Sticky Jump Diffusions: A Unifying View of Masked, Continuous, and Hybrid Diffusion

Pascal Jutras-Dubé
Department of Computer Science
Purdue University
pjutrasd@purdue.edu

Patrick Pynadath
Department of Computer Science
Purdue University
ppynadat@purdue.edu

Jeremy Lu
Department of Computer Science
Purdue University
lu1008@purdue.edu

Yuan Gao
Department of Mathematics
Purdue University
gao662@purdue.edu

Ruqi Zhang
Department of Computer Science
Purdue University
ruqiz@purdue.edu

Abstract

We introduce *sticky jump diffusions* (SJDs), continuous-time Markov processes on $\mathbb{R}^d$ whose discrete anchors are token embeddings. In forward time, anchors release their mass at a hazard rate and the released mass diffuses in the continuous ambient space; time reversal couples a score-driven SDE with a sticky jump kernel whose rate and destination are fixed by flux balance with the forward law. We estimate the score and the per-anchor reverse hazards from a single denoising classifier via *denoising hazard matching*, the hazard analogue of denoising score matching, with simulation-free cross-entropy training. SJD recovers masked diffusion, continuous diffusion, and hybrid diffusion as limits. Its reversal explains features that each family treats as given: the mask of masked diffusion carries no evidence about the source token because the unsticking kernel of every anchor collapses to the same absorbing point; the terminal projection of continuous diffusion is required due to the absence of atoms in its forward marginal, without which flux balance yields no reverse jumps; and the update rules of hybrid diffusion (commit rate, destination, and drift) all follow from flux balance rather than from separate design. Beyond these limits, the unsticking kernel becomes a design space: a cross-position blending corrupts each position toward a blend of its neighbors' clean values or embeddings, turning dependency structure such as spatial locality or a constraint graph into an inductive bias of the corruption itself, and improves over the identity-kernel hybrid on CIFAR-10, Text8, and Sudoku. Our code is available at `https://github.com/PascalJD/sticky-jump-diffusions`.

1 Introduction

Diffusion-based generative modeling has been extended from continuous data [Sohl-Dickstein et al., 2015, Song and Ermon, 2019, Ho et al., 2020, Song et al., 2021, Karras et al., 2022] to discrete sequences such as text. Two families dominate the discrete setting. *Masked diffusion* operates on the token space and outputs valid tokens by construction, but each position is mapped to a single

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

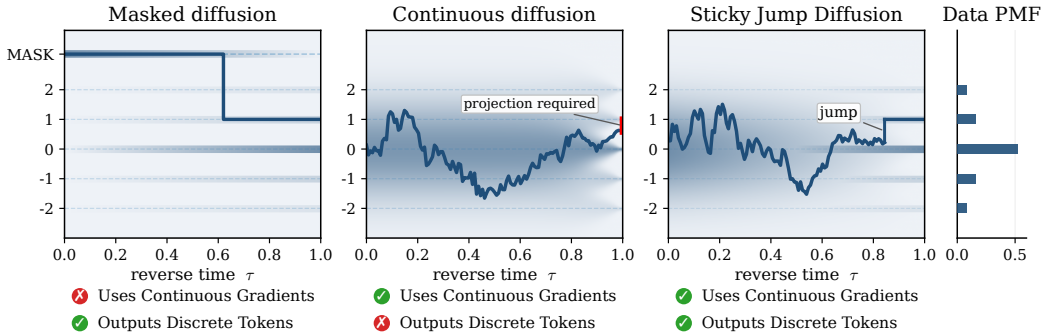

Figure 1: Reverse-time trajectories over each model's path space. Masked diffusion is discrete but gradient-free; continuous diffusion is gradient-based but requires a terminal projection; SJD couples ambient gradients with sticky jumps onto anchors.

absorbing token, and the per-position factorization erases graded information about how close the corruption is to a candidate token [Austin et al., 2021, Sahoo et al., 2024, Shi et al., 2024, Lou et al., 2024]. *Continuous diffusion* operates on token embeddings and uses standard score-based learning, but its terminal state lies off the discrete support and must be mapped back by a projection step external to the dynamics [Li et al., 2022, Dieleman et al., 2022, Chen et al., 2023]. Recent *hybrid diffusion* pairs the two: a masked chain decides which positions are committed, and a continuous latent carries graded information for the masked ones [Zheng et al., 2026, Pynadath et al., 2026, Zhou et al., 2026]. A review of these three families is given in Appendix B.

We define *sticky jump diffusions* (SJDs), a class of continuous-time Markov processes on a continuous state space $\mathsf{X} = \mathbb{R}^d$ with a finite or countable set of anchors $\mathcal{A} \subset \mathbb{R}^d$ identified with token embeddings. All three families above arise as limits of SJDs. In forward time, data start on anchors; an anchor releases its mass at a hazard rate $\vec{\lambda}_t$ into the continuous region $\mathsf{X}_{\mathcal{A}}$, where the trajectory diffuses by a standard SDE. The marginal p_t splits into a continuous density on $\mathsf{X}_{\mathcal{A}}$ and atoms on $\mathcal{A}$, generalizing the classical sticky-boundary diffusions of a single absorbing point on the half-line [Feller, 1952, 1954, Itô and McKean, 1963] to anchor sets of vocabulary size living in $\mathbb{R}^d$. Figure 1 visually compares SJD with masked and continuous diffusion.

