4	413.6	8.3
ReMDM-cap†	0.17	262	546.8	8.3	181.8	8.2
ReMDM-conf†	0.17	262	545.2	8.3	181.3	8.2
ReMDM-rescale†	0.17	262	555.8	8.4	181.1	8.2
MDLM-OWT†	0.17	262	815.6	8.5	304.3	8.3
SEDD†	0.32	–	683.7	8.4	305.6	8.3
SDTT (1 round)†	0.86	–	567.8	8.4	215.5	8.2
SDTT (3 rounds)†	0.86	–	305.4	8.2	128.2	8.1
SDTT (5 rounds)†	0.86	–	184.8	8.0	84.1	8.0
SDTT (7 rounds)†	0.86	–	105.8	7.8	53.6	7.7
Duo†	0.17	524	70.6	7.6	53.6	7.7
FS-DFM†	0.17	$65.5{+}13.2^{\ddagger}$	87.6	7.4	73.2	7.4
TS-DFM	0.17	$78.7{+}6.4^{\ddagger}$	**56.1**	7.2	**47.2**	7.4

ϵ_{safe} on the fixed FS-DFM student shows a narrow spread, with every setting better than blind sampling (Appendix E.9). This measures the sensitivity of the operator, not of the shaped student.

Comparison with state-of-the-art. Table 5 compares TS-DFM against state-of-the-art few-step discrete generation methods at 8 and 16 steps. TS-DFM achieves the best perplexity at both step counts. At 8 steps it reaches 56.1 GPT-2 PPL—19% lower than Duo (70.6, trained on $6\times$ more tokens) and 46% below SDTT after 7 rounds (105.8, using a $5\times$ larger model). At 16 steps the gap widens: 47.2 vs. 53.6 for both Duo and SDTT.

Training cost. Table 4 reports wall-clock training overhead measured on a single GPU. At the default $\tau{=}0.2$, TS-DFM adds $2.2\times$ overhead per step relative to FS-DFM, ranging from $2.0\times$ ($\tau{=}0.4$) to $2.4\times$ ($\tau{=}0.02$). This is modestly higher than the theoretical $1.7\times$ FLOP estimate (Appendix C.7), reflecting practical costs: generating K candidate CTMC jumps, memory allocation for batched energy scoring, and Python-level control flow. The K candidate energy evaluations are batched into a single GPU kernel, so overhead scales sublinearly with K. Higher thresholds reduce cost by activating guidance at fewer midpoints: $\tau{=}0.4$ guides only midpoints where over 40% of tokens are revealed. The narrow spread (2.0–$2.4\times$) indicates that the fixed cost of maintaining the energy compass dominates the variable cost of per-midpoint scoring. This is a one-time training cost; inference is unchanged.

Scaling to 1.3B parameters. We repeat the uniform-source comparison at 1.3B—$7.6\times$ our main model—guided by the same 90M energy compass (Table 6; Appendix E.3). (1) The shaped student beats its unguided $1\,024$-step teacher at just 8 steps: TS-DFM reaches 48.0 GPT-2 Large PPL versus the teacher's 57.5—a $128\times$ inference speedup with *improved* quality. (2) At 8 steps, the gap over FS-DFM remains at least as large at 1.3B: TS-DFM cuts FS-DFM's PPL by 41% at 1.3B (48.0 vs. 81.5) versus 36% at 170M (56.1 vs. 87.6). At 16 and 32 steps the reductions are similar at both scales (36% at 170M vs. 33% at 1.3B at 16 steps, 31% vs. 30% at 32 steps; Appendix E.3). (3) Relative cost shrinks at scale. The compass-to-student ratio drops from $\sim$1:2 at 170M to $\sim$1:14 at 1.3B, so navigation's (fixed-size) FLOP overhead becomes proportionally cheaper as students grow. Beyond unconditional generation, a transfer study on GSM8K and SVAMP at 1.3B scale (Appendix E.5) applies the same mechanism to task-fine-tuned math reasoning. At the 4- and 8-step budgets we evaluate, TS-DFM matches or exceeds FS-DFM, by small margins that we read as directionally consistent.

Table 6: **Scaling to 1.3B parameters (uniform source).** Perplexity (PPL ↓) and generation entropy (Ent) for 1 024-token unconditional generation at 4, 8, 16, and 32 sampling steps, with the 1 024-step DFM teacher as the quality reference. **Green** cells surpass the 1 024-step DFM teacher.

	DFM				FS-DFM				TS-DFM			
Steps	Ent	G^2 ↓	L^2 ↓	L^3 ↓	Ent	G^2 ↓	L^2 ↓	L^3 ↓	Ent	G^2 ↓	L^2 ↓	L^3 ↓
4	8.1	913.6	458.6	915.2	7.2	188.6	80.8	170.3	7.0	**98.9**	**51.0**	**92.4**
8	8.1	320.7	169.6	320.6	7.4	81.5	38.2	73.5	7.3	**48.0**	**26.6**	**46.7**
16	8.0	142.3	77.4	138.8	7.5	49.5	23.9	44.5	7.5	**33.1**	**18.8**	**32.0**
32	7.9	91.9	51.3	88.5	7.7	44.9	22.8	41.0	7.6	**31.5**	**18.1**	**30.7**
1 024	7.9	57.5	32.1	54.5	—				—			

6 Conclusion

We introduced TS-DFM, built on the observation that the performance ceiling in few-step discrete flow distillation is set by the training trajectories, not the student. Blind stochastic jumps at RK-4 midpoints compound errors that the student faithfully imitates. TS-DFM replaces these jumps with guided navigation via a lightweight energy compass and a two-phase Sequence-to-Token policy, at training time only. At 170M parameters, TS-DFM is state-of-the-art among few-step discrete generation methods on both uniform and mask sources. At 1.3B with the uniform source, guided by the same lightweight compass, the shaped student beats its unguided 1 024-step teacher at just 8 steps, a $128\times$ speedup with *improved* quality. At this step budget, the gap over the FS-DFM baseline remains at least as large at 1.3B as at 170M ($36\%\rightarrow41\%$ PPL reduction).

Limitations. The compass is trained once and frozen. As the student improves, its flow-state distribution shifts, potentially reducing compass effectiveness (iterative refinement could help but risks co-adaptation). In the worst-case run, the frozen compass shows no measurable degradation on our drift metrics over the full run (Appendix E.10), but we have not run the compass-refresh or EMA-recompute control. The activation threshold τ is fixed throughout training; adaptive or annealed schedules remain unexplored. Our evaluation relies mainly on generative perplexity, which coherence–diversity trade-offs can game. We report entropy with every perplexity and score under three evaluators, but we include no human evaluation. The scaling evidence covers two model sizes, with only the uniform source at 1.3B, so a third scale and the mask source at 1.3B remain future work. Downstream evidence is limited to GSM8K and SVAMP. Instruction-tuned students at larger scale, where benchmarks such as MATH, HumanEval and IFEval become meaningful, are also left for future work.

Acknowledgments and Disclosure of Funding

We thank Barry Theobald and Zhe Gan for feedback on early versions of this paper.

References

Marianne Arriola, Subham Sekhar Sahoo, Aaron Gokaslan, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Justin T Chiu, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tyEyYT267x.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *The Thirty-fifth Annual Conference on Neural Information Processing Systems*, pages 17981–17993, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/958c530554f78bcd8e97125b70e6973d-Paper.pdf.

Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 843–852, 2023.

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 *The Forty-first International Conference on Machine Learning*, pages 5453–5512, 2024. URL https://proceedings.mlr.press/v235/campbell24a.html.

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

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

Yuntian Deng, Anton Bakhtin, Myle Ott, Arthur Szlam, and Marc'Aurelio Ranzato. Residual energy-based models for text generation. In *The Eighth International Conference on Learning Representations*, 2020.

Justin Deschenaux and Caglar Gulcehre. Beyond autoregression: Fast LLMs via self-distillation through time. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=uZ5K4HeNwd.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *The Thirty-fifth Annual Conference on Neural Information Processing Systems*, pages 8780–8794, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/49ad23d1ec9fa4bd8d77d02681df5cfa-Paper.pdf.

Weihua Du, Yiming Yang, and Sean Welleck. Optimizing temperature for language models with multi-sample inference. In *The Forty-second International Conference on Machine Learning*, pages 14648–14668, 2025. URL https://proceedings.mlr.press/v267/du25f.html.

Yilun Du, Conor Durkan, Robin Strudel, Joshua B. Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Sussman Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and MCMC. In *The Fortieth International Conference on Machine Learning*, pages 8489–8510, 2023. URL https://proceedings.mlr.press/v202/du23a.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. URL https://openreview.net/forum?id=OlzB6LnXcS.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, pages 133345–133385, 2024. doi: 10.52202/079017-4239. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/f0d629a734b56a642701bba7bc8bb3ed-Paper-Conference.pdf.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The Llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Zhengfu He, Tianxiang Sun, Qiong Tang, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. DiffusionBERT: Improving generative masked language models with diffusion models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4521–4534, 2023. doi: 10.18653/v1/2023.acl-long.248. URL https://aclanthology.org/2023.acl-long.248/.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Muhammad Khalifa, Hady Elsahar, and Marc Dymetman. A distributional approach to controlled text generation. In *The Ninth International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=jWkw45-9AbL.

Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu Jie Huang. A tutorial on energy-based learning. In *Predicting Structured Data*. MIT Press, 2006.

Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B. Tenenbaum. Compositional visual generation with composable diffusion models. In *Computer Vision – ECCV 2022*, pages 423–439, 2022.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *The Forty-first International Conference on Machine Learning*, 2024.

Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14297–14306, 2023. doi: 10.1109/CVPR52729.2023.01374.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *The Fifth International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

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*, 2026. URL https://openreview.net/forum?id=ue1zFeD275.

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 *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=KnqiC0znVF.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.168. URL https://aclanthology.org/2021.naacl-main.168/.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, pages 30811–30849, 2024. doi: 10.52202/079017-0970. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/370df50ccfdf8bde18f8f9c2d9151bda-Paper-Datasets_and_Benchmarks_Track.pdf.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. MAUVE: Measuring the gap between neural text and human text using divergence frontiers. In *The Thirty-fifth Annual Conference on Neural Information Processing Systems*, pages 4816–4828, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/260c2432a0eecc28ce03c10dadc078a4-Abstract.html.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. OpenAI technical report, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

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 *The Thirty-eighth Annual Conference on Neural Information Processing Systems*,

2024. doi: 10.52202/079017-4135. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/eb0b13cc515724ab8015bc978fdde0ad-Paper-Conference.pdf.

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin T Chiu, and Volodymyr Kuleshov. The diffusion duality. In *The Forty-second International Conference on Machine Learning*, pages 52584–52619, 2025. URL https://proceedings.mlr.press/v267/sahoo25a.html.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *The Tenth International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI.

Yang Song and Diederik P. Kingma. How to train your energy-based models, 2021. URL https://arxiv.org/abs/2101.03288.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *The Fortieth International Conference on Machine Learning*, pages 32211–32252, 2023. URL https://proceedings.mlr.press/v202/song23a.html.

Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, Yuwei Fu, Jing Su, Ge Zhang, Wenhao Huang, Mingxuan Wang, Lin Yan, Xiaoying Jia, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Yonghui Wu, and Hao Zhou. Seed Diffusion: A large-scale diffusion language model with high-speed inference, 2025. URL https://arxiv.org/abs/2508.02193.

Hugo Touvron, Louis Martin, Kevin Stone, et al. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Remasking discrete diffusion models with inference-time scaling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=IJryQAOy0p.

Zhihui Xie, Jiacheng Ye, Lin Zheng, Jiahui Gao, Jingwei Dong, Zirui Wu, Xueliang Zhao, Shansan Gong, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream-Coder 7B: An open diffusion language model for code, 2025. URL https://arxiv.org/abs/2509.01142.

Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. MMaDA: Multimodal large diffusion language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=wczmXLuLGd.

Jiacheng Ye, Shansan Gong, Liheng Chen, Lin Zheng, Jiahui Gao, Han Shi, Chuan Wu, Xin Jiang, Zhenguo Li, Wei Bi, and Lingpeng Kong. Diffusion of thought: Chain-of-thought reasoning in diffusion language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, pages 105345–105374, 2024. doi: 10.52202/079017-3343. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/be30024e7fa2c29cac7a6dafcbb8571f-Paper-Conference.pdf.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7B: Diffusion large language models, 2025. URL https://arxiv.org/abs/2508.15487.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T. Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6613–6623, 2024. doi: 10.1109/CVPR52733.2024.00632.

Shimao Zhang, Yu Bao, and Shujian Huang. EDT: Improving large language models' generation by entropy-based dynamic temperature sampling, 2024. URL https://arxiv.org/abs/2403.14541.

Tunyu Zhang, Xinxi Zhang, Ligong Han, Haizhou Shi, Xiaoxiao He, Zhuowei Li, Hao Wang, Kai Xu, Akash Srivastava, Hao Wang, Vladimir Pavlovic, and Dimitris N. Metaxas. T3D: Few-step diffusion language models via trajectory self-distillation with direct discriminative optimization, 2026. URL https://arxiv.org/abs/2602.12262.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1097–1100, 2018. doi: 10.1145/3209978.3210080. URL https://doi.org/10.1145/3209978.3210080.

Appendix

Appendix Contents

A Energy compass training details **17**

A.1 Architecture 17

A.2 Training objective 17

A.3 Negative strategies 17

A.4 Positive sample construction 19

A.5 Optimization 19

B Energy compass validation **19**

B.1 Time-energy monotonicity 20

B.2 Generation-relevant corruption detection 21

C Sequence-to-Token compass navigation **22**

C.1 Motivation: the granularity dilemma in discrete guidance 22

C.2 Phase 1: sequence exploration via diverse candidate scoring 23

C.3 Phase 2: token refinement via velocity-aligned token correction 24

C.4 Energy safeguard 25

C.5 Full algorithm 26

C.6 Why both phases are necessary 27

C.7 Computational cost and practical considerations 27

D Empirical validation of the navigation policy **28**

D.1 Navigation policy comparison across step counts 28

D.2 Token refinement behavior across step counts 30

D.3 Effect of candidate count K 30

E Additional experiments **31**

E.1 Effect of time threshold τ: additional details 31

E.2 Midpoint diversity analysis 32

E.3 Scaling to larger models 33

E.4 Comparison to diffusion language models in the few-step regime 34

E.5 Transfer to mathematical reasoning at 1.3B scale 35

E.6 Variance across decoding seeds 37

E.7 Energy-guided teacher sampling 38

E.8 Inference-time deployment of the compass 39

E.9 Sensitivity of navigation constants 40

E.10 Compass robustness under student distribution shift 40

F Qualitative examples **41**

F.1 Generated samples 41
F.2 Generation trajectories 43

A Energy compass training details

Section 4.1 describes the energy compass's role, training objective, and key design choices. This section provides the full training specification, including the objective decomposition, negative construction details, source-distribution handling, and optimization configuration.

We train a separate energy compass for each source family: one for the *uniform* source distribution (x_0 sampled uniformly over $\mathcal{V}$) and one for the *mask* source distribution ($x_0 = [\text{MASK}]^L$). The two compasses share identical architecture and training procedure (Tables 7 and 9); they differ only in the source tokens that populate unrevealed positions during positive and negative construction. All validation results in Appendix B are reported for both variants where applicable.

A.1 Architecture

The energy compass is a transformer encoder with attention pooling that maps a token sequence to a scalar energy. It shares the same tokenizer (GPT-2, $|\mathcal{V}|$=50 257) and block size (L=1 024) as the velocity model but is substantially smaller (Table 7). The lack of time conditioning is a deliberate design choice: the compass scores intrinsic sequence quality, not quality-relative-to-timestep. In place of a time embedding, the model uses a learned constant buffer, ensuring the architecture is identical to a time-conditioned variant (for checkpoint compatibility) but receives no time information.