Time reversal of an SJD produces a generative process with two coupled mechanisms: a score-driven SDE on $\mathsf{X}_{\mathcal{A}}$ and a sticky jump kernel that commits mass to anchors. The per-anchor reverse intensities are fixed by flux balance [Conforti and Léonard, 2022], so the choice of where to commit and at what rate is determined by the forward law. Token commitment is therefore a reverse-time jump event of the stochastic process, with a rate and a destination that the dynamics specify, rather than a hand-designed schedule applied at sampling time.

To turn the reverse-time identity into a usable generative model, we introduce *denoising hazard matching* (DHM), the hazard analogue of denoising score matching [Vincent, 2011]: the reverse hazard equals the conditional expectation of a forward-known target under the SJD corruption. Combined with a denoising classifier P_θ, DHM yields both the score and the per-anchor reverse intensities from the same network, by an analytic reweighting of P_θ that is Rao–Blackwellized and needs no separate hazard head. Training is plain cross-entropy on SJD-corrupted pairs and is simulation-free under standard conditions on the forward dynamics.

The same construction places existing model classes inside SJD, and each limit inherits an explanation from the reversal. In masked diffusion, the unsticking kernel collapses to the absorbing token, and commitment is forced to be time-only. In hybrid diffusion, the kernel matches the forward perturbation kernel: commit rate, destination, and drift follow from flux balance. In continuous diffusion, there is no sticking jump: without atoms in the forward marginal, there are no reverse jumps, so a terminal projection outside the dynamics is needed. The intermediate regime is the one we study empirically: the unsticking kernel there carries a cross-position blending toward a weighted combination of neighbors' clean values or embeddings, which improves over the identity-kernel hybrid on CIFAR-10, Text8, and Sudoku.

Contributions.

(i) **The SJD process class.** We introduce sticky jump diffusions, continuous-time jump-diffusions on $\mathbb{R}^d$ whose forward marginal splits into a density on a continuous off-anchor region and atoms on a discrete anchor set identified with token embeddings.

(ii) **Reverse dynamics from a single classifier.** We derive the time reversal process of SJD: a score-driven SDE coupled with a sticky jump kernel whose per-anchor rates are fixed by flux balance. Both the score and the reverse hazard are estimated by a denoising classifier through denoising hazard matching, the hazard analogue of denoising score matching, an analytic Rao-Blackwellized reweighting that needs no second network and trains by cross-entropy.

(iii) **A unifying view of masked, continuous, and hybrid diffusion.** SJD recovers absorbing-state masked diffusion, hybrid diffusion, and continuous diffusion as limits. Where these families assemble the reverse sampler by hand, SJD makes the discrete and continuous coordinates one reversed state: commit rate, destination token, and drift all follow from the forward corruption by flux balance.

(iv) **The jump kernel yields a new design space.** We empirically study cross-position jump kernels: each position's corruption is centered on a blend of the other positions' clean values or embeddings rather than its own embedding alone. We instantiate this on three data types whose structure it can express, encoding spatial locality on CIFAR-10, local sequential correlation on Text8, and the row/column/box constraint graph on Sudoku; it improves FID, valid-word generation, and board completion over the identity-kernel hybrid under matched settings.

2 Sticky jump diffusion (SJD) models

Let the state space be $\mathsf{X} = \mathbb{R}^d$ and let $\mathcal{A} = \{a_k\}_{k=1}^K \subset \mathbb{R}^d$ be a finite (or countable) set of anchors. We write $\mathsf{X}_{\mathcal{A}} := \mathbb{R}^d \setminus \mathcal{A}$ for the continuous region and allow the law p_t of X_t to have an absolutely continuous part on $\mathsf{X}_{\mathcal{A}}$ and atoms on $\mathcal{A}$. Notation and assumptions are introduced throughout the text and reiterated in Appendix A. Proofs are in Appendix C.

2.1 Forward noising dynamics: unsticking

SJDs are defined for a single particle in what follows; a running example lifts the construction to sequences and is carried throughout the paper.

Definition 2.1 (Forward (noising) process)**.** Fix a terminal time $T > 0$. The forward process $(\vec{X}_t)_{t\in[0,T]}$ starts at an anchor, $\vec{X}_0 = a \in \mathcal{A}$ with $a \sim p_{\mathrm{data}}$, and evolves as follows. While stuck at its anchor, $\vec{X}_t = a$, it waits and

$$\text{with hazard } \vec{\lambda}_t, \;\; \text{unsticks to a new state } Y \sim R_t(\cdot \,|\, a). \tag{1}$$

Here $R_t(\cdot \,|\, a)$ is a fixed *unsticking* kernel supported on $\mathsf{X}_{\mathcal{A}}$; when it is absolutely continuous we write $R_t(\mathrm{d}y \,|\, a) = r_t(y \,|\, a)\mathrm{d}y$. Once $\vec{X}_t \in \mathsf{X}_{\mathcal{A}}$, it diffuses according to:

$$\mathrm{d}\vec{X}_t = \vec{b}_t(\vec{X}_t)\mathrm{d}t + g_t \mathrm{d}\vec{W}_t. \tag{2}$$

The diffusion coefficient $g_t > 0$ is a scalar, $\vec{b}_t$ is a drift, $\mathrm{d}\vec{W}_t$ is a Wiener increment. There are no forward jumps into anchors, so once a trajectory unsticks, it never returns to $\mathcal{A}$.

In words: data start at anchors, wait at their anchor for a hazard-governed amount of time, then unstick into the continuous region and diffuse. Intuitively, the forward process transports mass from the discrete anchor set into the continuous ambient space. The third panel of Figure 1 shows an SJD trajectory.

The following construction is used in all of our experiments. It introduces two parameters: a blending matrix W, the axis our experiments vary, and a variance ratio η, which we keep at $\eta = 1$, the width of the hybrid perturbation kernel (Section 4.3).

Example 2.2 (VP-matched sequence SJD with non-local jump kernel)**.** In practice, $\vec{X}_0$ is typically a sequence of anchors $\vec{X}_0 = (a^{(1)}, \ldots, a^{(L)}) \in \mathcal{A}^L$ (e.g. a sentence). The unsticking kernel at a position may depend on the whole clean sequence, not on that position's anchor alone. We realize

this here through a blending matrix $W \in \mathbb{R}^{L\times L}$ that couples the positions. With VP coefficients $\alpha(t), \sigma(t)$ and an embedding $E : \mathcal{A} \to \mathbb{R}^d$, define the *blended mean* at position i,

$$\mu_i(\vec{X}_0) := \big(WE(\vec{X}_0)\big)_i = \sum_{j=1}^{L} W_{ij}\, E(a^{(j)}), \tag{3}$$

where $E(\vec{X}_0) \in \mathbb{R}^{L\times d}$ stacks the per-position embeddings. Position i is corrupted toward μ_i rather than its own embedding only. At $W = I$ we recover $\mu_i = E(a^{(i)})$ and the positions' jumps are independent. Position i then unsticks to

$$r_t\big(y \mid i, \vec{X}_0\big) = \mathcal{N}\Big(y; \alpha(t)\mu_i(\vec{X}_0), \eta^2\sigma^2(t)I\Big), \qquad \eta \in (0, 1]. \tag{4}$$

The single-particle kernel $r_t(\cdot \mid a)$ of Definition 2.1 has become $r_t(\cdot \mid i, \vec{X}_0)$, now depending on the whole initial sequence. It departs from the usual VP perturbation kernel in its mean, set by W, and the variance, set by $\eta \in (0, 1]$. Figure 4 shows the effect of η.

Remark 2.3 (Mass split). At each t, the law p_t decomposes as $p_t = p_t^{\mathrm{ac}} + p_t^{\mathrm{at}}$, where p_t^{ac} is an absolutely continuous density on $\mathsf{X}_{\mathcal{A}}$ and $p_t^{\mathrm{at}} = \sum_{a\in\mathcal{A}} m_t(a)\delta_a$ is an atomic density with anchor mass $m_t(a) \in [0, 1]$. The anchor mass $m_t(a)$ decays deterministically given $\vec{\lambda}_t$ when the diffusion does not re-attach to anchors in forward time.

Remark 2.4 (Conditional law is a mixture over unstick times). Conditioned on $\vec{X}_0 = a$, a particle that has already unstuck by time t did so at some random time $\tau \in (0, t)$, and its continuous density at t is the convolution of the unsticking kernel at τ with the diffusion semigroup from τ to t. Integrating over τ against the hazard-survival density gives

$$p_t^{\mathrm{ac}}(y \mid a) = \int_0^t \vec{\lambda}_\tau(a) S_a(\tau)\big(r_\tau(\cdot \mid a) * K_{\tau\to t}\big)(y)\mathrm{d}\tau,$$

where $S_a(\tau) = \exp\big(-\int_0^\tau \vec{\lambda}_s(a)\mathrm{d}s\big)$ is the anchor survival function and $K_{\tau\to t}$ is the forward diffusion semigroup. In general, $p_t^{\mathrm{ac}}(\cdot \mid a)$ is therefore a mixture and not a single Gaussian, even when both $r_\tau(\cdot \mid a)$ and $K_{\tau\to t}$ are. Two particles that started at the same anchor but unstuck at different times carry different accumulated variances, and $p_t^{\mathrm{ac}}(\cdot \mid a)$ averages over these.

Remark 2.5 (Terminal prior). In Example 2.2, when the VP coefficients $\alpha(T) \to 0$ and $\sigma(T) \to 1$ and every anchor loses its mass by T (i.e. $S_a(T) = 0$ for all a, so $p_T^{\mathrm{at}} = 0$), every mixture component of Remark 2.4 converges to $\mathcal{N}(0, I_d)$ *regardless of* τ, η, *or the blending matrix* W.

2.2 Reverse-time generative dynamics: sticking

Intuitively, if the forward process jumps off anchors, then the reverse-time process must jump into anchors. We state this formally below.