Table 7: Energy compass architecture.

Component	Configuration
Transformer blocks	6
Hidden dimension	768
Attention heads	12
Pooling	Attention (learned query $\rightarrow$ scalar)
Time conditioning	None (sequence only)
Parameters	$\sim$90M

A.2 Training objective

The total loss is a weighted sum of three components, each addressing a distinct training challenge:

$$\mathcal{L}_{\text{energy}} = \mathcal{L}_{\text{NCE}} + \lambda_{\text{reg}} \cdot \mathcal{L}_{\text{reg}} + \lambda_{\text{order}} \cdot \mathcal{L}_{\text{order}}, \qquad \lambda_{\text{reg}} = \lambda_{\text{order}} = 1.0$$

The noise-contrastive loss ($\mathcal{L}_{\text{NCE}}$): for each positive flow state x_t, we generate N_{neg}=12 corrupted negatives $\{\tilde{x}_t^{(j)}\}$ via the strategies in Appendix A.3 and minimize:

$$\mathcal{L}_{\text{NCE}} = -\log \frac{\exp(-E_\phi(x_t))}{\exp(-E_\phi(x_t)) + \sum_{j=1}^{N_{\text{neg}}} \exp(-E_\phi(\tilde{x}_t^{(j)}))}, \tag{11}$$

training the compass to assign lower energy to the real flow state than to any corruption.

Regularization loss ($\mathcal{L}_{\text{reg}}$): prevents energy scale drifts during training—the mean energy can swing by several units between evaluation checkpoints, destabilizing the NCE gradients and collapsing the time-ordering signal. We penalize deviation from zero, which anchors the energy scale without constraining the *relative* ordering between samples:

$$\mathcal{L}_{\text{reg}} = E_\phi(x_t)^2 \tag{12}$$

Time-ordering loss ($\mathcal{L}_{\text{order}}$): The NCE objective teaches within-timestep discrimination (corrupt vs. clean at fixed t), but does not coordinate energy levels *across* timesteps. To enforce monotonically decreasing energy as more tokens are revealed, we construct a less-revealed version $x_{t'}$ at $t' = \max(t - \delta, 0.01)$ where $\delta \sim \text{Uniform}(0.05, 0.5)$, using the same (x_0, x_1) pair, and apply a margin-based hinge loss:

$$\mathcal{L}_{\text{order}} = \max\big(0,\ E_\phi(x_t) - E_\phi(x_{t'}) + \gamma \cdot \delta\big), \tag{13}$$

with a proportional margin γ=0.3. The margin scales with δ so larger time gaps require larger energy differences, providing a smooth gradient signal toward monotonicity.

A.3 Negative strategies

Negatives are constructed by corrupting *revealed positions only*—positions where x_t already carries a target token from x_1. Unrevealed positions retain their source tokens (uniform random tokens or [MASK]), since these are not informative to corrupt.

We use two general categories of negatives, each targeting a distinct failure mode the compass must detect:

- **Generation-realistic negatives** simulate the actual errors that arise during discrete flow matching—teacher reconstruction mistakes (`velocity_step`) and under-revealed flow states (`time_downstep`). These are the hardest negatives and most directly represent the comparisons the compass makes during trajectory navigation.
- **Token-corruption negatives** degrade revealed tokens with wrong or repeated substitutions (`random_replace`, `frequency_replace`, `token_repeat`). These are cheaper to construct and teach the compass to reject content-level corruptions that are unrelated to the flow dynamics.

Each negative is drawn from one of the five strategies in Table 8, sampled in proportion to the probabilities listed. Generation-realistic negatives together account for $\sim 45\%$ of training negatives. Within this group, `velocity_step` produces the most realistic negatives but is sampled less often (9.1%) to limit the cost of the frozen teacher forward pass it requires; all other strategies incur no additional model inference.

Table 8: Negative strategies for energy model training, grouped by category. Rate ranges only apply to token-level strategies.

Strategy	Prob.	Rate range	Description
Generation-realistic negatives			
`time_downstep`	36.4%	$\delta \in [0.025, 0.5]$	*Temporal.* Construct $x_{t'}$ at lower $t' < t$ from same (x_0, x_1). A real but less-revealed flow state.
`velocity_step`	9.1%	—	*Near-miss.* Step backward then forward through the teacher to produce realistic reconstruction-error negatives (see below).
Token-corruption negatives			
`random_replace`	18.2%	[0.05, 0.25]	*Wrong tokens.* Replace revealed tokens with uniform random vocabulary items.
`frequency_replace`	18.2%	[0.03, 0.20]	*Wrong tokens.* Replace with tokens from the same unigram frequency bin—cannot be detected by frequency statistics alone.
`token_repeat`	18.2%	[0.05, 0.50]	*Repetition.* Repeat a single token across revealed positions. Simulates degenerate repetition.

A.3.1 Velocity near-miss construction

The `velocity_step` strategy produces the most realistic negatives and deserves additional detail. Given a positive x_t at time t:

1. Sample a step size h from a set of discrete options spanning $[\sim 0.001, 0.0625]$.
2. Construct x_{t-h} by probabilistically unrevealing positions: each revealed position in x_t is kept with probability $\alpha(t-h)/\alpha(t)$ and replaced with a source token otherwise (uniform random for the uniform compass, [MASK] for the mask compass). This yields a sequence with $\sim \alpha(t-h)$ fraction revealed—consistent with the flow dynamics at time $t-h$.
3. Run the frozen teacher velocity model on $(x_{t-h}, t-h)$ to obtain logits, then take a forward Euler step back to time t, producing $\hat{x}_t$.
4. Return $\hat{x}_t$ as the negative, paired with the positive x_t at the same time t.

The resulting negative competes directly with the positive: both are plausible sequences at time t, but $\hat{x}_t$ contains the teacher's reconstruction errors at the positions that were unrevealed in step 2. These are precisely the kind of errors the compass must detect when comparing CTMC jump candidates during trajectory construction.

Teacher velocity model. The `velocity_step` strategy requires a frozen teacher velocity model to generate realistic near-miss negatives. Each compass variant uses its matching DFM teacher (170M parameters)—the uniform-trained teacher for the uniform compass, the mask-trained teacher for the mask compass—identical to the teacher used in TS-DFM and FS-DFM distillation. The teacher is loaded in eval mode with gradients disabled; its forward-pass cost is incurred only when `velocity_step` is sampled for a given batch element.

A.3.2 Corruption rate sampling

For token-level strategies (`random_replace`, `frequency_replace`, `token_repeat`), the corruption rate is sampled from a Beta(1,3) distribution over $[\text{rate}_{\min}, \text{rate}_{\max}]$. Beta(1,3) has mean 0.25 of its range, biasing toward low corruption rates and thus toward harder negatives where only a few tokens differ from the positive. A floor of 10% of revealed positions ensures that even at high t—where few positions remain revealed to corrupt—negatives still differ from positives enough to provide a meaningful gradient signal.

A.4 Positive sample construction

Positive samples are flow states x_t constructed via the standard interpolation path (Equation (1)) from (x_0, x_1) pairs, where x_1 is a real text sequence from FineWeb-Edu [Penedo et al., 2024]. The source x_0 is drawn from the corresponding source distribution: uniform random tokens over $\mathcal{V}$ for the uniform compass, or $[\text{MASK}]^L$ for the mask compass. The time t is sampled uniformly from $(0, 1)$.

To anchor the energy scale, 10% of training positives use the clean target x_1 directly (at t=1) rather than a path-interpolated x_t. This ensures the compass learns that fully-revealed, real text should receive the lowest energy—providing a stable reference point for the energy landscape.

A.5 Optimization

The learning rate and warmup schedule were tuned to keep the energy scale stable during early training, which the time-ordering loss (Appendix A.2) depends on. With a higher peak LR of 3×10^{-4}, the mean energy drifted by more than one unit across the warmup ramp—large enough to overwhelm the margin in $\mathcal{L}_{\text{order}}$ and collapse the time-ordering signal. Lowering the peak to 1×10^{-4} with $5\,000$ warmup steps eliminated this instability.

Table 9: Energy compass optimization.

Parameter	Value
Optimizer	AdamW (β_1=0.9, β_2=0.999)
Peak learning rate	1×10^{-4}
LR schedule	Cosine decay (min ratio 0.1)
Warmup steps	$5\,000$
Weight decay	0.01
Gradient clipping	1.0
Batch size	64
Precision	bf16
GPUs	8
Training data	FineWeb-Edu
Validation data	WikiText-103

Checkpoint selection. The energy compass must satisfy two properties: within-timestep discrimination (high NCE accuracy per strategy, Appendix B.2) and across-timestep monotonicity (smooth energy decrease with t, Appendix B.1). These can conflict: stronger NCE gradients widen per-timestep energy gaps without coordinating absolute energy levels across timesteps. We select the best checkpoint via a composite score that sums the overall NCE accuracy across strategies and the fraction of consecutive time-bin pairs with correctly decreasing energy (smooth_pct):

$$\text{score} = \text{strategy_acc} + \text{smooth_pct} \tag{14}$$

Equal weighting reflects that both properties are necessary for reliable navigation: high discrimination without monotonicity produces a compass that ranks well within a timestep but assigns inconsistent absolute energies across timesteps, while monotonicity without discrimination fails to distinguish subtle quality differences among candidates.

B Energy compass validation

Before using the energy compass to guide trajectory construction, we must verify two properties: (1) that it assigns monotonically lower energy to sequences with more revealed tokens (reflecting

increasing quality along the flow path), and (2) that it assigns lower energy to real flow states than to corrupted versions exhibiting the specific failure modes that arise during discrete flow matching generation—not just arbitrary noise. We validate both properties on held-out sequences from FineWeb-Edu using the frozen, pretrained energy model. Since we train a separate compass for each source family (Appendix A), below we report results for both the *uniform* and *mask* variants.

B.1 Time-energy monotonicity

In discrete flow matching, a sequence x_t at time t has approximately $\alpha(t)$ fraction of its positions revealed (matching the target x_1); the rest remain source noise. As t increases, $\alpha(t)$ grows and the sequence becomes intrinsically higher-quality text. For the energy compass to be a reliable navigator, its scores must reflect this ordering: $E_\phi(x_t)$ should decrease monotonically with t. Formally, for any two flow states constructed from the same (x_0, x_1) pair at times $t > t'$:

$$E_\phi(x_t) < E_\phi(x_{t'}) \qquad \text{(higher } t = \text{more revealed} = \text{lower energy)}. \tag{15}$$

To measure how consistently the trained compass satisfies this criterion on held-out data, we take 200 held-out (x_0, x_1) pairs and construct flow states at 20 randomly sampled timesteps and the clean target (t=1.0), producing 4 200 total evaluations per source distribution. We compute $E_\phi(x_t)$ and report binned mean energy, monotonicity, rank correlation, and aggregate metrics in Tables 10 and 11.

Table 10: **Time-energy monotonicity.** Mean energy $E_\phi(x_t)$ per time bin; both source distributions.

		Time bin									
Src		[0,.1)	[.1,.2)	[.2,.3)	[.3,.4)	[.4,.5)	[.5,.6)	[.6,.7)	[.7,.8)	[.8,.9)	[.9,1)	t=1
Uni.	$\bar{E}_\phi$	−0.12	−0.57	−0.75	−0.90	−1.02	−1.10	−1.17	−1.22	−1.29	−1.34	−1.45
	std	0.50	0.07	0.06	0.06	0.08	0.07	0.12	0.21	0.21	0.38	0.38
Mask	$\bar{E}_\phi$	−1.94	−2.24	−2.35	−2.43	−2.47	−2.55	−2.59	−2.66	−2.68	−2.75	−3.20
	std	0.34	0.13	0.19	0.21	0.34	0.31	0.42	0.38	0.52	0.55	0.61

The binned means are *perfectly* monotone for both source distributions: all 10 consecutive bin pairs show strictly decreasing energy. At the sample level, the uniform compass achieves 97.2% smoothness with only 44 of 200 samples exhibiting any violation, while the mask compass achieves 94.0% with 74 violating samples. The mask compass assigns lower (more negative) energies overall because [MASK] tokens provide a uniform, low-entropy background at unrevealed positions, making partially revealed sequences more structured. This compresses the total energy dynamic range (1.26 vs. 1.33 for uniform) and the gap between adjacent timesteps, which explains both the higher per-sample violation rate and the weaker Pearson correlation (-0.54 vs. -0.80)—the energy–time relationship is less linear for the mask source because the energy curve flattens more quickly in the mid-t range. None of this affects navigation, however, which depends only on relative energies between candidates at the same midpoint; the perfect bin-level monotonicity of both variants confirms the compass's suitability for this purpose.

Table 11: **Aggregate monotonicity metrics.** Both compasses achieve perfect bin-level monotonicity.

Aggregate metric	**Uniform**	**Mask**
Adjacent bins strictly monotone	10/10 (100%)	10/10 (100%)
Per-sample smoothness (adj. pairs satisfying Equation (15))	97.2% (3 886/4 000)	94.0% (3 760/4 000)
Samples with any violation	44/200 (22%)	74/200 (37%)
Spearman ρ (t vs. E_ϕ, all 4 200 points)	−0.94	−0.85
Pearson r (t vs. E_ϕ, all 4 200 points)	−0.80	−0.54
Energy gap (mean E at lowest t − mean E at t=1)	1.33	1.26

Violations by time region. Table 12 breaks down per-sample violations by time region for both source distributions. The two distributions exhibit qualitatively different violation profiles. For the uniform compass, violations concentrate at high t where adjacent timesteps differ by only a few tokens and energy differences are small relative to estimation noise—as reflected in the growing standard deviation from 0.07 at t≈0.2 to 0.38 at t=1.0 (Table 10). The mask compass instead distributes violations more broadly, peaking in the $[0.3, 0.7)$ range. This pattern is consistent with the compressed energy curve: in the mid-t regime, the mask compass's energy differences between

adjacent bins are small (0.04–0.08 units), comparable to the per-sample standard deviation, making local ordering more susceptible to noise.

Importantly, violations in both cases are mild. The largest energy gap in the wrong direction is 1.14 for the uniform compass and 1.40 for the mask compass, both concentrated in a small number of outlier samples. These violations do not affect the bin-level monotonicity, which is perfect for both variants. This result is notable because neither energy model receives *any time input*—they score the raw token sequence only. The monotonic decrease confirms that both compass variants learn to assess intrinsic text quality from content alone: sequences with more coherent, on-target tokens receive lower energy regardless of which timestep produced them. This property is precisely what enables the compass to compare candidate midpoints during RK-4 trajectory construction, where the "correct" timestep is not well-defined for stochastically generated candidates.

Table 12: **Monotonicity violations by time region.** Number of adjacent time pairs where E_ϕ incorrectly increases with t, for both source distributions.

Time region	[0, .1)	[.1, .2)	[.2, .3)	[.3, .4)	[.4, .5)	[.5, .6)	[.6, .7)	[.7, .8)	[.8, .9)	[.9, 1)
Uniform	3	4	4	4	11	12	18	19	17	22
Mask	8	16	14	35	30	34	30	32	25	16

B.2 Generation-relevant corruption detection

A compass that assigns low energy to clean text is insufficient; it must assign *higher* energy to sequences exhibiting the specific corruptions that arise during discrete flow matching generation. We evaluate both energy model variants on their ability to correctly rank real flow states below corrupted versions across four types of corruption, organized by the failure mode they simulate. Formally, for a real flow state x_t and a corrupted version $\tilde{x}_t$ constructed by applying any generation-relevant corruption strategy:

$$E_\phi(x_t) < E_\phi(\tilde{x}_t) \qquad \text{(real flow state = lower energy than corrupted).} \tag{16}$$

We measure how consistently the trained compass satisfies this criterion across four corruption strategies on held-out data:

- **Random replacement** (`random_replace`): Replace revealed tokens with uniform random vocabulary items (∼99 tokens changed on average). Tests basic quality detection.
- **Frequency-matched replacement** (`frequency_replace`): Replace revealed tokens with tokens from the same unigram frequency bin (∼104 tokens changed). A harder variant that cannot be detected by frequency statistics alone.
- **Token repetition** (`token_repeat`): Repeat a single token across revealed positions (∼93–103 tokens changed). Simulates the degenerate repetition failure mode common in discrete diffusion.
- **Time downstep** (`time_downstep`): Construct $x_{t'}$ at $t' < t$ from the same (x_0, x_1) pair (∼116 tokens changed). This is a less-revealed version of the same text—testing whether the compass detects the quality difference between two *real* flow states at different times.