Theorem 2.6 (Time reversal for jump-diffusions)**.** *Under mild regularity assumptions, the reverse-time process* $(\overleftarrow{X}_\tau)_{\tau\in[0,T]}$ *with* $\tau = T - t$ *is Markov on* $\mathbb{R}^d$ *and admits:*

$$\mathrm{d}\overleftarrow{X}_\tau = \big(\overleftarrow{b}_t(\overleftarrow{X}_\tau)\big)\mathrm{d}\tau + g_t\mathrm{d}\overleftarrow{W}_\tau \quad \text{on } \mathsf{X}_{\mathcal{A}}, \tag{5}$$

$$\overleftarrow{b}_t(x) = -\vec{b}_t(x) + g_t^2\nabla\log p_t(x), \quad x \in \mathsf{X}_{\mathcal{A}}, \tag{6}$$

and a jump kernel that sticks to anchors, uniquely defined by the flux equation

$$p_t(\mathrm{d}y)\overleftarrow{J}_t(y, \mathrm{d}x) = p_t(\mathrm{d}x)\vec{J}_t(x, \mathrm{d}y). \tag{7}$$

In particular, for $y \in \mathsf{X}_{\mathcal{A}}$ *and* $x = a \in \mathcal{A}$*,*

$$\overleftarrow{J}_t(y, \{a\}) = \frac{p_t(\{a\})}{p_t(y)}\vec{\lambda}_t r_t(y \mid a). \tag{8}$$

3 Learning sticky dynamics

To turn Theorem 2.6 into a usable generative model we need to evaluate, at every (y, t), the reverse drift (6) and the sticky-jump rates (8). Both involve the SJD marginal p_t, which we cannot evaluate

in closed form: the off-anchor conditional $p_t^{\mathrm{ac}}(\cdot \mid a)$ is the mixture over unstick times of Remark 2.4, and the marginal $p_t(y) = \sum_{a \in \mathcal{A}} p_0(a) p_t^{\mathrm{ac}}(y \mid a)$ averages this mixture across anchors.

Our strategy is to learn the time-reversed SJD by training a single object: a denoising classifier $P_\theta(a \mid y, t)$ fit by cross-entropy on SJD-corrupted pairs. We will show that an analytic reweighting of P_θ recovers both the score and the reverse hazard exactly — so that *no separate hazard head is needed* — and that the whole training pipeline is simulation-free under standard conditions on the forward dynamics.

3.1 When and where to stick

By Theorem 2.6, the sticky jump law has per-anchor backward intensities

$$\Lambda_t^\star(a \mid y) \equiv \overleftarrow{J}_t(y, \{a\}) = \frac{m_t(a)}{p_t(y)} \vec{\lambda}_t r_t(y \mid a), \quad y \in \mathsf{X}_{\mathcal{A}},\ a \in \mathcal{A}. \tag{9}$$

Summing over anchors gives the hazard at which y commits, regardless of the destination,

$$\overleftarrow{\lambda}_t^\star(y) := \sum_{a \in \mathcal{A}} \Lambda_t^\star(a \mid y) = \frac{\vec{\lambda}_t \sum_{a \in \mathcal{A}} m_t(a) r_t(y \mid a)}{p_t(y)}, \tag{10}$$

and normalizing gives the allocation probability over destinations, conditional on a sticky jump,

$$\pi_t^\star(a \mid y) := \frac{\Lambda_t^\star(a \mid y)}{\sum_{b \in \mathcal{A}} \Lambda_t^\star(b \mid y)} = \frac{m_t(a) r_t(y \mid a)}{\sum_{b \in \mathcal{A}} m_t(b) r_t(y \mid b)}. \tag{11}$$

These are the reverse sampler's two decisions: *when* and *where* to commit, and together they fully factorize the sticky jump law:

$$\Lambda_t^\star(a \mid y) = \overleftarrow{\lambda}_t^\star(y) \pi_t^\star(a \mid y), \qquad \sum_{a \in \mathcal{A}} \pi_t^\star(a \mid y) = 1. \tag{12}$$

Crucially, $p_t(y)$ cancels from the allocation (11); so does the forward hazard whenever it is *anchor-agnostic*, i.e. $\vec{\lambda}_t(a) \equiv \vec{\lambda}_t$. The reverse hazard (10) thus carries all of the dependence on the unknown $p_t(y)$ that we still need to eliminate.

3.2 Denoising posterior and classifier training

Define the SJD denoising posterior

$$p_t(a \mid y) := \Pr\left(\vec{X}_0 = a \mid \vec{X}_t = y\right) = \frac{p_0(a) p_t^{\mathrm{ac}}(y \mid a)}{p_t(y)}, \quad a \in \mathcal{A},\ y \in \mathsf{X}_{\mathcal{A}}, \tag{13}$$

where $p_t^{\mathrm{ac}}(\cdot \mid a)$ is the mixture from Remark 2.4. With paired samples $(\vec{X}_0, \vec{X}_t)$ drawn $\vec{X}_0 \sim p_0$ and $\vec{X}_t \sim p_t^{\mathrm{ac}}(\cdot \mid \vec{X}_0)$, we train a classifier $P_\theta(a \mid y, t)$ by the standard cross-entropy

$$\mathcal{L}(\theta) := \mathbb{E}\left[-\log P_\theta\big(\vec{X}_0 \mid \vec{X}_t, t\big)\right], \tag{14}$$

whose Bayes-optimal predictor is exactly (13).

The classifier sees y and t but not the latent unstick time τ that produced $\vec{X}_t$ from $\vec{X}_0$ which is the same information pattern available to the reverse sampler at inference. This is by design: the time-reversal theorem operates on p_t, not on trajectories, and has already integrated out τ by flux balance.

3.3 Denoising hazard matching (DHM)

The reverse hazard $\overleftarrow{\lambda}_t^\star(y)$ (10) still contains the unknown marginal $p_t(y)$. We eliminate it by writing $\overleftarrow{\lambda}_t^\star$ as a conditional expectation of a forward-known quantity.

Definition 3.1 (DHM target)**.** For $\vec{X}_0 \in \mathcal{A}$ and $y \in \mathsf{X}_{\mathcal{A}}$, define

$$\widehat{\lambda}_t(y, \vec{X}_0) := \frac{\vec{\lambda}_t(\vec{X}_0) S_{\vec{X}_0}(t) r_t(y \mid \vec{X}_0)}{p_t^{\mathrm{ac}}(y \mid \vec{X}_0)}, \tag{15}$$

where $S_{\vec{X}_0}(t) = \exp\big(-\int_0^t \vec{\lambda}_s(\vec{X}_0)\mathrm{d}s\big)$ is the survival.

The target involves only forward-specified quantities: the forward hazard, its survival, the unsticking density at the current time, and the off-anchor conditional from Remark 2.4. It contains no intractable marginal $p_t(y)$.

Theorem 3.2 (Denoising hazard matching)**.** *Let $(\vec{X}_0, \vec{X}_t)$ be paired by the SJD forward law, $\vec{X}_0 \sim p_0$ and $\vec{X}_t \sim p_t^{\mathrm{ac}}(\cdot \mid \vec{X}_0)$. Then*

$$\mathbb{E}\left[\widehat{\lambda}_t(\vec{X}_t, \vec{X}_0) \;\middle|\; \vec{X}_t = y\right] = \overleftarrow{\lambda}_t^{\star}(y) \qquad \textit{for almost every } y \in \mathsf{X}_{\mathcal{A}}. \tag{16}$$

Theorem 3.2 is the hazard analogue of denoising score matching [Vincent, 2011]: the reverse hazard, like the score, is an intractable functional of the marginal that nonetheless equals a tractable conditional expectation under the same forward corruption used for training.

3.4 Plug-in: one classifier, no hazard head

Because $\vec{X}_0$ is discrete, (16) expands as a finite sum,

$$\overleftarrow{\lambda}_t^{\star}(y) = \sum_{a \in \mathcal{A}} \widehat{\lambda}_t(y, a) p_t(a \mid y), \tag{17}$$

which substitutes a known weighting against an unknown distribution. Replacing $p_t(\cdot \mid y)$ with the trained classifier P_θ yields the Rao–Blackwellized plug-ins

$$\Lambda_{\mathrm{plug},\theta}(a \mid y, t) := \widehat{\lambda}_t(y, a) P_\theta(a \mid y, t), \qquad \lambda_{\mathrm{plug},\theta}(y, t) := \sum_{a \in \mathcal{A}} \Lambda_{\mathrm{plug},\theta}(a \mid y, t). \tag{18}$$

Both ingredients of the factorization (12) are now in hand: the total commit rate is $\lambda_{\mathrm{plug},\theta}(y, t)$, and the anchor allocation at a jump is

$$\pi_{\mathrm{plug},\theta}(a \mid y, t) = \frac{\Lambda_{\mathrm{plug},\theta}(a \mid y, t)}{\lambda_{\mathrm{plug},\theta}(y, t)} \propto \widehat{\lambda}_t(y, a) P_\theta(a \mid y, t). \tag{19}$$

If $P_\theta(\cdot \mid y, t) = \mathrm{softmax}(\ell_\theta(\cdot; y, t))$, sampling the destination anchor reduces to a softmax over reweighted logits $\ell_\theta(a; y, t) + \log \widehat{\lambda}_t(y, a)$. Training is plain cross-entropy; all SJD-specific structure enters at inference through the analytic reweighting $\widehat{\lambda}_t$. Appendix C.3 describes three properties that justify (18).

3.5 Simulation-free training

Sampling $\vec{X}_t \sim p_t^{\mathrm{ac}}(\cdot \mid \vec{X}_0)$ is simulation-free under three conditions on the forward process.

(i) *Closed-form survival.* The survival $S_a(\tau)$ should be available in closed form so that an unstick time τ can be drawn by inverse CDF on $(0, t)$. For hazards of the form $\vec{\lambda}_\tau(a) = \beta(\tau) w(a)$ with β elementary (e.g. constant, linear, polynomial, piecewise-constant), S_a is a single exponential.

(ii) *Gaussian closure of the convolution.* The convolution $r_\tau(\cdot \mid a) * K_{\tau \to t}$ should be Gaussian in closed form. This holds whenever r_τ is Gaussian and the forward SDE is linear-Gaussian.

(iii) *One-sample unbiasedness.* A single draw $(\tau, \vec{X}_t)$ is an unbiased sample from $p_t^{\mathrm{ac}}(\cdot \mid \vec{X}_0)$. The cross-entropy loss therefore receives an unbiased gradient from one sample per step; we never need to evaluate the mixture density $p_t^{\mathrm{ac}}(y \mid a)$ during training.