For each source distribution, we evaluate 1 130 held-out (positive, negative) pairs per strategy (4 520 pairs total per variant), measuring the fraction satisfying $E_\phi(\text{real}) < E_\phi(\text{corrupted})$, i.e., Equation (16).

Both compass variants satisfy Equation (16) near-perfectly across all four strategies. The uniform compass achieves a mean accuracy of 98.5%, with no strategy falling below 98%. The mask compass achieves a mean accuracy of 99.8%, with perfect match on three of four strategies and 99.4% on token repetition.

The mask compass's higher overall accuracy may appear to contradict the weaker monotonicity metrics shown in Table 10; however, these measure different properties. Corruption detection compares a real flow state against an explicitly degraded version at the *same* timestep—a within-timestep discrimination task where the quality gap is large. Monotonicity requires distinguishing real flow states at *adjacent* timesteps—a much finer-grained comparison where even a few tokens of difference can be within the estimation noise. The mask compass excels at the former while struggling more with the latter, consistent with its compressed energy range.

Table 13: **Per-strategy energy ranking accuracy.**

Strategy	Uniform (%)	Mask (%)
`random_replace`	98.6	100.0
`frequency_replace`	98.4	100.0
`token_repeat`	98.8	99.4
`time_downstep`	98.2	100.0
Overall	**98.5**	**99.8**

Both variants' uniformly high performance across strategies is notable. The strategies span fundamentally different corruption types—wrong tokens (random and frequency-matched replacement), degenerate repetition (token repeat), and reduced revelation (time downstep)—yet both compasses handle all of them equally well. This indicates that E_ϕ has learned a general notion of sequence quality rather than a narrow definition that fits only a single failure mode.

Frequency-matched replacement achieves near-identical accuracy to random replacement for both variants (98.4% vs. 98.6% for uniform; both 100% for mask). Since frequency-matched negatives preserve the unigram token distribution of the original sequence, the energy model cannot rely on simple frequency cues—it must assess contextual coherence to distinguish real from corrupted flow states.

We deliberately exclude order-sensitive corruptions (token shuffling, segment reversal, neighbor swaps) from both training and evaluation. During discrete flow matching generation, the velocity model reveals tokens *at their correct positions*—it never permutes correct tokens into wrong positions. Order-based corruptions test a capability irrelevant to the downstream use case and waste training signal on a task that a small transformer struggles with. Focusing the training budget on generation-relevant corruptions produces a compass better calibrated for its actual role.

C Sequence-to-Token compass navigation

We expand Section 4.2 with the complete specification of the NAVIGATE procedure used at each RK-4 midpoint during navigation shaping.

C.1 Motivation: the granularity dilemma in discrete guidance

Recall that the energy compass $E_\phi(x_t)$ provides a scalar assessment of global sequence quality. Using it to guide CTMC midpoint jumps requires converting the global signal into per-position transition decisions. The mathematically ideal guided rate modifies each single-token transition by its energy differential:

$$\tilde{u}_t(y, x) = u_t^\theta(y, x) \cdot \exp\big(-\beta_t \left[E_\phi(y) - E_\phi(x)\right]\big), \tag{17}$$

where $y = x[i \to v]$ differs from x by one token at position i. Evaluating this for all positions and all vocabulary entries requires $|\mathcal{V}| \times L$ energy forward passes per midpoint, which is intractable.

The two naive strategies for approximating Equation (17) occupy opposite extremes of a granularity spectrum, and neither is satisfactory for trajectory construction:

Naive strategy	Cost per midpoint	Failure mode
Sequence-level only	K energy calls	Cannot fix individual bad tokens
Position-factorized	$\lvert\mathcal{V}\rvert \cdot L$ calls	Assumes $E(x) \approx \sum_i e_i(x_i)$; rarely holds

One could score entire candidate sequences and select the best (sequence-level), which is affordable but coarse: the energy function evaluates holistically, so a candidate can win despite containing individually poor tokens simply because the rest of the sequence compensates. Alternatively, one could assume the energy decomposes across positions (position-factorized), enabling per-token guidance, but assuming token independence is at odds with the energy model whose purpose is to capture global coherence.

Figure 4 illustrates the dilemma and how Sequence-to-Token Navigation sidesteps it by decomposing guidance into two complementary phases that operate at different granularities:

1. **Sequence phase** (sequence-level): Generate K diverse candidate jumps, score each with one energy call, and select the globally best candidate. This ensures the trajectory enters a low-energy region of sequence space.

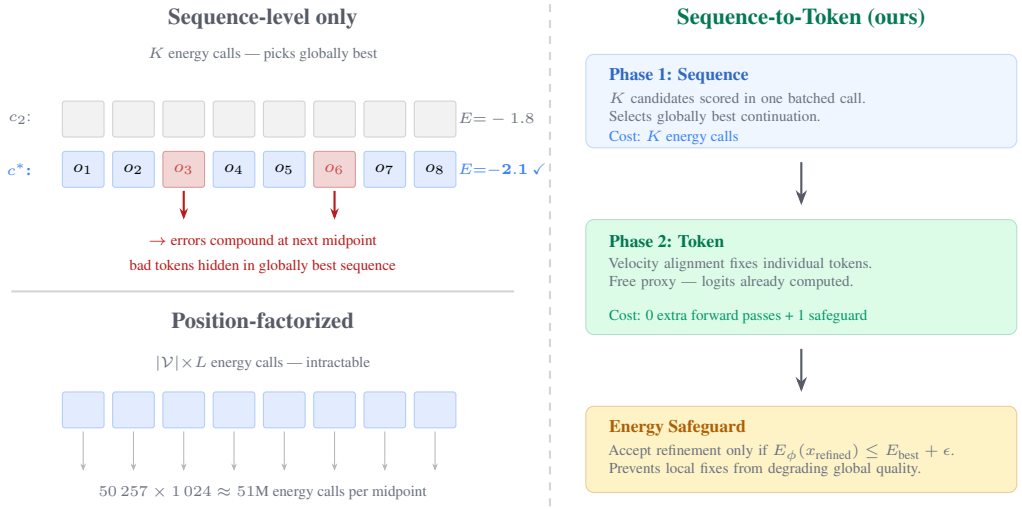


Figure 4: **The granularity dilemma.** *Left:* Sequence-level scoring selects the globally best candidate but cannot fix individually bad tokens (in red). Position-factorized guidance fixes each token at a cost of millions of energy evaluations per midpoint. *Right:* Sequence-to-Token Navigation resolves the tension—Phase 1 finds the best continuation via K energy calls; Phase 2 refines individual tokens using velocity alignment at zero model-inference cost; the energy safeguard ensures refinement never degrades global quality.

2. **Token phase** (fine-level): Refine the selected candidate using the velocity network's own confidence as a *free proxy* for per-token quality (no additional energy calls). A single energy evaluation on the refined result serves as a safeguard against degradation.

The key insight of the token phase is that the velocity model v_θ already encodes a distribution over plausible tokens at every position. Positions where the selected candidate disagrees with a high-confidence velocity prediction are likely suboptimal; those already aligned with the velocity's top prediction are likely correct. The velocity-alignment signal is available at zero cost since the logits are already computed and provides a principled, position-level refinement criterion that complements the sequence-level energy signal.

C.2 Phase 1: sequence exploration via diverse candidate scoring

Temperature-diverse candidate generation. To ensure the K candidates span a meaningful range of the trajectory space, we sample each at a different temperature [Zhang et al., 2024, Du et al., 2025]. The base temperature adapts to the velocity's current uncertainty via its per-position entropy:

$$T_{\text{base}} = T_{\min} + \Delta_T \cdot (1 - \tilde{H}), \tag{18}$$

where $\tilde{H} \in [0, 1]$ is the batch-normalized mean entropy of the velocity distribution, $T_{\min}$ $(= 0.8)$ sets the minimum base temperature, and Δ_T $(= 0.4)$ controls the entropy-dependent range, so that $T_{\text{base}} \in [T_{\min},\ T_{\min} + \Delta_T]$. The values $T_{\min}$=0.8 and Δ_T=0.4 are chosen so that the base temperature stays in the near-unity regime: temperatures far below 1.0 would collapse diversity among candidates, while temperatures far above 1.0 would introduce too much noise into the CTMC jumps, producing candidates that are diverse but uniformly poor. The range $[0.8, 1.2]$ keeps sampling close to the velocity's learned distribution while allowing meaningful variation.

This schedule is counterintuitive: when the velocity is *confident* ($\tilde{H}$ low), we raise the temperature slightly to verify that no better alternatives exist; when the velocity is *uncertain* ($\tilde{H}$ high), we lower it to concentrate on high-probability tokens rather than amplifying noise. Each of the K candidates is sampled at temperature

$$T_c = T_{\text{base}} \cdot \left(\lambda_{\min} + (\lambda_{\max} - \lambda_{\min}) \cdot \tfrac{c-1}{K-1}\right), \qquad c = 1, \ldots, K, \tag{19}$$

where the per-candidate multiplier spans $[\lambda_{\min},\ \lambda_{\max}]$ $(= [0.7,\ 1.3])$, producing effective temperatures in $[\lambda_{\min} T_{\text{base}},\ \lambda_{\max} T_{\text{base}}]$. Low-temperature candidates ($\lambda_{\min}$) exploit the velocity's modal

predictions; high-temperature candidates ($\lambda_{\max}$) explore the tails. The symmetric spread of $\pm 30\%$ around T_{base} reflects a balance: narrower spreads produce near-identical candidates that waste the multi-candidate budget, while wider spreads push high-temperature candidates into a regime where the sampled tokens are effectively random, contributing little useful signal to the energy-based selection. Each sampled prediction $\hat{x}_1^{(c)}$ is evolved under the standard CTMC jump dynamics to produce the candidate midpoint state $x^{(c)} = \text{CTMCJump}(x_t, \hat{x}_1^{(c)}, h)$.

Energy-based selection. The energy compass scores all candidates in a single batch to identify the best candidate $x_{\text{best}} = x^{(c^*)}$ with energy E_{best}:

$$c^* = \arg \min_{c \in \{1,\ldots,K\}} E_\phi(x^{(c)}). \tag{20}$$

The sequence phase guarantees that the selected candidate has the lower *global* energy. Since the energy function evaluates sequences holistically, a sequence can win despite containing some weak tokens. The weak tokens, which may lead to compounding errors in subsequent RK-4 midpoints, go uncorrected (the red tokens in the left side of Figure 4).

C.3 Phase 2: token refinement via velocity-aligned token correction

The token phase corrects individual token-level weaknesses in x_{best} using the velocity network's own predictions as a refinement signal—*without any per-token energy calls*.

Velocity alignment as a free proxy for token quality. The velocity network's logits at x_t have already been computed as part of the RK-4 velocity evaluation—no additional forward pass is needed. We reuse these logits to sample a fresh prediction $\hat{x}_1^{\text{ref}} \sim p^\theta_{1|t}(\cdot \mid x_t)$ and to extract the velocity's argmax prediction $x^{(i)}_{\text{top}} = \arg\max_v p^{(i)}_{1|t}(v \mid x_t)$ and its confidence $\text{conf}_i = \max_v p^{(i)}_{1|t}(v \mid x_t)$. All three quantities—the refinement sample, the top prediction, and the confidence—are derived from the same cached logits via cheap categorical operations (sampling, argmax, and max), with zero model-inference cost. The alignment signal compares the refinement proposal against the selected candidate at each position:

$$\Delta^{(i)}_{\text{align}} = \mathbf{1}\big[\hat{x}^{\text{ref}}_{1,i} = x^{(i)}_{\text{top}}\big] - \mathbf{1}\big[x^{(i)}_{\text{best}} = x^{(i)}_{\text{top}}\big] \;\in\; \{-1,\, 0,\, +1\}. \tag{21}$$

The interpretation is direct: $\Delta^{(i)}_{\text{align}} = +1$ means the refinement proposal matches the velocity's most confident prediction while the current token does not—a strong signal to replace. $\Delta^{(i)}_{\text{align}} = -1$ means the current token already matches but the proposal does not—leave it. $\Delta^{(i)}_{\text{align}} = 0$ means no alignment difference—default to the base jump rate.

Why velocity alignment works as a proxy. The velocity network was trained on the data distribution and captures the conditional probability of each token given the current flow state. At positions where the velocity is highly confident (large conf_i), its top prediction is a strong indicator of the "correct" token under the data distribution. A candidate token that disagrees with a high-confidence velocity prediction is statistically more likely to be an artifact of the stochastic CTMC jump than a genuine feature of the target sequence. By using this signal, which needs no additional forward pass, we obtain position-level guidance that is complementary to the energy model's sequence-level assessment. Figure 5 illustrates this mechanism on a concrete example, showing how confidence gating prevents overriding uncertain positions.

Adaptive guidance strength. The token refinement modulates jump probabilities using a guidance coefficient β_{adp} that controls how strongly the velocity-alignment signal influences token replacement decisions. A fixed coefficient throughout the flow trajectory is suboptimal: at low t (mostly noise), the velocity network's predictions are uncertain and the alignment signal should carry more weight; at high t (mostly revealed text), the velocity is increasingly reliable on its own and heavy-handed modulation risks overriding well-calibrated local predictions. We therefore decay the guidance intensity:

$$\beta_{\text{adp}} = \beta_0 \cdot (1 - \alpha_\beta\, t), \tag{22}$$

where α_β ($= 0.5$) controls the decay rate, yielding full strength β_0 at t=0 and $\beta_0/2$ at t=1. The linear form is the simplest monotone schedule, and the choice α_β=0.5 ensures guidance is never fully switched off—even at late timesteps where the sequence is mostly revealed, a moderate energy signal helps avoid last-mile errors at the remaining unrevealed positions.

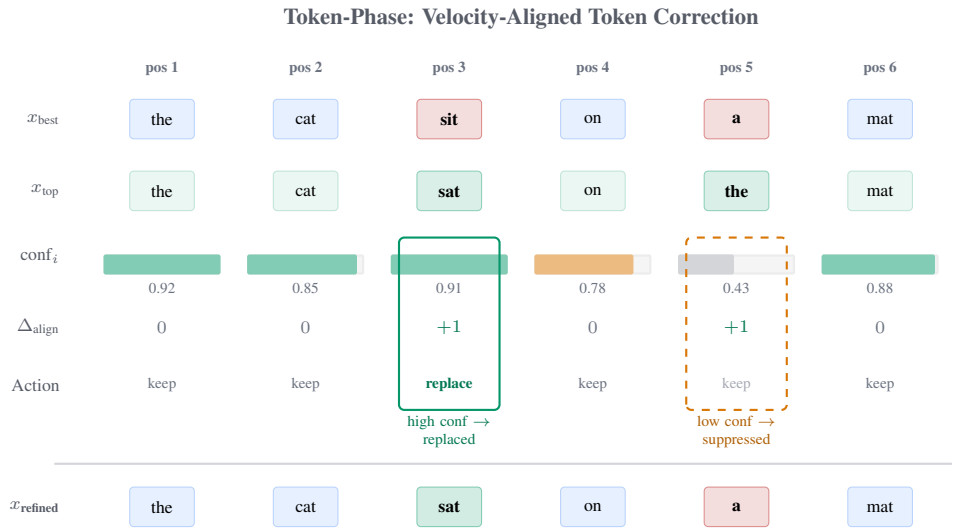


Figure 5: **Token-phase token-level detail.** Phase 1's x_{best} is compared against the velocity's argmax prediction x_{top} at each position. Where the two disagree ($\Delta_{\text{align}}{=}{+}1$), the replacement decision is modulated by velocity confidence: position 3 ("sit"→"sat") has high confidence (0.91) and is replaced; position 5 ("a"→"the") has low confidence (0.43) and is kept. *Confidence gating* prevents overriding well-calibrated local predictions when velocity is uncertain.