Example 2.2, continued. Example 2.2 satisfies all three conditions. The VP-SDE is linear-Gaussian; for an anchor-agnostic hazard $\vec{\lambda}_t(\vec{X}_0) \equiv \vec{\lambda}(t)$, the survival $S(t) = \exp\big(-\int_0^t \vec{\lambda}(s)\mathrm{d}s\big)$ is a single

Algorithm 1 SJD simulation-free training.

Require: Prior p_0; $\vec{\lambda}_t$, S_a, r_τ, $K_{\tau\to t}$; embedding E and blending matrix W; P_θ; T.

while not converged **do**
  Sample $\vec{X}_0 \sim p_0$, $t \sim \mathrm{Unif}(0, T)$.
  Compute $\mu \leftarrow WE(\vec{X}_0) \in \mathbb{R}^{L\times d}$.
  Sample $u \sim \mathrm{Unif}(0, 1)$.
  if $u < S_{\vec{X}_0}(t)$ **then continue**
  Sample $\tau \sim \vec{\lambda}_\tau(\vec{X}_0) S_{\vec{X}_0}(\tau)/(1-S_{\vec{X}_0}(t))$.
  Sample $\vec{X}_t \sim (r_\tau * K_{\tau\to t})(\cdot \mid \vec{X}_0, i)$ at each unstuck position i, using (20).
  Step θ on $-\log P_\theta(\vec{X}_0 \mid \vec{X}_t, t)$.
end while

Algorithm 2 SJD predictor sampling.

Require: P_θ; $\vec{b}_t$, g_t, $\vec{\lambda}_t$, r_t; T, $\Delta\tau$.

$y \sim p_T$; $\tau \leftarrow 0$; mark all sites active.
while $\tau < T$ and any site is active **do**
  $t \leftarrow T - \tau$.
  One forward pass: $P_\theta(\cdot \mid y, t)$, $s_\theta(y, t)$.
  Evaluate $\widehat{\lambda}_t(y, a)$ for $a \in \mathcal{A}$ by (65).
  Form $\lambda_{\mathrm{plug},\theta}$, $\pi_{\mathrm{plug},\theta}$ by (18).
  Drift: $y \leftarrow y + (-\vec{b}_t(y) + g_t^2 s_\theta)\Delta\tau + g_t\sqrt{\Delta\tau}\xi$.
  Jump: per active site, with prob. $1 - e^{-\lambda_{\mathrm{plug},\theta}\Delta\tau}$, draw $a \sim \pi_{\mathrm{plug},\theta}$, freeze site at a.
  $\tau \leftarrow \tau + \Delta\tau$.
end while
return y.

exponential and τ is sampled by inverse CDF in closed form. The convolution evaluates to a single Gaussian whose mean is the blended mean and whose variance $v_t(\tau)$ (Appendix D.1):

$$(r_\tau * K_{\tau\to t})(\cdot \mid \vec{X}_0, i) = \mathcal{N}\Big(\cdot;\, \alpha(t)\mu_i(\vec{X}_0),\, v_t(\tau) I_d\Big), \quad v_t(\tau) = \sigma^2(t) - (1-\eta^2)\tfrac{\alpha^2(t)}{\alpha^2(\tau)}\sigma^2(\tau). \tag{20}$$

Sampling $\vec{X}_t$ given $(\vec{X}_0, \tau)$ is a single Gaussian draw at each position, with the per-position mean $\alpha(t)\mu_i(\vec{X}_0)$ obtained from one $L \times L$ matmul $\mu = WE(\vec{X}_0)$.

Algorithm 1 ties the pieces together. In practice, it is vectorized across a minibatch: the never-unstuck branch is implemented as a loss mask. Appendix D.4 and Algorithm 2 describe how to sample from a trained SJD. When $W \neq I$, W enters the sampler only through the reverse drift: at $\eta = 1$ the classifier-induced score evaluates the residual to the blended posterior mean at the cost of one blending product per reverse step, while the commit rates and allocation are unchanged (Appendix D.3).

Cost. Despite the added machinery, SJD trains and samples at the same cost as existing hybrids. Training is one classifier forward–backward pass per step on a cross-entropy loss, the same as a masked or hybrid baseline, because all SJD-specific structure enters through the jump corruption: closed-form draws of $(\tau, \vec{X}_t)$ and one $L \times L$ blending product per minibatch. At sampling, each reverse step is one forward pass through P_θ plus the analytic reweighting, an N_τ-point quadrature at $O(N_\tau|\mathcal{A}|d)$ that runs one to two orders of magnitude below the network pass (Appendix D.2).

4 Connections to existing diffusions

The SJD construction of Section 2 unifies several existing model classes by varying three forward-process levers: the unsticking kernel r_a, the forward hazard $\vec{\lambda}_t$, and, in the VP-matched family of Example 2.2, the blending matrix W and off-anchor variance ratio η. Different limits of these quantities recover absorbing-state discrete diffusion, pure continuous diffusion, and recently-proposed hybrid diffusion. We make these correspondences precise below.