Confidence-modulated jump probability. The base CTMC jump probability $p_{\text{base}}^{(i)} = 1 - \exp(-h \cdot \lambda_{\text{base}})$ (where $\lambda_{\text{base}} = \dot{\kappa}_t/(1-\kappa_t)$, applied only at positions where the refinement proposes a different token) is modulated by the alignment signal:

$$p_{\text{jump}}^{(i)} = \text{clamp}\Big(p_{\text{base}}^{(i)} \cdot \exp\big(\Delta_{\text{align}}^{(i)} \cdot \text{conf}_i \cdot \beta_{\text{adp}} \cdot \alpha_{\text{soft}}\big),\ 0,\ p_{\max}\Big), \tag{23}$$

where α_{soft} ($= 0.5$) is a softening factor and $p_{\max}$ ($= 0.95$) caps the jump probability. Three design choices in Equation (23) merit explanation:

- **Confidence gating** (conf_i): The modulation magnitude scales with the velocity's certainty at that position. At uncertain positions, the alignment signal is unreliable and the adjustment is automatically suppressed, preventing overconfident refinement of ambiguous tokens (see position 5 in Figure 5 for a concrete example).
- **Softening factor** α_{soft} ($= 0.5$): Without softening ($\alpha_{\text{soft}}{=}1$), the exponential modulation can produce extreme adjustments—a position with $\text{conf}_i{=}0.9$ and $\beta_{\text{adp}}{=}1$ would scale the jump rate by $\exp(\pm 0.9) \approx 2.5\times$ or $0.4\times$. The factor $\alpha_{\text{soft}}{=}0.5$ halves the exponent, keeping rate adjustments within a moderate range ($\exp(\pm 0.45) \approx 1.6\times$ or $0.6\times$) that steers without overwhelming the base dynamics.
- **Upper bound** $p_{\max}$ ($= 0.95$): Capping below 1.0 preserves stochasticity. Deterministic jumps ($p{=}1$) risk oscillations between competing token choices at successive midpoints and remove the randomness needed for the CTMC dynamics. The value 0.95 is the standard near-deterministic cap used in discrete sampling; any value in $[0.9, 0.99]$ would serve the same purpose.

Tokens are then stochastically replaced according to the adjusted probabilities:

$$x_{\text{refined}}^{(i)} = \begin{cases} \hat{x}_{1,i}^{\text{ref}} & \text{with probability } p_{\text{jump}}^{(i)}, \\ x_{\text{best}}^{(i)} & \text{otherwise.} \end{cases} \tag{24}$$

C.4 Energy safeguard

The token-level refinement in Phase 2 is guided by velocity alignment, which is a proxy for—but not identical to—sequence-level energy. It is possible (though uncommon in practice) that a set

Algorithm 1 Sequence-to-Token compass navigation (NAVIGATE)

Require: State x_t, velocity logits, energy compass E_ϕ, time t, step h, threshold τ, candidates K
Require: Hyperparameters: $T_{\min}$, Δ_T, $\lambda_{\min}$, $\lambda_{\max}$, α_β, α_{soft}, $p_{\max}$, ϵ_{safe}
1: **if** $t < \tau$ **then**
2: **return** StandardCTMCJump(x_t, logits, h) {Below threshold: no guidance}
3: **end if**
4: *// — Phase 1: Sequence exploration —*
5: Compute per-position entropy H from logits
6: $T_{\text{base}} \leftarrow T_{\min} + \Delta_T \cdot (1 - \tilde{H})$
7: **for** $c = 1, \ldots, K$ **do**
8: $T_c \leftarrow T_{\text{base}} \cdot (\lambda_{\min} + (\lambda_{\max} - \lambda_{\min}) \cdot (c-1)/(K-1))$
9: $\hat{x}_1^{(c)} \sim \text{softmax}(\text{logits}/T_c)$; $x^{(c)} \leftarrow \text{CTMCJump}(x_t, \hat{x}_1^{(c)}, h)$
10: **end for**
11: $\mathbf{E} \leftarrow E_\phi(x^{(1:K)})$; $c^* \leftarrow \arg\min_c \mathbf{E}^{(c)}$
12: $x_{\text{best}} \leftarrow x^{(c^*)}$; $E_{\text{best}} \leftarrow \mathbf{E}^{(c^*)}$
13: *// — Phase 2: Token refinement —*
14: $\hat{x}_1^{\text{ref}} \sim p^\theta_{1|t}(\cdot \mid x_t)$; $x_{\text{top}} \leftarrow \arg\max_v p^{(i)}_{1|t}(v \mid x_t)$; $\text{conf} \leftarrow \max_v p^{(i)}_{1|t}(v \mid x_t)$
15: $\beta_{\text{adp}} \leftarrow \beta_0 \cdot (1 - \alpha_\beta\, t)$
16: $\Delta^{(i)}_{\text{align}} \leftarrow \mathbf{1}[\hat{x}^{\text{ref}}_{1,i} = x^{(i)}_{\text{top}}] - \mathbf{1}[x^{(i)}_{\text{best}} = x^{(i)}_{\text{top}}]$
17: $p^{(i)}_{\text{base}} \leftarrow \mathbf{1}[\hat{x}^{\text{ref}}_{1,i} \neq x^{(i)}_{\text{best}}] \cdot (1 - \exp(-h \cdot \lambda_{\text{base}}))$
18: $p^{(i)}_{\text{jump}} \leftarrow \text{clamp}\big(p^{(i)}_{\text{base}} \cdot \exp(\Delta^{(i)}_{\text{align}} \cdot \text{conf}_i \cdot \beta_{\text{adp}} \cdot \alpha_{\text{soft}}),\ 0,\ p_{\max}\big)$
19: $x^{(i)}_{\text{refined}} \leftarrow \begin{cases} \hat{x}^{\text{ref}}_{1,i} & \text{w.p. } p^{(i)}_{\text{jump}}, \\ x^{(i)}_{\text{best}} & \text{otherwise} \end{cases}$
20: *// — Energy safeguard —*
21: $E_{\text{ref}} \leftarrow E_\phi(x_{\text{refined}})$
22: **if** $E_{\text{ref}} \leq E_{\text{best}} + \epsilon_{\text{safe}}$ **then**
23: **return** x_{refined}
24: **else**
25: **return** x_{best}
26: **end if**

of individually well-aligned token replacements produces a sequence with higher energy when considered jointly. A single energy evaluation on the refined sequence provides a safety net:

$$x'_{\text{mid}} = \begin{cases} x_{\text{refined}} & \text{if } E_\phi(x_{\text{refined}}) \leq E_{\text{best}} + \epsilon_{\text{safe}}, \\ x_{\text{best}} & \text{otherwise,} \end{cases} \tag{25}$$

with tolerance ϵ_{safe} ($= 0.1$). The tolerance accommodates energy-estimation noise: a refinement that slightly increases estimated energy may still be genuinely beneficial (e.g., improving velocity alignment at the cost of a negligible energy increase within the estimator's error margin). The value ϵ_{safe}=0.1 is small relative to the typical energy range across candidates (which spans several units in our experiments), so it permits only marginal degradation while avoiding the brittleness of an exact $\leq$ check on a noisy scalar. In our experiments, the safeguard triggers on fewer than 8% of midpoints, confirming that velocity-aligned refinement is well-correlated with energy improvement.

C.5 Full algorithm

Algorithm 1 consolidates the complete NAVIGATE procedure described in the preceding subsections. Phase 1 (lines 4–12) generates K temperature-diverse candidates and selects the lowest-energy one via a single batched energy call. Phase 2 (lines 13–19) refines the winner through velocity-aligned token replacement, modulated by the adaptive guidance coefficient β_{adp} (Equation (22)) and the velocity's per-position confidence. The energy safeguard (lines 20–24) accepts the refinement only if it does not significantly increase energy, falling back to the Phase 1 winner otherwise. All symbols and hyperparameters are summarized in Table 14.

Hyperparameter choices. We did not tune the navigation constants by search. A one-factor-at-a-time sweep of β_0 and ϵ_{safe} at inference shows a narrow spread (Appendix E.9). The two values

Table 14: Sequence-to-Token compass navigation hyperparameters with default values and rationale. Only β_0 and ϵ_{safe} are swept (Appendix E.9), and K is ablated separately (Appendix D.3). The remaining constants, including the temperature range ($T_{\min}$, Δ_T, $\lambda_{\min}$, $\lambda_{\max}$), are set structurally and not swept.

Symbol	Value	Role	Rationale
K	5	Candidates per midpoint	Balances diversity vs. cost (Table 15)
$T_{\min}$	0.8	Min base temperature	Keeps sampling near the velocity's mode
Δ_T	0.4	Entropy-dependent range	Bounds $T_{\text{base}} \in [0.8, 1.2]$, near-unity regime
$\lambda_{\min}$	0.7	Per-candidate temp lower	$\pm 30\%$ spread: diverse yet informative
$\lambda_{\max}$	1.3	Per-candidate temp upper	(see text)
β_0	0.3	Base guidance strength	Scales token-refinement modulation (Equation (22))
α_β	0.5	Guidance decay rate	Halves guidance at $t{=}1$; never fully off
α_{soft}	0.5	Jump-rate softening	Keeps rate adjustments in moderate range
$p_{\max}$	0.95	Max jump probability	Preserves CTMC stochasticity
ϵ_{safe}	0.1	Safeguard tolerance	Accommodates energy-estimation noise

that materially affect quality, K and τ, are ablated in the paper. The remainder are chosen from structural requirements of the method—keeping sampling near the velocity's distribution, ensuring guidance never fully switches off, capping jumps below determinism to preserve stochasticity—and are held fixed across all experimental settings: 170M and 1.3B, uniform and mask sources, both initializations. The same constants work unchanged at two model scales and two source distributions, which, together with that sweep, is our evidence that they are not tuned artifacts.

C.6 Why both phases are necessary

The two phases address fundamentally different failure modes of trajectory construction, and neither alone suffices. Figure 4, left panel, illustrates the failure mode of each naive strategy in isolation.

Phase 1 (sequence) alone is insufficient. Best-of-K selection by energy is powerful but blunt. The energy function scores sequences holistically: a candidate can win despite containing individually weak tokens, provided the remainder of the sequence compensates. These uncorrected weak tokens become compounding errors through subsequent RK-4 midpoints—precisely the "blind navigation" problem that navigation shaping aims to solve. Sequence selection reduces the *frequency* of such errors but cannot eliminate them.

Phase 2 (token) alone is insufficient. Token-level refinement via velocity alignment is cheap and fine-grained, but it has no access to the energy landscape. Velocity confidence is a useful proxy for per-token quality, but it is not a substitute for energy: the velocity was trained to predict the data distribution, not to minimize the compass energy. Without a good starting point from Phase 1, local refinement would be polishing the wrong candidate—a trajectory that is locally coherent at each position but globally incoherent.

Together: global selection + local correction. Phase 1 ensures the trajectory enters a globally low-energy region of sequence space. Phase 2 then makes targeted, confidence-weighted corrections within that region—fixing the individual token-level weaknesses that the holistic energy score cannot isolate. The energy safeguard (Equation (25)) ensures that local refinement never degrades the globally low-energy state identified by Phase 1. This sequence-to-token decomposition applies a natural two-stage principle: first select the best global continuation, then refine it locally.

C.7 Computational cost and practical considerations

C.7.1 Forward-pass budget

Each call to NAVIGATE requires K energy evaluations for sequence scoring (parallelized in a single batched call) plus one energy evaluation for the safeguard, totaling $K + 1$ energy forward passes per navigated midpoint. With our default $K{=}5$, this is 6 energy calls per midpoint, or up to $3 \times 6 = 18$

calls per RK-4 step when all three midpoints are navigated. The energy model ($\sim$90M parameters) is $1.9\times$ smaller than the 170M velocity model, so each energy call costs proportionally fewer FLOPs.

Two factors temper this overhead when amortized across training. First, not every training step invokes RK-4: following the FS-DFM training schedule [Monsefi et al., 2026], roughly 10 out of every 30 training steps use the RK-4 estimator, while the remaining 20 steps operate in the shortcut regime where the target is the ground-truth x_1 directly—requiring neither trajectory construction nor energy guidance. Second, within each RK-4 step, the time-threshold mechanism (Section 4.3) skips navigation at midpoints where $t < \tau$. Table 15 decomposes the resulting per-step cost and amortization. In the worst case (all midpoints navigated), the theoretical overhead is $\sim 2.4\times$ FS-DFM. In practice this is an upper bound—the time threshold τ skips some midpoints entirely, and for $K<5$ the energy cost shrinks proportionally. Measured wall-clock overhead and analysis are in the main paper.

Table 15: Theoretical cost decomposition for navigation shaping at the 170M scale with default $K=5$. FLOPs are expressed relative to a single velocity forward pass on the 170M student. The amortization accounts for the RK-4/shortcut split in the training schedule.

	Model calls		**Relative FLOPs**	
Component	**Velocity**	**Energy**	**FS-DFM**	**TS-DFM**
Per RK-4 step ($\sim$10 of 30 training steps):				
Velocity (4 semi-teacher + 1 student)	5	0	$5.0\times$	$5.0\times$
Energy: sequence scoring ($K=5\times$ $\leq$3 midpts)	0	≤ 15	—	$\leq 7.9\times$
Energy: safeguard ($\leq$3 midpoints)	0	≤ 3	—	$\leq 1.6\times$
RK-4 step subtotal	5	≤ 18	$5.0\times$	$\sim 14.5\times$
Per shortcut step ($\sim$20 of 30 training steps):				
Velocity (student only)	1	0	$1.0\times$	$1.0\times$
Amortization $(10 \cdot \text{RK-4} + 20 \cdot \text{shortcut})/30$:				
Amortized per step			$2.33\times$	$\sim 5.5\times$
Theoretical TS-DFM overhead vs. FS-DFM			$\sim \mathbf{2.4}\times$	

D Empirical validation of the navigation policy

The energy compass and Sequence-to-Token (S2T) navigation policy are used exclusively at training time in our main experiments. To validate these components independently of the training dynamics, we apply them *at inference time*—directly guiding the baseline FS-DFM student's generation process. Inference-time guidance is too slow for practical deployment (Appendix C.7), but it provides a controlled testbed: the student's weights are held fixed, so any improvement is attributable solely to the compass and navigation policy. All experiments in this section use the same 170M-parameter baseline FS-DFM student trained with the *uniform* source distribution and *without* trajectory shaping.

The energy compass itself is validated separately in Appendix B, where it achieves 98.5–99.8% corruption-detection accuracy and perfect bin-level monotonicity across both source distributions. The experiments below focus on whether the navigation policies built on top of the compass—sequence selection, token refinement, and their combination—translate these properties into actual generation improvements.

D.1 Navigation policy comparison across step counts

We compare four navigation policies applied to the same baseline FS-DFM student: *no guidance* (standard blind CTMC jumps), *sequence only* ($K=5$ best-of-K selection), *token only* (velocity-aligned refinement without candidate selection), and *Sequence-to-Token (S2T)* (the full two-phase policy). All four use identical student weights and the same energy compass; they differ only in how midpoint states are constructed. We apply the policies at inference time so that any perplexity difference is attributable solely to the policy itself, with student weights held fixed. We evaluate at 8,

16, 32, and 64 inference steps: the 8-step setting is the student's final inference target, and the larger step counts probe how each policy scales when steps are smaller and more numerous.

Table 16: **Navigation policy comparison across step counts (inference time).** All policies use the same baseline student and energy compass; they differ only in how midpoint states are constructed during generation.

Steps	Policy	Ent	PPL (G^2-L)↓	PPL (L^2-7B)↓	PPL (L^3-8B)↓
8	No guidance	7.32	89.6	58.2	109.7
	Sequence only (K=5)	6.99	**81.2**	**52.2**	**99.6**
	Token only	7.04	90.7	62.7	106.8
	Sequence-to-Token	6.93	86.2	58.3	104.3
16	No guidance	7.51	70.4	47.3	88.7
	Sequence only (K=5)	7.21	51.9	**33.3**	61.3
	Token only	7.29	58.9	41.7	69.3
	Sequence-to-Token	7.10	**50.8**	36.3	**59.5**
32	No guidance	7.70	70.6	39.7	68.1
	Sequence only (K=5)	7.40	53.3	36.5	58.4
	Token only	7.39	46.8	35.7	55.4
	Sequence-to-Token	7.26	**43.1**	**31.2**	**49.1**
64	No guidance	7.82	75.0	41.1	72.0
	Sequence only (K=5)	7.50	52.0	31.5	51.3
	Token only	7.54	47.6	31.0	47.1
	Sequence-to-Token	7.36	**38.0**	**24.4**	**37.0**

Key observations. (1) At 8 steps, differences are modest. At the student's target inference budget (8 steps), all navigation policies produce similar perplexity: the gap between no guidance and the best policy is ∼9% on GPT-2 Large (89.6 → 81.2) and ∼10% on LLaMA-2 7B (58.2 → 52.2). With so few steps, each step covers a large portion of the flow trajectory, leaving little room for per-step guidance to accumulate improvements. Sequence selection alone provides the largest benefit in this regime by avoiding the worst stochastic realizations at each of the few available decision points.

(2) At higher step counts, S2T dominates. Starting at 16 steps, S2T consistently achieves the best or near-best perplexity across all three evaluators. The advantage grows with step count: at 32 steps, S2T reduces GPT-2 Large perplexity by 39% relative to no guidance (70.6 → 43.1) and outperforms both sequence-only (53.3) and fine-only (46.8). At 64 steps—the regime most relevant to RK-4 midpoint construction—S2T achieves 38.0 GPT-2 Large perplexity, a 49% reduction from the unguided baseline and 20% better than fine-only (47.6). This confirms the complementarity argument from Appendix C.6: neither phase alone matches their combination.

(3) Unguided generation degrades at high step counts. Counterintuitively, the unguided baseline's perplexity *worsens* from 32 to 64 steps (GPT-2 Large: 70.6 → 75.0). More steps mean more stochastic jumps, each introducing error; without guidance, errors compound and eventually outweigh the benefit of finer-grained integration. All guided policies avoid this degradation entirely, with S2T continuing to improve from 32 to 64 steps (43.1 → 38.0 on GPT-2 Large). This divergence underscores exactly why navigation shaping is needed during RK-4 target construction: the blind-jump regime where training operates is precisely where compounding errors are most damaging.

(4) The two phases have complementary strengths across step regimes. At 8 steps, sequence selection dominates: with few decision points, picking the globally best candidate among K options is more impactful than token-level corrections diluted across large steps. At 16 steps, S2T begins to overtake sequence-only as token refinement accumulates enough small corrections to matter. By 32–64 steps, the combination is decisive: sequence selection ensures each midpoint starts in a low-energy region, and token refinement corrects the individual token-level errors that sequence scoring cannot isolate—yielding improvements that neither phase achieves alone. This is precisely the operating regime of RK-4 midpoint construction during training, where the S2T design provides its largest advantage.

Connection to training-time dynamics. The behavior at 16–64 inference steps is more than a stress test—it is a proxy for how S2T behaves when it runs inside RK-4 target construction during

training. Training a student for 8-step generation (h=0.125) requires RK-4 midpoints at sub-step sizes of $h/2 = 0.0625$, corresponding to the 16–32-step regime, and S2T is called at each midpoint. The inference-time results at 16–64 steps therefore reflect the regime in which S2T actually operates when it shapes the student's training signal. Two of the patterns above are directly informative in this light: S2T's growing advantage with step count (observation 2) predicts that its benefit compounds inside RK-4 chains, and the unguided baseline's high-step degradation (observation 3) shows exactly the failure mode navigation shaping prevents.

D.2 Token refinement behavior across step counts

To understand what the token refinement phase actually does, we collect refinement statistics over 504 sequences at each step count. Recall that at each guided midpoint, the token phase proposes token-level replacements based on velocity alignment (Appendix C.3), then the energy safeguard decides whether to accept or reject the refined sequence (Appendix C.4). We measure four quantities:

- **Accept %**: The fraction of midpoints where the energy safeguard accepts the refined sequence, i.e., the refinement does not increase energy beyond the tolerance ϵ_{safe}. A high rate indicates that velocity-aligned corrections are well-correlated with energy improvement.
- **Tokens / step**: The mean number of tokens replaced per midpoint by the token phase (out of $1\,024$ total positions). This measures how aggressive the refinement is—lower values indicate more targeted correction.
- $\Delta E_{\textbf{acc}}$: The mean energy change for accepted refinements. Negative values confirm that accepted refinements genuinely improve sequence quality.
- $\Delta E_{\textbf{rej}}$: The mean energy change for rejected refinements. Positive values confirm that the safeguard correctly blocks modifications that would have degraded quality.

Table 17 reports these metrics across step counts. The results reveal a consistent pattern: as step count increases and each step covers a smaller portion of the trajectory, refinement becomes more selective, more reliable, and smaller in magnitude.

Table 17: **Token refinement statistics** (504 sequences per step count).

Steps	**Accept %**	**Tokens / step (%)**	$\Delta E_{\textbf{acc}}$	$\Delta E_{\textbf{rej}}$
8	94.7%	79.4 (7.8%)	−0.081	+0.629
16	97.8%	36.1 (3.5%)	−0.041	+0.757
64	99.9%	8.2 (0.8%)	−0.010	+0.348

The results reveal several consistent patterns. First, refinement is selective at every step count: even at 8 steps, only 7.8% of tokens are replaced per step, dropping to 0.8% at 64 steps. This confirms the design intent of targeted correction rather than wholesale rewriting. Second, the safeguard acceptance rate increases monotonically from 94.7% at 8 steps to 99.9% at 64 steps, indicating that velocity-aligned token replacements are increasingly well-correlated with energy improvement as step size shrinks—the velocity network's confidence is a more reliable proxy for token quality when each step is small. Third, rejected refinements would have *increased* energy at every step count ($\Delta E_{\text{rej}} > 0$), confirming that the safeguard correctly identifies and blocks the small fraction of refinements that would degrade sequence quality.

D.3 Effect of candidate count K

The number of candidates K in the sequence phase controls the trade-off between navigation quality and computational cost: more candidates increase the chance of finding a low-energy midpoint, but each requires an additional energy forward pass. We ablate K at inference time by applying S2T guidance with varying candidate counts to the baseline student (Table 18). All experiments generate 504 sequences at 8 steps on a single GPU. Wall-clock time measures the total generation time for all 504 sequences (excluding perplexity evaluation), synchronized across GPU operations.

The results show clear diminishing returns. The largest improvement comes from introducing candidate selection at all: K=2 reduces LLaMA-3 8B PPL by 23% relative to no guidance (115.9

Table 18: **Effect of candidate count K (S2T, 8 steps).**

K	Ent	PPL (G^2-L)↓	PPL (L^2-7B)↓	PPL (L^3-8B)↓	Wall time (s)
No guidance	7.26	90.8	59.4	115.9	31.0
2	6.56	61.2	49.9	88.7	147.8
3	6.50	59.3	41.8	75.7	185.7
5	6.68	**56.0**	**37.8**	**66.2**	261.9
8	6.57	57.6	40.3	70.6	376.5
10	6.56	57.4	40.4	69.9	449.1

$\rightarrow$ 88.7). Each subsequent increase in K yields a progressively smaller gain—$K{=}3$ reaches 75.7 (35% reduction) and $K{=}5$ reaches 66.2 (43% reduction)—while $K{=}8$ provides no additional benefit (70.6), confirming that $K{=}5$ is an effective default.

Inference-time cost vs. training-time cost. The wall-clock times in Table 18 reflect inference-time guidance, where the energy compass runs at *every* step of generation—producing the $8.5\times$ slowdown observed at $K{=}5$. This cost structure is fundamentally different from our actual approach, where all navigation shaping occurs during training and inference is completely unchanged: the shaped student generates with the same architecture, forward pass, and speed as a standard distilled model.

At training time, the cost of navigation shaping is further reduced by three factors. First, the FS-DFM training schedule allocates only approximately one-third of training steps to the RK-4 regime (large step sizes where trajectory construction is needed); the remaining two-thirds use direct teacher targets with no trajectory construction and therefore no energy guidance (Section 3). Second, within each RK-4 step, the time-threshold mechanism (Section 4.3) activates navigation only at midpoints where $t \geq \tau$; midpoints in the source-dominated regime ($t < \tau$) use standard blind jumps at zero additional cost. Third, the energy compass ($\sim$90M parameters) is substantially smaller than the velocity model, and the K candidate evaluations are batched into a single GPU call. Together, these factors reduce the amortized training overhead to $\sim 2\times$ at the 170M scale (see the full cost breakdown in Appendix C.7).

E Additional experiments

This section presents experiments omitted from the main text due to space constraints. All experiments use the 170M-parameter model with the uniform source distribution unless otherwise noted.

E.1 Effect of time threshold τ: additional details

The main results and table for the threshold ablation are in Section 5.2. This section provides additional analysis.

Why $\tau{=}0.02$ rather than $\tau{=}0$. We use $\tau{=}0.02$ as the lowest threshold instead of $\tau{=}0$ for two reasons. First, at $t{=}0$ the sequence is pure source noise—$\alpha(0){=}0$, so no positions are revealed—and the energy compass receives a sequence with no meaningful content to score; all candidates are indistinguishable noise, making energy-based selection equivalent to random selection. Second, at very early flow times the CTMC transition rates involve the ratio $\dot{\kappa}_t/(1-\kappa_t)$, which can produce numerical instabilities when $\kappa_t \approx 0$. Setting $\tau{=}0.02$ ensures that at least $\sim 2\%$ of positions ($\sim$20 out of $1\,024$) carry revealed tokens before the compass is invoked, providing a minimal but nonzero signal for candidate comparison while avoiding numerical edge cases.

Observations (uniform source). The uniform source results show clear trends. First, **FS-DFM init consistently outperforms DFM init** at every threshold and step count—for example, at $\tau{=}0.2$ and 8 steps, FS-DFM init achieves 56.1 PPL versus 79.0 for DFM init—confirming that trajectory shaping compounds with a stronger starting point rather than substituting for it. Second, **lower thresholds yield better perplexity but at the cost of diversity**. The lowest threshold ($\tau{=}0.02$) achieves the best raw PPL across both initializations (e.g., 43.5 at 8 steps with FS-DFM init), but its generative entropy drops to 6.4 at 8 steps—below the 6.5 threshold at which token repetition and reduced lexical diversity become prevalent in discrete diffusion models. This suggests that activating the compass at nearly

every midpoint over-optimizes for the energy objective, collapsing the distribution toward high-energy (low-perplexity) but repetitive sequences. Third, τ=0.2 **offers the best quality–diversity trade-off** for the uniform source: it achieves strong perplexity (56.1 at 8 steps, 47.2 at 16 steps with FS-DFM init) while maintaining healthy entropy above 7.0, indicating that the generated text retains natural lexical diversity. The default τ=0.4 is more conservative, producing the highest entropy (7.2–7.5) but sacrificing perplexity.

Observations (mask source). The mask source exhibits a strikingly different pattern from uniform in two respects. First, **DFM init consistently outperforms FS-DFM init**—the opposite of the uniform setting. At 8 steps with τ=0.02, DFM init achieves 92.9 PPL while FS-DFM init reaches only 137.0; at 16 steps with τ=0.2, the gap is even larger (61.9 vs. 109.9). This suggests that the FS-DFM student distilled under the mask source has converged to velocity patterns that are difficult to redirect with trajectory shaping, whereas the DFM teacher provides a more flexible starting point that shaping can mold more effectively. Second, **the mask source benefits less from low τ than uniform does**. While τ=0.02 with DFM init gives reasonable PPL at 8 steps (92.9), its entropy drops to 6.2—well below the 6.5 diversity threshold—and the PPL improvement over τ=0.2 (91.4) is negligible. At 16 steps, τ=0.2 with DFM init achieves the best result overall (61.9 PPL, 7.6 entropy), confirming τ=0.2 as the preferred threshold for the mask source as well.

E.2 Midpoint diversity analysis

A key assumption underlying navigation shaping is that stochastic CTMC jumps at intermediate flow times produce meaningfully different continuations—if all candidates converged to the same output regardless of the jump, energy-based selection among K candidates would have no material to work with. We test this assumption directly by measuring how much diversity remains when multiple independent completions are generated from a shared midpoint state.

Setup. We use the 170M-parameter TS-DFM student trained with the uniform source distribution. For each trial, we sample a source sequence x_0 and run the flow solver from t=0 to a midpoint time t_{mid}, producing a shared state $x_{t_{\text{mid}}}$. From this shared state, we launch N=10 independent completions to t=1, each using fresh random seeds for all stochastic jumps. We repeat this procedure for 48 independent prefixes and report average diversity metrics across all completions.

Step allocation. Given a total budget of S steps, the prefix phase uses $S_{\text{pre}} = \max(1, \lfloor S \cdot t_{\text{mid}} \rfloor)$ steps to reach t_{mid}, and each completion uses $S_{\text{comp}} = \max(1, \lfloor S \cdot (1 - t_{\text{mid}}) \rfloor)$ steps for the remainder. Since $S_{\text{pre}} + S_{\text{comp}}$ may not equal S due to rounding, the effective step sizes in each phase ($t_{\text{mid}}/S_{\text{pre}}$ and $(1-t_{\text{mid}})/S_{\text{comp}}$) differ slightly from the uniform step size $1/S$ used during training. For example, with S=8 and t_{mid}=0.3: the prefix uses 2 steps (step size 0.15) and each completion uses 5 steps (step size 0.14), totaling 7 steps. This mismatch is minor and does not affect the qualitative conclusions, as the diversity metrics are consistent across step counts (Table 19).

Metrics. We measure three complementary aspects of diversity among the N completions from each shared midpoint.

Self-BLEU [Zhu et al., 2018] measures n-gram overlap by scoring each completion against the remaining $N-1$:

$$\text{Self-BLEU} = \frac{1}{N} \sum_{i=1}^{N} \text{BLEU}\big(x^{(i)},\ \{x^{(j)}\}_{j \neq i}\big). \tag{26}$$

A score near 1.0 means completions share identical phrasing; near 0 means entirely different word sequences. This captures phrase-level diversity: candidates may differ at scattered positions yet still produce essentially the same text (high Self-BLEU) or represent genuinely different continuations (low Self-BLEU).

Pairwise edit distance counts the fraction of positions where two completions disagree, averaged over all pairs:

$$\text{EditDist} = \frac{1}{\binom{N}{2}} \sum_{i<j} \frac{1}{L} \sum_{\ell=1}^{L} \mathbf{1}\big[x_\ell^{(i)} \neq x_\ell^{(j)}\big]. \tag{27}$$

This directly quantifies how many token positions are affected by the stochastic jumps—the "material" available to the energy compass for candidate selection.