4.1 Masked diffusion as a degenerate-kernel limit

Masked diffusion models (MDMs) [Austin et al., 2021, Sahoo et al., 2024, Shi et al., 2024] are the SJD whose off-anchor region collapses to a single absorbing point. Identify $\mathcal{A} = \{e(v) : v \in \mathcal{V}\}$ with the embedded vocabulary, fix a mask location $m \notin \mathcal{A}$, and set

$$r_a(\cdot, t) \equiv \delta_m, \qquad g_t \equiv 0, \qquad \vec{\lambda}_t(a) = \beta(t) := -\dot{\alpha}_t/\alpha_t. \tag{21}$$

Then $p_t^{\mathrm{ac}}(\cdot \mid a) = (1 - \alpha_t)\delta_m$, and the plug-in (19) at $y = m$ reduces to $\pi_{\mathrm{plug},\theta}(a \mid m, t) \propto P_\theta(a \mid m, t)$ with commit rate $\beta(t)\alpha_t/(1 - \alpha_t)$, the standard MDM step. SJD strictly contains

this limit and additionally provides an absolutely continuous off-anchor region whenever r_a has a Lebesgue density.

This limit makes the "information void" of Zheng et al. [2026] structural: because every anchor unsticks to the same point, the conditional law $p_t^{\mathrm{ac}}(\cdot|a) = (1-\alpha_t)\delta_m$ is anchor-independent, so the corrupted state at a masked position carries no evidence about its source token. Correspondingly, the spatial factor of the DHM target (15) reduces to the constant $1/(1-\alpha_t)$: the commit rate is time-only, and the allocation can be informed only from context, through the classifier (Section 4.5).

4.2 Continuous diffusion as the no-jump limit

Continuous diffusion is the limit in which every anchor releases its mass at $t = 0^+$, that is $S_a(0^+) = 0$ for all $a \in \mathcal{A}$, after which $\vec{\lambda}_t \equiv 0$. For $t > 0$ the marginal p_t then has no atomic part, (2) is a standard SDE, and Theorem 2.6 reduces to the classical score-based time-reversal of Song et al. [2021]. Because $p_t(\{a\}) = 0$ for every anchor, the flux identity (7) assigns zero reverse intensity to every $y \in \mathsf{X}_{\mathcal{A}}$: the absence of atoms in the forward marginal is precisely what removes the reverse jumps. For discrete data, three external mechanisms appear in the literature. *(i) Decoder:* train a continuous diffusion in latent space and push samples through a separately-trained decoder [Li et al., 2022, Dieleman et al., 2022, Lovelace et al., 2023]. *(ii) Quantization:* Bit Diffusion [Chen et al., 2023] thresholds the terminal state. *(iii) Doob's h-transform:* Liu et al. [2023] bias the SDE by the gradient of a singular log-density so its terminal law lands on a discrete set. A recent wave of embedding- and one-hot-space models closes the gap to discrete diffusion [Yang et al., 2026, Hu et al., 2026, Chen et al., 2026, Lee et al., 2026]; in all of them, discreteness is still restored by a step outside the dynamics, whether a decoder, a readout, or an argmax. SJD instead enforces discreteness through the dynamics: anchors are absorbing in reverse time as a direct consequence of (7).

4.3 Hybrid diffusion

Hybrid diffusion models such as CADD [Zheng et al., 2026], CANDI [Pynadath et al., 2026], and CCDD [Zhou et al., 2026] couple a masked chain with a continuous latent that carries graded information at masked positions. The VP-matched SJD of Example 2.2 at $(W, \eta) = (I, 1)$ recovers this construction exactly. With $W = I$, the unsticking kernel reduces to the per-anchor form $r_\tau(\cdot \mid a)$, with $\eta = 1$ it coincides with the VP perturbation kernel ($r_\tau = q_\tau$), and the semigroup property collapses Remark 2.4 to

$$p_t^{\mathrm{ac}}(y \mid a) = (1 - S(t))\mathcal{N}\big(y; \alpha(t)a, \sigma^2(t)I\big). \tag{22}$$

Proposition 4.1 (The plain hybrid coincides with CADD)**.** *Consider the VP-matched SJD of Example 2.2 at $(W, \eta) = (I, 1)$ with anchor-agnostic hazard $\vec{\lambda}(t)$ and survival $S(t)$, and CADD with masking survival α_t, continuous kernel $\mathcal{N}(\sqrt{\bar{\gamma}_t}E(x_0), (1-\bar{\gamma}_t)I_d)$ at masked positions, and a common embedding map E. Identify $S(t) \leftrightarrow \alpha_t$ and $(\alpha(t), \sigma^2(t)) \leftrightarrow (\sqrt{\bar{\gamma}_t}, 1-\bar{\gamma}_t)$. Then:*

- *(i) the time-t laws of the corrupted state given the data coincide for every t, and the SJD cross-entropy objective of Algorithm 1 equals CADD's cross-entropy objective;*
- *(ii) the reverse process of Theorem 2.6 reduces to a commit rate $\vec{\lambda}(t)S(t)/(1-S(t))$ shared across anchors, an allocation equal to the classifier posterior, and a reverse SDE whose score is the residual to the posterior-mean embedding;*
- *(iii) CADD's sampler is a first-order discretization of (ii), so the two generative laws coincide in the small-step limit.*

The sampler that hybrids assemble component by component thus arise jointly as the time reversal of one forward process, with rate, destination, and drift fixed by flux balance (7). This turns the unsticking kernel into a new design space: the continuous coordinate is reversed state rather than conditioning information, so any forward kernel yields exact reverse dynamics through Theorem 2.6.