Token agreement measures per-position consensus by computing the fraction of completions that share the most popular token at each position:

$$\text{Agree} = \frac{1}{L}\sum_{\ell=1}^{L}\frac{1}{N}\max_{v\in\mathcal{V}}\sum_{i=1}^{N}\mathbf{1}\big[x_\ell^{(i)}=v\big]. \tag{28}$$

This complements edit distance by revealing the distribution of disagreement: positions with low agreement are where stochastic jumps are genuinely uncertain, and where the token refinement phase must decide whether to intervene (Appendix C.3).

Table 19 reports these metrics at 8 and 64 steps (bracketing the range of step counts used in our experiments) across seven midpoint times. The diversity profiles are nearly identical across step counts, indicating that midpoint diversity is a property of the flow structure rather than the solver granularity.

Table 19: **Midpoint diversity** (48 prefixes, 10 completions each, uniform source).

	8 steps			**64 steps**		
t_{mid}	Self-BLEU	Edit dist. ($\approx$ pos.)	Agree	Self-BLEU	Edit dist. ($\approx$ pos.)	Agree
0.0	0.146	0.908 (930)	0.242	0.060	0.962 (985)	0.181
0.1	0.688	0.199 (204)	0.860	0.642	0.219 (224)	0.845
0.2	0.749	0.144 (147)	0.900	0.777	0.141 (144)	0.900
0.3	0.864	0.096 (98)	0.933	0.838	0.100 (102)	0.930
0.5	0.946	0.031 (32)	0.979	0.938	0.040 (41)	0.972
0.7	0.981	0.013 (13)	0.991	0.978	0.012 (12)	0.991
0.9	0.993	0.004 (4)	0.997	0.994	0.003 (3)	0.998

Implications for navigation shaping. The results confirm that midpoint diversity decays gradually rather than collapsing abruptly, validating the central assumption of the sequence selection phase. At $t_{\text{mid}}{=}0.2$—the activation threshold used in our main experiments ($\tau{=}0.2$)—independent completions from a shared midpoint differ at approximately 147 out of 1 024 token positions (14.4% edit distance). At $t_{\text{mid}}{=}0.3$, completions still differ at approximately 98 positions (9.6% edit distance). This means that $K{=}5$ candidates generated from the same midpoint will present genuinely different options at $\sim$100–150 positions, giving the energy compass meaningful material to select among.

Even at $t_{\text{mid}}{=}0.5$, where approximately half the sequence is revealed, $\sim$32 positions still vary across completions. These remaining variable positions are precisely the ones most susceptible to compounding errors at subsequent midpoints: they are unrevealed tokens where the stochastic jump makes a consequential choice. Navigation shaping targets exactly these positions—the sequence phase selects the globally best candidate across all variable positions, and the token phase corrects individual misalignments within the winner.

The residual diversity at high t_{mid} ($\sim$13 positions at $t{=}0.7$, $\sim$4 at $t{=}0.9$) may appear negligible, but even a few misplaced tokens in a nearly complete sequence can propagate through subsequent RK-4 stages. The token refinement phase is specifically designed for this regime: it operates at individual positions with velocity-confidence gating, making targeted corrections where the stochastic jump most likely introduced an error (Appendix C.3).

E.3 Scaling to larger models

The main experiments in Section 5 use a 170M-parameter student. A natural question is whether trajectory shaping continues to help as model size grows, or whether a larger velocity model produces trajectories of sufficient quality that navigation shaping becomes redundant. To address this, we repeat the uniform-source comparison with a 1.3B-parameter student—roughly $7.6\times$ larger than the main-text model.

A notable aspect of this setup: the 1.3B student is guided by the same $\sim$90M energy compass used at 170M. The compass is therefore $\sim 14\times$ smaller than the student it supervises. If trajectory shaping

remained useful only when the compass matched the student in capacity, this ratio would be far too unfavorable. Any gains at 1.3B thus also speak to a practical scaling property of the approach: a lightweight compass can guide substantially larger students.

The headline numbers and table for this experiment appear in Section 5.2 and Table 6; this section adds the breakdowns and discussion that did not fit in the main paper.

(1) The TS-DFM advantage over FS-DFM remains at least as large at 1.3B. At 1.3B, TS-DFM at 8 steps reaches 48.0 GPT-2 Large PPL versus FS-DFM's 81.5—a **41% reduction**, compared with the 36% reduction observed at 170M (56.1 vs. 87.6). At 16 and 32 steps the reduction is similar at both scales: 36% at 170M (47.2 vs. 73.2) and 33% at 1.3B (33.1 vs. 49.5) at 16 steps, and 31% and 30% at 32 steps. A larger student does not make trajectory shaping redundant.

(2) TS-DFM beats the $1\,024$*-step DFM teacher starting at 8 steps, across all evaluators.* At just 8 steps, TS-DFM achieves 48.0 GPT-2 Large PPL—below the $1\,024$-step DFM teacher's 57.5—while being $128\times$ faster. The same ordering holds on LLaMA-2 7B (26.6 vs. 32.1) and LLaMA-3 8B (46.7 vs. 54.5). At 16 steps the advantage grows further (33.1 vs. 57.5 on GPT-2 Large, a 42% reduction over the teacher), and at 32 steps TS-DFM reaches 31.5—a 45% reduction over the $1\,024$-step teacher at $32\times$ fewer steps. Every TS-DFM cell from 8 steps onward surpasses the full-step teacher across all three evaluators. The few-step quality advantage demonstrated at 170M is not an artifact of small-scale distillation. It also holds at 1.3B.

(3) A 90M compass effectively guides a 1.3B student—at a fraction of the relative overhead. The energy compass is held at $\sim$90M parameters across both scales, so the compass-to-student ratio shifts from $\sim$1:2 at 170M to $\sim$1:14 at 1.3B. Despite this asymmetry, the compass provides enough discriminative signal to steer the larger student's trajectories, and the 8-step quality gain over FS-DFM is at least as large as at 170M (a 41% reduction at 1.3B versus 36% at 170M). The shift in ratio also sharply reduces the relative training overhead: at 170M each energy forward pass costs roughly $1/1.9 \approx 53\%$ of a velocity pass, while at 1.3B it costs only $90/1300 \approx 7\%$. In other words, the same navigation shaping procedure that added a $\sim 2.2\times$ training-time overhead at 170M (Table 4) becomes proportionally much cheaper at larger scales—navigation's FLOP cost is dominated by the compass, while the bulk of step FLOPs shifts to the (larger) velocity model. From 170M to 1.3B, the method therefore becomes *more* favorable to apply: an 8-step quality gain at least as large, at lower relative cost.

(4) Diversity is preserved at scale. Generation entropy for TS-DFM at 1.3B ranges from 7.0 (4 steps) to 7.6 (32 steps), well above the 6.5 threshold below which token repetition and reduced lexical diversity become prevalent in discrete diffusion models (Appendix E.1). The perplexity improvements at scale therefore reflect genuine quality gains rather than entropy collapse toward repetitive high-probability sequences.

These results extend the main conclusion of the paper to a substantially larger model: the bottleneck in few-step discrete flow matching is trajectory quality, not student capacity. At 1.3B, a student trained with shaped trajectories beats its $1\,024$-step teacher starting at just 8 steps—a $128\times$ inference speedup with *improved* quality—using the same lightweight 90M energy compass used at 170M. At 1.3B, trajectory shaping is at least as effective at 8 steps (41% versus 36% PPL reduction over FS-DFM) and *cheaper* in relative terms (smaller compass-to-student FLOP ratio). The practicality of the method decouples cleanly from the capacity of the supervising signal.

E.4 Comparison to diffusion language models in the few-step regime

We extend the comparison to contemporary diffusion language models—LLaDA [Nie et al., 2025] and Dream [Ye et al., 2025]—in the prefix-conditioned few-step regime, following the protocol of FS-DFM [Monsefi et al., 2026]. This places TS-DFM in the broader landscape of discrete diffusion models and highlights the gap between distilled and undistilled few-step behavior.

Setup. Each model receives a 512-token prefix sampled from WikiText-103 [Merity et al., 2017], and metrics are computed on the 512-token continuation only. We report perplexity (PPL $\downarrow$) under GPT-2, generation entropy (Ent), and MAUVE [Pillutla et al., 2021] (MVE $\uparrow$) at step budgets of 4, 8, and 16. We compare TS-DFM (0.17B) against Dream (7B) and LLaDA (8B), with FS-DFM (0.17B) as the direct distillation baseline. Dream, LLaDA, and FS-DFM numbers are taken from Table 2 of Monsefi et al. [2026].

Table 20: **TS-DFM vs. diffusion language models in the few-step prefix-conditioned regime.** Each method receives a 512-token prefix; PPL (↓), entropy (Ent), and MAUVE (MVE ↑) are computed on the 512-token continuation. Step budgets span $4 \rightarrow 16$. Dream, LLaDA, and FS-DFM numbers are reproduced from Table 2 of Monsefi et al. [2026].

		4 steps			**8 steps**			**16 steps**		
Method	**Size**	ppl	ent	MVE	ppl	ent	MVE	ppl	ent	MVE
Dream	7.0B	752.11	1.53	0.005	739.40	1.74	0.006	630.30	2.31	0.005
LLaDA	8.0B	495.17	0.47	0.005	441.26	0.42	0.005	432.65	0.50	0.005
FS-DFM	0.17B	97.07	7.89	**0.053**	75.78	7.95	0.270	67.42	7.97	0.390
TS-DFM	0.17B	**68.95**	7.70	0.038	**57.11**	7.80	**0.339**	**53.97**	7.86	**0.707**

Observations. *(1) Undistilled diffusion LMs do not reach usable few-step quality.* Across all step budgets, Dream (7B) and LLaDA (8B) sit at PPL above 400 with entropy collapsed to 0.42–2.31 and MAUVE stuck near 0.005, essentially indistinguishable from random. Their entropy values—marked in red in Table 20—fall well below the natural-text range ($\sim$ 7–8 under GPT-2), indicating that both models repeat a small set of high-probability tokens rather than producing diverse continuations. Even with $40\times$–$47\times$ more parameters than TS-DFM, neither model converges to natural-language statistics within 16 steps. This is the regime that distillation is designed to fix.

(2) TS-DFM extends the FS-DFM advantage. At every step count, TS-DFM (0.17B) outperforms FS-DFM (0.17B) on PPL: $97.07 \rightarrow 68.95$ at 4 steps (29% reduction), $75.78 \rightarrow 57.11$ at 8 steps (25%), and $67.42 \rightarrow 53.97$ at 16 steps (20%). The MAUVE improvement is even sharper at higher step counts: $0.270 \rightarrow 0.339$ at 8 steps and $0.390 \rightarrow 0.707$ at 16 steps—a $1.8\times$ improvement that places generations substantially closer to the natural-text distribution. Entropy stays in the 7.70–7.86 range, indicating these gains come without diversity collapse. The only metric where FS-DFM holds a marginal lead is MAUVE at 4 steps (0.053 vs. 0.038); at this extreme step budget, neither method produces enough coherent text for MAUVE to differentiate reliably, and both rise sharply with more steps.

(3) Quality at a fraction of parameter count. TS-DFM (0.17B) reaches PPL 54 and MAUVE 0.71 at 16 steps; Dream (7B) and LLaDA (8B) sit at PPL ≥ 432 with MAUVE ≤ 0.005 at the same step count. The size disparity is not a small-model handicap to overcome but a positive result: a $40\times$–$47\times$ smaller distilled student produces text that the much larger undistilled models cannot match at this step budget. Trajectory shaping, not parameter count, is the lever for usable few-step quality.

Summary. The few-step prefix-conditioned regime separates distilled from undistilled discrete diffusion sharply. Dream (7B) and LLaDA (8B) collapse across 4–16 steps, with PPL above 400, collapsed entropy, and MAUVE near zero. Within the distilled family, TS-DFM (0.17B) extends FS-DFM's advantage with 20–29% PPL reductions and a near-doubling.

E.5 Transfer to mathematical reasoning at 1.3B scale

The experiments in Table 1 evaluate trajectory shaping on unconditional language modeling at 170M parameters. To test whether the gains transfer to a fundamentally different regime—task-specific fine-tuning at larger scale—we apply TS-DFM to mathematical reasoning on GSM8K [Cobbe et al., 2021] and SVAMP (Table 22). We present this as a transfer study of the mechanism: it tests whether trajectory shaping carries over to a task-fine-tuned setting on two math benchmarks.

Setup. We use the 1.3B DFM model with a uniform source distribution, fine-tuned on GSM8K's $7{,}473$ training problems. The pipeline follows the same three-stage structure as Section 5: a pretrained DFM teacher is first supervised fine-tuned on GSM8K with cross-entropy loss; an FS-DFM student is then distilled with step-size conditioning; finally, a TS-DFM student is distilled using the energy-navigated RK-4 procedure from Section 4, with the compass frozen throughout. The step-size conditioning weights of both students are warm-started from an existing FS-DFM checkpoint. We evaluate using *pass@k* for $k \in \{8, 16, 32\}$, estimated via the unbiased estimator of Chen et al. [2021] with 32 independent samples per problem across all $1{,}319$ test problems.

Table 21: **pass@k on GSM8K at 1.3B scale.** Both students maintain strong accuracy down to 4 steps while the teacher degrades sharply. TS-DFM at 4 and 8 steps matches the *full-step* teacher's (256-step, gray row) pass@8 accuracy at up to $64\times$ less inference compute. Cells in **bold** are the better of FS-DFM and TS-DFM at each setting.

Model	Steps	pass@8	pass@16	pass@32
Teacher (full-step reference)				
Teacher	4	4.84	10.30	18.49
Teacher	8	5.68	11.69	20.92
Teacher	32	7.74	15.69	22.16
Teacher	256	9.12	16.39	27.52
FS-DFM				
FS-DFM	4	9.41	15.29	23.05
FS-DFM	8	8.99	15.48	24.94
TS-DFM				
TS-DFM	4	**9.61**	**15.43**	**25.47**
TS-DFM	8	**9.17**	**15.79**	**25.19**

Few-step TS-DFM matches the full-step teacher. The headline result is that TS-DFM at very low step counts matches the teacher's full-step (256-step) accuracy on pass@8: TS-DFM reaches 9.61 at 4 steps and 9.17 at 8 steps, against the teacher's full-step 9.12, with $32\times$–$64\times$ less inference compute. This is the central claim few-step distillation needs to justify itself. The distilled student with a small step budget should not merely *approach* the full-step teacher; it should match or exceed it. TS-DFM clears this bar on pass@8 and reaches 92–93% of the teacher's full-step quality on the harder pass@32 metric (TS-DFM at 4 steps: 25.47; teacher at 256 steps: 27.52). The same-step gap is even more striking: at 8 steps, TS-DFM (9.17, 15.79, 25.19) outperforms the teacher at the same 8 steps (5.68, 11.69, 20.92) by $+61\%$, $+35\%$, and $+20\%$ on pass@8, pass@16, and pass@32 respectively—the difference between a teacher that has not converged and a student that has internalized few-step generation.

TS-DFM matches or exceeds FS-DFM at low step counts. At every cell in Table 21, TS-DFM scores above FS-DFM at the same step count and pass@k metric. The largest gap is at the lowest step budget. At 4 steps, TS-DFM improves over FS-DFM by $+0.20$ on pass@8, $+0.14$ on pass@16, and $+2.42$ on pass@32; at 8 steps, the gains are $+0.18$, $+0.31$, and $+0.25$, respectively. These margins are small relative to binomial sampling noise on $1{,}319$ test problems and are reported without variance, so we read them as directionally consistent rather than as statistically established gains. The pass@32 improvement at 4 steps ($23.05 \to 25.47$) is particularly informative: to gain a comparable amount on pass@32, the teacher requires going from 4 to 32 inference steps ($18.49 \to 22.16$, $+3.67$ at $8\times$ compute) or from 32 to 256 steps ($22.16 \to 27.52$, $+5.36$ at $8\times$ compute). TS-DFM captures roughly half of that latter gain at $64\times$ less inference compute than the teacher's full-step setting. Trajectory shaping, in effect, converts inference-step compute into one-time training-step compute—and the conversion is favorable.

Why trajectory shaping may matter for reasoning. The mathematical reasoning regime is one where each token decision carries unusual weight: a single wrong digit, a misplaced operator, or an inverted comparison can flip the entire answer. This is the structural opposite of unconditional language modeling, where local errors can be absorbed by surrounding context. The hypothesis underlying TS-DFM—that compounding errors at intermediate trajectory midpoints corrupt the student's training signal—predicts that the gains should be *larger* on tasks where trajectory errors propagate into outcome errors rather than cosmetic ones. The pass@32 gain at 4 steps ($23.05 \to 25.47$) is consistent with this, although the other margins are small: trajectory shaping targets the kind of mid-generation drift that turns "48" into "84" or "+" into "−." The energy compass and the Sequence-to-Token policy never see GSM8K problems—the compass is trained on unconditional text—yet the same shaping mechanism transfers to a reasoning task at 1.3B parameters without degrading accuracy. This is what we mean by saying the bottleneck is the trajectory, not the student.

A second reasoning benchmark: SVAMP. To test whether the task-level transfer observed on GSM8K is specific to that benchmark, we repeat the study on SVAMP [Patel et al., 2021], a math word-problem benchmark with 700 training and 300 test problems. The protocol is identical to the GSM8K study above: the same 1.3B pipeline (teacher SFT → FS-DFM distillation → TS-DFM distillation), fine-tuned on the SVAMP training split, guided by the same frozen 90M energy compass. As before, the compass is trained only on general web text and never sees a math problem at any stage—in pretraining, negative construction, or fine-tuning—so any effect on this task reflects transfer of the shaping mechanism rather than task-specific tuning.

Table 22: **pass@k on SVAMP at 1.3B scale** (300 test problems, 32 samples per problem). TS-DFM matches or exceeds FS-DFM at every step count and metric, reproducing the GSM8K ordering on a second verifier-scored task.

Model	Steps	pass@8	pass@16
Teacher (full-step reference)			
Teacher	4	15.2	25.4
Teacher	8	16.9	27.6
Teacher	256	23.4	34.3
FS-DFM			
FS-DFM	4	23.8	32.8
FS-DFM	8	23.2	33.0
TS-DFM			
TS-DFM	4	**24.2**	**33.0**
TS-DFM	8	**23.5**	**33.5**

Three observations. First, the ordering is directionally consistent with GSM8K: TS-DFM matches or exceeds FS-DFM at every step count on both metrics. We note that the per-cell margins over FS-DFM ($+0.2$ to $+0.5$ points on 300 problems) are small relative to binomial sampling noise at this test-set size, so we report this as directional consistency rather than a statistically established gain. The value of the result lies in the same pattern appearing independently on a second task. Second, the few-step advantage over the teacher at matched step counts is large: at 4 steps, TS-DFM reaches 24.2 pass@8 versus the teacher's 15.2 ($+59\%$), and its 4-step accuracy is on par with the 256-step teacher's (24.2 vs. 23.4) at $64\times$ less inference compute. Third, the transfer evidence rests on two independent reasoning benchmarks rather than one, both showing the same pattern with a compass that never saw the task domain—consistent with the compass evaluating general sequence quality rather than task-specific content.

Generalization conclusion. Three claims follow. First, the mechanism transfers: the same energy-navigated RK-4 procedure that improves unconditional generation at 170M parameters carries over to task-fine-tuned mathematical reasoning at 1.3B on GSM8K and SVAMP, with directionally consistent gains over FS-DFM—no methodological modification, no task-specific compass, no per-task tuning. Second, the few-step regime is where shaping matters most: at 4 and 8 steps, TS-DFM matches the full-step teacher on pass@8 on both GSM8K and SVAMP, and recovers 92–93% of full-step pass@32 accuracy on GSM8K at up to $64\times$ less compute. Third, the bottleneck for few-step reasoning is the same as for few-step language modeling: trajectory quality, not student capacity. The same lever pulls in both regimes.

E.6 Variance across decoding seeds

The main tables report point estimates from a single generation run per cell. To quantify sampling variability, we re-evaluate with 5 independent decoding seeds per cell (full $1\,024$-token unconditional generation, fixed model checkpoints) and report mean $\pm$ standard deviation: Table 23 covers the central 170M comparison of TS-DFM against FS-DFM across both source distributions and initializations, and Table 24 covers the state-of-the-art comparison against SDTT and Duo under the same protocol. We emphasize the scope of this analysis: because the checkpoints are fixed and only the decoding seed varies, these intervals bound *sampling variance at a fixed checkpoint*, not variance of the method across independent training runs.

Table 23: **Sampling variance at fixed checkpoints** (5 decoding seeds per cell; mean $\pm$ std). 170M models, 1 024-token unconditional generation.

Source	Model	Steps	$G^2\downarrow$	$L^2\downarrow$	$L^3\downarrow$
Uniform	TS-DFM (FS-DFM init)	8	56.97 ± 1.48	34.82 ± 1.79	58.63 ± 3.91
		16	45.73 ± 1.27	27.01 ± 0.90	45.08 ± 1.42
	TS-DFM (DFM init)	8	79.96 ± 1.51	48.23 ± 0.48	80.03 ± 1.49
		16	69.17 ± 1.55	42.06 ± 1.14	69.57 ± 1.35
	FS-DFM	8	88.23 ± 1.50	56.26 ± 3.53	102.31 ± 4.97
		16	73.67 ± 2.75	43.98 ± 2.61	78.69 ± 5.66
Mask	TS-DFM (DFM init)	8	92.76 ± 1.14	74.04 ± 0.74	90.27 ± 1.03
		16	60.97 ± 1.00	45.88 ± 0.75	62.75 ± 1.13
	TS-DFM (FS-DFM init)	8	189.66 ± 2.13	94.95 ± 0.96	178.63 ± 1.83
		16	109.64 ± 1.42	54.39 ± 0.59	101.32 ± 1.14
	FS-DFM	8	515.75 ± 8.33	202.90 ± 3.61	490.22 ± 9.05
		16	276.35 ± 2.95	120.87 ± 1.34	265.15 ± 2.75

Table 24: **Sampling variance for the state-of-the-art comparison** (uniform source, GPT-2 PPL $\downarrow$; 5 decoding seeds per cell; mean $\pm$ std).

Method	8 steps	16 steps
SDTT (5 rounds)	193.58 ± 1.62	87.53 ± 0.53
SDTT (7 rounds)	108.63 ± 1.13	54.95 ± 0.53
Duo	71.69 ± 0.66	54.70 ± 1.15
FS-DFM	88.23 ± 1.50	73.67 ± 2.75
TS-DFM (ours)	$\mathbf{56.97 \pm 1.48}$	$\mathbf{45.73 \pm 1.27}$

Across both tables, the margins between methods are large relative to decoding variance: in Table 23, the TS-DFM and FS-DFM intervals are separated by many standard deviations in every configuration, and in Table 24, TS-DFM remains below Duo and SDTT (7 rounds) by gaps far exceeding the seed spread at both step counts. Welch two-sample tests on the GPT-2 perplexities (5 seeds per side) confirm this: TS-DFM vs. FS-DFM in the main uniform configuration, $t = 33.2$, $p \approx 10^{-9}$ at 8 steps and $t = 20.6$, $p \approx 10^{-6}$ at 16 steps; for the mask source, all TS-DFM vs. FS-DFM comparisons exceed 50 pooled standard deviations ($p \leq 10^{-7}$); against the strongest external baselines, TS-DFM vs. Duo gives $p \leq 3 \times 10^{-6}$ and TS-DFM vs. SDTT (7 rounds) gives $p \leq 2 \times 10^{-5}$ at both step counts. Under decoding variance at fixed checkpoints, the orderings in Tables 1 and 5 are therefore not sampling artifacts.

E.7 Energy-guided teacher sampling

A natural question about the student-beats-teacher results in Table 1 is whether the shaped student is genuinely stronger than the teacher, or whether it merely *amortizes* a stronger sampling procedure—the energy-guided navigation—that the teacher could equally apply at inference time. To separate these readings, we equip the 1 024-step teacher itself with the compass: the same NAVIGATE operator (K=5, identical constants) is applied during teacher sampling at two guidance densities, every step and every 8th step. Table 25 compares these guided teacher samplers against the blind (compass-free) TS-DFM students under the setting of Table 1 (uniform source, 170M student, 1 024-token generations).

The compass makes the teacher a substantially stronger sampler: dense guidance improves GPT-2 perplexity from 83.98 to 49.03. The key result, however, is that the blind 16-step student surpasses even the most strongly guided teacher (45.73 vs. 49.03 on GPT-2; 27.01 vs. 31.13 on LLaMA-2) at $\sim 0.2\%$ of its inference cost—with no compass at inference. This rules out the pure-amortization reading: a student that merely amortized the guided sampler would approach it from below, whereas the student exceeds it. Training-time shaping integrates the corrections into the weights rather than re-applying them per step. Two further observations sharpen the claims: (i) sparse guidance (every 8th step) recovers only $\sim 24\%$ of the dense-guidance gain, independently confirming that errors compound between guided steps—the mechanism motivating TS-DFM in Section 3; (ii) stated precisely, the 8-step student surpasses the blind 1 024-step teacher, and the 16-step student exceeds even the compass-guided teacher at three orders of magnitude lower cost.

Table 25: **Energy-guided teacher sampling vs. blind TS-DFM students** (uniform source). Guided samplers apply NAVIGATE during teacher inference at the stated density; TS-DFM uses no compass at inference. Latency is per 1 024-token generation on a single GPU. Teacher rows are a separate evaluation with 64 samples per cell. TS-DFM rows are 5-seed means from Table 23. Values therefore differ slightly from the point estimates in Table 1.

Sampler	Steps	$G^2\downarrow$	$L^2\downarrow$	$L^3\downarrow$	Lat. (s)	vs. TS-DFM
Teacher, Inference blind	1 024	83.98	47.38	79.84	18.9	105×
Teacher + NAV. (every 8th)	1 024	75.66	46.77	75.43	41.2	230×
Teacher + NAV. (every step)	1 024	49.03	31.13	46.45	186	1 038×
TS-DFM, Inference blind	8	56.97 ± 1.48	34.82 ± 1.79	58.63 ± 3.91	0.18	1×
TS-DFM, Inference blind	16	45.73 ± 1.27	27.01 ± 0.90	45.08 ± 1.42	0.36	2×

E.8 Inference-time deployment of the compass

Training-time shaping is one way to deploy the energy compass; an alternative family keeps the student fixed and applies the compass at inference. To isolate how much of TS-DFM's gain comes from training-time trajectory shaping rather than from the learned energy model itself, we compare four inference-time deployments on the fixed, unshaped FS-DFM student (uniform source, GPT-2 PPL), all using the same frozen compass: *best-of-5 rerank* (5 complete blind generations; the compass scores the finished sequences and returns the lowest-energy one—endpoints only); *final-step selection* (blind until the last step, where the compass selects among 5 candidate final transitions—one intervention point); *full* NAVIGATE (the complete S2T policy at every inference step, as in Table 16); and *stacking* (full NAVIGATE applied on top of the already-shaped TS-DFM student).

Table 26: **Inference-time compass deployments vs. training-time shaping** (uniform source, GPT-2 PPL $\downarrow$). All compass-using rows share the same frozen 90M compass. Latency is per 1 024-token generation. This is a separate evaluation from Table 16, so its FS-DFM rows differ from that table. TS-DFM blind rows are 5-seed means from Table 23.

Configuration	8 steps	16 steps	Lat.@8 (ms)	Lat.@16 (ms)
FS-DFM, blind (deployed cost)	89.30	63.24	179	359
FS-DFM + best-of-5 rerank	80.41	63.49	940	1 574
FS-DFM + final-step selection	81.36	64.67	380	771
FS-DFM + full NAVIGATE	80.18	57.36	1 266	2 789
TS-DFM (FS-DFM init), blind	56.97 ± 1.48	45.73 ± 1.27	179	361
TS-DFM (DFM init), blind	79.96 ± 1.51	69.17 ± 1.55	176	358
TS-DFM (DFM init) + full NAVIGATE	78.47	59.26	1 196	2 396

The pattern in Table 26 is sharp. The three inference-time deployments give the energy model every post-hoc opportunity—endpoints, the final transition, every step—and all plateau near 80 PPL at 8 steps. Blind TS-DFM, with no compass at inference, reaches 56.97 at the cost of the blind baseline; no inference-time deployment comes within 23 PPL. What separates the rows is not access to the compass (all have it) but *when it acts*.

Why training-time guidance at 8 steps beats inference-time guidance at 8 steps. The two act at different granularities. Table 16 shows the compass's effect scales with the number of decision points: at 8 steps it improves the blind sampler by only $\sim 4\%$ ($89.6 \to 86.2$)—each step is a large jump, with little to select between—while at 64 steps it improves it by $\sim 49\%$ ($75.0 \to 38.0$). Inference-time guidance at the deployed budget is locked to the large-jump regime where the compass is weakest. Training-time shaping is not: RK-4 target construction decomposes each large jump into small sub-steps (the $h/2$ midpoints, corresponding to the 16–64-step regime where guidance is strongest), NAVIGATE acts at that granularity, and the KL objective distills the guided small-step behavior into the student's large-step predictions. The student learns what a large jump should land on from supervision assembled out of well-guided small jumps—quality that ends up in the weights and is free at 8 steps.

Stacking. Applying full NAVIGATE on top of the shaped student yields the expected residual: under 2% at the deployed 8-step budget ($79.96 \rightarrow 78.47$) but 14% at 16 steps ($69.17 \rightarrow 59.26$) for the weaker DFM-init student—consistent with the same account, since more decision points make inference-time guidance more effective. Shaping and inference-time guidance are thus complementary rather than redundant, but shaping alone captures the bulk of the benefit at the deployed budget.

E.9 Sensitivity of navigation constants

The two navigation constants with the largest quality effect—K and τ—are searched and ablated in Appendix D.3 and Section 5.2. The remaining constants (Table 14) are set structurally rather than searched. Among them, we expect β_0 to be the most impactful, as it directly scales the token-refinement strength (Equations (22) and (23)), and ϵ_{safe} the least, as it acts only on the $<8\%$ of midpoints where the safeguard fires (Appendix C.4). We sweep these two, one factor at a time around the defaults, holding all other constants fixed.

The sweep applies the identical NAVIGATE operator *at inference* on the fixed FS-DFM student (uniform source, GPT-2 PPL, 320 samples per cell). This design isolates *operator sensitivity*—how the navigation procedure itself responds to the constants—and does not measure sensitivity of the shaped student's training-time gains, which would require retraining a TS-DFM student per setting.

Table 27: **One-factor-at-a-time sensitivity of β_0 and ϵ_{safe}** (NAVIGATE at inference on the fixed FS-DFM student; uniform source, GPT-2 PPL $\downarrow$, 320 samples per cell). All non-swept constants at their defaults; the all-defaults configuration is a single run appearing as the center of each group. This is a separate evaluation from Table 16, so its blind and default rows differ from that table.

Swept	Value	8 steps	16 steps	32 steps
—	blind (no compass)	90.94	67.58	71.97
β_0	0.1	86.74	64.18	57.69
	0.3 (default)	88.87	58.95	53.52
	1	85.92	56.39	52.79
ϵ_{safe}	0.02	85.98	59.06	53.80
	0.1 (default)	88.87	58.95	53.52
	0.5	85.07	58.25	54.55

The spread in Table 27 is narrow: across a $10\times$ range of β_0 and a $25\times$ range of ϵ_{safe}, every setting improves over the blind baseline at every step count, so the operator's benefit does not depend on hitting a tuned optimum. We note that $\beta_0{=}1$ slightly outperforms the default at all three step counts; we retain $\beta_0{=}0.3$ as the (conservative) default used in all main experiments, and read the sweep as evidence of robustness rather than an invitation to tune. ϵ_{safe} is, as predicted, nearly inert. The temperature-range constants ($T_{\min}$, Δ_T, $\lambda_{\min}$, $\lambda_{\max}$) remain unswept, as the caption of Table 14 notes.

E.10 Compass robustness under student distribution shift

The energy compass is trained once and frozen while the TS-DFM student's flow-state distribution evolves over distillation (Section 6). If the student drifted away from the compass's training distribution, the compass could keep selecting midpoints that were useful under the original distribution but are misaligned with the student's current trajectories, especially late in training. We measure this directly: the frozen compass is evaluated on the TS-DFM student's *own* flow states across saved training checkpoints (10k–130k iterations, every 20k), using the DFM-initialized uniform 170M run—the longest path from initialization to convergence, and thus the worst case for drift.

Table 28 tracks four quantities. *Compass accuracy*: how often the compass correctly ranks a clean state below a corrupted copy of itself (the discrimination task it was NCE-trained for; corruption here is random token replacement, the model-independent probe). *Energy improvement*: the energy drop from the current state to the compass-selected best of $K{=}5$ candidates—the direct measure of guidance strength, where a stable value is the healthy outcome and a downward trend would indicate signal loss. *Refinement acceptance*: the fraction of token-phase edits passing the energy safeguard. $\bar{E}_{stu} - \bar{E}_{real}$: the mean energy gap between the student's states and real interpolated data, where

convergence toward zero indicates the student approaching the distribution on which the compass's positives were built.

Table 28: **Frozen-compass diagnostics on the TS-DFM student's own flow states across training** (DFM-init uniform run, 170M; worst case for drift). A stable energy-improvement column is the healthy outcome.

Ckpt iter	Compass acc.	Energy improv.	Refine accept	$\bar{E}_{\text{stu}} - \bar{E}_{\text{real}}$
10k (burn-in)	0.799	0.166	0.804	$+1.98$
30k	0.982	0.101	0.929	$+0.23$
50k	0.978	0.131	0.982	-0.05
70k	0.991	0.126	0.987	-0.06
90k	0.982	0.121	0.978	-0.08
110k	0.987	0.113	0.991	-0.08
130k	0.969	0.127	1.000	-0.04

The result: **no measurable degradation on these metrics over the full run**. Compass accuracy holds at 0.97–0.99 from 30k onward, guidance strength is flat at 0.10–0.13 from 30k to 130k with no downward trend, and safeguard acceptance rises to 0.93–1.00. The distribution shift itself is real, but directional—and the direction is benign: the $\bar{E}_{\text{stu}} - \bar{E}_{\text{real}}$ column shows the student's states converging *toward* the real-data energy level ($+1.98$ at 10k to $|\Delta| \leq 0.08$ from 50k on), i.e., toward the energy level of the interpolated real text on which the compass was trained, not away from it. The hypothesized failure mode—midpoints useful under the original distribution but misaligned with the student's current trajectories late in training—would appear as declining selection strength and falling acceptance; both are flat-to-rising through 130k. An independent rerun of the full diagnostic agrees within sampling noise.

Two caveats bound the conclusion. First, the accuracy column uses random token replacement, the simplest of the negative strategies in Table 8 (98.6% held-out accuracy in Table 13), and is therefore a coarse probe of drift. Second, one model-dependent probe does decline over training: detection of perturbations constructed by resampling a state's revealed tokens from the student's own logits falls from 0.674 to 0.245, in step with the closing $\bar{E}$ gap. This decline admits two readings that the diagnostic cannot separate—the student's resampled perturbations become genuinely uncorrupted text as the student converges (no corruption remains to detect), or the compass discriminates less well on the student's own distribution. The model-independent probe holding at 0.97–0.99 from 30k onward supports the former, but a definitive answer would require a compass-refresh or EMA-recompute control, which we have not run. We list it as the missing control in the Limitations paragraph (Section 6).

F Qualitative examples

This section presents uncurated generated text samples and a generation trajectory for qualitative inspection. All outputs are generated by the 170M-parameter model with $1\,024$ tokens from a random source state.

F.1 Generated samples

The metadata header of each sample indicates the source distribution (uniform or mask), generation method (DFM, FS-DFM, or TS-DFM), and number of inference steps.

Source: Mask **Method: TS-DFM** **Steps: 16**

When applying soil Hollis, collect the necessary information on the flower to determine the suitable stem dimension for their needs. This can include analyzing factors such as temperature, pH, air and dioxide, and color on the stem to determine a specific nutrient solution required for flower growth and development.
Using Growth Data and Application techniques:
To illustrate the importance of determining light quality on soil Hollis, let's examine different flowering process testing techniques and application methodologies. One of the primary steps is to test flowering cultivars to measure the specific amounts of nitrogen and phosphorus. The data which these tests provide will help plants adjust the nutrient solution for each cultivar and application based on various factors involved, such as temperature, humidity, soil texture, soil moisture, and soil temperature. Additionally, growth data that be collected on fertilizing conditions and related rooting hormone supplementation, ensuring that the nutrient solution reaches the specific stages of growth. By reducing risk to these factors, plants can be able to determine the best nutrient solution for specific flower growth and

adjust their application accordingly.
Preparation on Soil Hollis:
Consideration on soil color is essential for the improved the light quality and appearance of the flower. Some steps include:
- Testing pH Levels: Conducting tests for soil nutrient needs prevents activity and determining if a pH above ppm is most likely for plant growth development or decline. This may involve applying a balanced pH solution such as mild sulfur,, which can increase the pH of vegetables, fruits, and flowers
Using Proper Techniques: Fertilizing with high-quality fertilizers increases risk of nutrient runoff and water treatment. This can also lead to root damage, and it must be managed to prevent overwatering
Monitoring pH Levels: The nutrient solution should be monitored by local coloring levels. pH levels, regular testing and maintenance to determine the ideal range for growth. and incorporation of tests solutions, such as eremosiloba, to maintain a balanced pH contributes to root development, healthy root and fiber growth.
Understanding Color Depth Distribution: Maintaining a balanced depth between the roots, including the leaves and stems, helps prevent injury, as the leaves facilitate air circulation and ensure light absorption. By ensuring proper distribution to the roots, plants measure their nutrient needs and support root growth and development.
Practicing Pest Control Regular: Regular pest control practices on soil Hollis can reduce the number of pests and can maintain the integrity of the soil. This includes regular inspections to promote healthy root growth, transportation to the roots, and optimizing nutrient distribution.
Improving Application Schedule: Creating a nutrient application schedule is essential in maintaining good soil nutrient health. Make sure that applying nutrients at the given application is properly so that the roots develop roots and then absorb them correctly.
Proper aeration: The soil hole should be amended with a balanced nutrient solution to aer soil, promote growth, optimize aeration, help distribute the nutrient solution, and adjust pH levels accordingly.
Remember, requirements for aeration can vary and various factors such as air quality, current soil conditions, and soil structure should be considered thoroughly before creating hole design.
Overall, a successful soil Hollis requires regular consultation and maintenance is essential for the success of this tool, and can vary significantly on several factors. Check the nutrient solution based on specific water and soil requirements, and adjust their application accordingly. Soil aeration and proper soil moisture prevent the effective application of the roots.What Are Kids History Works Worksheets?
- We show upcoming Kids History Works Worksheets on subjects via our Online Platform All Oregon, North Dakota
- We Try Kids History Works Worksheets on our Online Platform All Oregon November 21, 2023
- The Kids History Academy curriculum ranks across the increasingly modern countries. You've found more than 2,000 applications and would be happy to see if you can use this website if you have over 100.- Lithamic disc discs a rehabche disc that helps distribute nerve cells and helps protect the body from harms and that are free radicals.
- Most medical doctors and neurologists understand that Lithamic disc disc had anti-inflammatory and anti-inflammatory properties and maximize its side effects.
- Studies of researchers have shown that Lithamic disc pain causes include drug therapy, pain disorders and nerve pain pain, muscle and joint pain and pain, and high blood pressure.
- Chinglich disc discs are found in sciatica that disperse nerve cells to the spinal cord, and from any of them, it may cause pain and discomfort.
- Chinglich disc is a large sodium channel that travels the proper amount of sodium through the brain and your skull can absorb radio waves from it.
- When birth, our body weight is shorter growth, thyroid dysfunction, nephropin, and the prevalent side prone bone loss.

Source: Uniform **Method: TS-DFM** **Steps: 8**

In the movement, there is an awareness of the size and breaking cubes of the elder jammed by means of theaida of the People's Liberation Rosesi unit.
Now, researchers are trying to incorporate gender into response to the Switch Woman, the "first project surveys will identify women between 15 and 20% men and 100% men between 30-60. However, it should be noted that people between the age of 40 will only be affected.People who suffer from bruxism are affected that different teeth.
This includes include this, are soft, inflamed, irritated, swollen, stripe-bruising and also may not require comprehensive orthodontic orthopedic care, the diet, or foxesgut.
Treatment of teeth may include new tooth resins, shrinkage and focal balances.
- Tooth density
- Material mutation process
Like speech, a treatment decision is made based on the combination of factors that may lead to bruxism depending on the individual, but not to their diet or foxesgut.
- Before purchasing any hard- or hard, it will be you to select the best choice.
- Once the established type of tooth, care will be undertaken in a village centre.
- If concerned if your patient has a difficult to swallow, be less unlikely to treatment.
- You may also consider other treatment options at MeatAnimal Animalcare
Here is are an overview of treatment options available at Meat Animalcare centres.
- Assistsists provide care for 24 hours prior to delivering the services.
What information printed below.
Different types treatment therapy including conventional therapy.
Inventional therapy is be about preserving tooth, and preserving it, Even if it is severe, dentist may recommend dental care or observation care simply in village centre care. Depending on the severity of your condition, treatment include endodontic surgery, this surgery using an incision in the mouth of you and with blood possessing equipment.
The types of bruxism are often many, but treatment should vary depending on how the tooth is visualized by each individual. The construction of a filling should be tried and the dentalesthetic should be used for it to accumulate.
Other options more include ball-applied materials used for brushing or flossing, machinery and toothletting. stable materials that can be weight are also a great idea as a conventional treatment, as some drinks can contain more than a species of materials the material is used. If bruxism is not used for its filling, the crown should be placed in the mouth to prevent infection.
Whatever information is speak about, your specialist should vary depending on your specific patient. But remember: what is with patients love. Veterinists often take an eye out for the largest number of individual teeth during bruxism screening and notice no age that the right tooth has fallen out.
When eating, these experts should be aware of how to clean and make sure your oral hygiene is comfortably and children more likely develop the condition than adults.
Other types teeth include:
The size and height the tooth, known is the number of is teeth can affect health thebone, gums and overall teeth. The larger the number of teeth the more space in the tooth should fill, which also can lead to a root canal, sleep apnea and sleep issues.

Other types treatments can include:
1. a preventative root canal and people who need to also be carefully about with this since they may feel a sore jaw, which leading to increased sensitivity. There are some ways to prevent teeth from leaving them poorist health they could feel pulled into space tooth and comforted in eating, avoiding hard foods, and a feeling of being.
2. a tooth is likely to develop decay if it is not reaching the point where it will fall out and eventually become smaller and bigger. However, there are also associated with problems take for dental caries when they get older.
3. they may experience some headaches from decay tobacco, as well as if they are brushing or flossing and when sleep is noisy and is during the daytime, people and animals are more susceptible to falls. However, there is a changing time contact falls are treated as part in a preventative treatment on walls and floors.
4. bite and teethanus teeth guards are another serious mouth problem during sleep. They can occur when you bite a dentist with a bullet. Some teeth problems such as loud biting or loose teeth and jaw infections can have some effect of the guards, thus trauma that their way in.
5. people with fractured, teeth should be taken and treated right up.
Sleep apnea is a breathing disorder that affects sleep. Additionally, apnenea due to the weakness of the respiratory muscles can also affect sleep overall sleep quality How people specialize in teeth and foxes ayurved Sleep apnea can affect your sleep a and and severity.

F.2 Generation trajectories

To illustrate how discrete flow matching progressively reveals text, Figure 6 shows a generation trajectory from the mask source at selected steps. Each snapshot shows the first $\sim$200 tokens of the $1\,024$-token sequence. Gray blocks ( ) represent unrevealed [MASK] positions; black text shows previously revealed tokens; **blue bold** text highlights tokens newly revealed at that step. As the flow progresses from $t{=}0$ to $t{=}1$, the sequence transitions from pure [MASK] tokens to coherent (if imperfect) text.

Source: Mask **Method:** TS-DFM **Steps:** 16 **Tokens shown:** first 200 of 1 024

Step 0/16 ($t = 0.0$) **0% revealed**

. . .

Step 4/16 ($t = 0.25$) **∼25% revealed**

landscape of as to customer particular product make , they on is 50 percent This topic is is Tem has . important developments focus profile cutting together for factor the · · ·

Step 8/16 ($t = 0.5$) **∼50% revealed**

related landscape of as **economic tools to develop and the** customer **a** particular **product to help in** product make **consumers for , or to and products that** they **have taste . placed** on **consumers** is to **for** 50 percent **this** This topic is **important as** it is **Future Templates** has you. important that developments focus **a** profile of and cutting **, and putting** together **framework** to **through** for **product development .** · · ·

Step 12/16 ($t = 0.75$) **∼75% revealed**

related **products in** the landscape **and** of, **served** as economic tools to develop and **experiences** the customer **with** a particular product to help in the **purchase** product,, make consumers for the , or to and purchase products that they have taste **or smell. The value** placed on consumers is to **account for around** 50 percent this This topic is as important as it is **within significant ITS** Future Templates has you. important **dimension** that **industrial** developments **often** focus on **is** a **product** profile of and cutting **techniques,** and putting together a framework to **go** through the for product development. · · ·

Step 16/16 ($t = 1.0$) **100% revealed**

and related products in the **business** landscape and of, served as economic tools to develop and **tailor the** experiences **for** the customer with a particular product to **assist and** help in the purchase **of the** product,, make consumers **care** for the **product**, or to and purchase products that they have **register** taste or smell. The value placed on consumers is **estimated** to account for around 50 percent of this **value**. | This topic is as important as it is within the significant **ITS** Future Templates has for you. **An** important dimension that industrial developments often focus on is **developing** a **new** product profile of **cutting** and cutting techniques, and putting together a framework to go through the **negotiation process** for product development. **Second, qualitatively defining and interpreting key conclusions factor in the development of a new development plan.** · · ·

Figure 6: **Generation trajectory (mask source, 16 steps).** The sequence evolves from pure [MASK] tokens (gray blocks) to completed text. Gray blocks ( ) = unrevealed positions; **blue bold** = tokens newly revealed at this step; black text = previously revealed tokens. Only the first ∼200 tokens are shown. Note how the model first reveals high-frequency function words and content words at scattered positions (steps 1–4), then progressively fills in context-dependent tokens (steps 8–12), with the final steps resolving remaining ambiguities.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: All claims are backed by experimental results, and their scope (two model sizes, the uniform source only at 1.3B, two downstream tasks) is stated in the Limitations paragraph of Section 6.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.

- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Included with Conclusion

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not present theoretical results.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Training setup including hyperaramenters, GPUs, optimizers, and data sets are included in the main paper and the appendices.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The paper does not promise open code and data at this point.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Provided in the main paper and appendices.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Appendix E.6 reports mean $\pm$ standard deviation over 5 decoding seeds at fixed checkpoints for the main 170M comparison and the state-of-the-art comparison, with Welch two-sample tests. These intervals capture sampling variance at fixed checkpoints only, not variance across training runs. The main-text tables report single-run point estimates, and the GSM8K and SVAMP results (Appendix E.5) are reported without variance and described as directionally consistent.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: In the setup description and appendices.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: N/A

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [No]

Justification: The paper focuses on improving LLM generation speed and makes no other changes that would require discussion of impacts.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not claim to release code or data.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All methods and datasets are cited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.