4.4 Sticky boundaries: a classical antecedent

SJD's theoretical antecedent is the classical theory of sticky-boundary diffusions [Feller, 1952, 1954, Itô and McKean, 1963]. A one-dimensional sticky diffusion holds at the boundary for an exponential time and re-enters the interior according to a fixed jump-out measure; its law splits as an interior

density plus a boundary atom [Bou-Rabee and Holmes-Cerfon, 2020]. SJD lifts this model to a multi-anchor, high-dimensional setting: $\mathcal{A}$ replaces the boundary point, $\vec{\lambda}_t$ replaces the sticky coefficient, and R_a replaces the jump-out measure, with Remark 2.3 as the corresponding mass split.

4.5 Commitment schedules and token ordering

The DHM target (15) factors into temporal and spatial components:

$$\widehat{\lambda}_t(y, \vec{X}_0) = \underbrace{\vec{\lambda}_t(\vec{X}_0) S_{\vec{X}_0}(t)}_{\text{temporal}} \cdot \underbrace{\frac{r_t(y \mid \vec{X}_0)}{p_t^{\text{ac}}(y \mid \vec{X}_0)}}_{\text{spatial}}.$$

At $(W, \eta) = (I, 1)$, (22) forces the spatial factor to $1/(1 - S(t))$ and the allocation (19) reduces to the classifier posterior, recovering the time-only schedule of MDM-style samplers. Choosing $\eta < 1$ introduces a per-anchor mixture over unstick times whose density depends on proximity $\|y - \alpha(t)E(a^{(i)})\|^2$; $W \neq I$ then replaces the per-anchor distance with a cross-position distance to the blended mean $\mu_i(\vec{X}_0)$, so the hazard ratio at one position depends on the configuration of its neighbors. This gives a process-level reading of adaptive ordering [Kim et al., 2025, Chao et al., 2025]: at $(I, 1)$ the state carries no timing signal, so a planner can only rank positions by classifier confidence, an ordering imposed at sampling time. State-dependent forward hazards $\vec{\lambda}_t(a)$, in the spirit of MD4's GenMD4 [Shi et al., 2024], are a further axis (See Appendix C.5). We fix $\eta = 1$ in all experiments and recommend it as the practical default: the kernel then coincides with the perturbation kernel, so the identity-blend member is exactly CADD (Proposition 4.1), and the commit rate and allocation are closed-form and identical for every W (Appendix D.3).

5 Experiments

The cross-position structure in Example 2.2 and Section 4.3 should be exploitable whenever positions are statistically dependent under a known incidence structure, so that a fixed linear functional of a position's neighbors is itself diagnostic of that position's value. Such structure is common in real-world data, and we instantiate three examples: spatial locality on CIFAR-10 through a value-space blur, local sequential correlation on Text8, and the row/column/box constraint graph on Sudoku.

5.1 Image generation

Figure 2 sweeps $\sigma_B \in \{1.5, 2.0, 2.5\}$, the spatial correlation length of the corruption, against the $\sigma_B = 0$ endpoint across sampling budgets. The identity kernel, the point at which existing hybrids sit, is the weakest member of the sweep: the corruption geometry is a substantive degree of freedom on images as well. Full settings are in Appendix E.1. At the full budget of 256 NFE, Table 2 compares the best bandwidth from the sweep, σ_B=2.5, with masked, continuous, and hybrid baselines under matched settings: SJD attains the best FID overall, 1.3 points below the identity-kernel hybrid, with DDPM the closest external baseline.

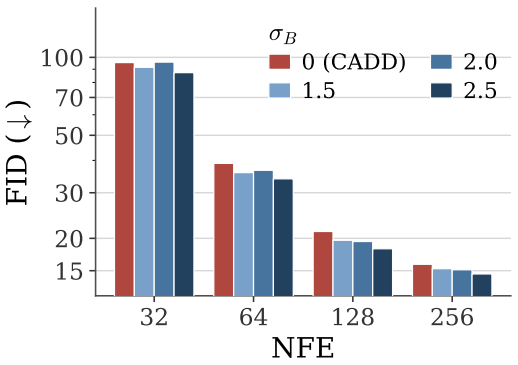

Figure 2: CIFAR-10 FID (↓) across sampling budgets as a function of the value-space blur bandwidth σ_B. σ_B=0 is the plain hybrid, which coincides with CADD [Zheng et al., 2026] (Proposition 4.1); $\sigma_B \in \{1.5, 2.0, 2.5\}$ are SJD.

5.2 Text generation

Setting W to a 1D Gaussian convolution of bandwidth σ_W replaces a token with a noisy blend of a local neighborhood. This is a natural inductive bias for characters, where adjacent characters are correlated, so a blended window ties each position's corruption to its local context. Figure 3 reports valid-word counts at length thresholds ≥ 5 and ≥ 6 for NFE budgets of 32, 64, and 128. For $\sigma_W \geq 1.0$ SJD surpasses both CANDI [Pynadath

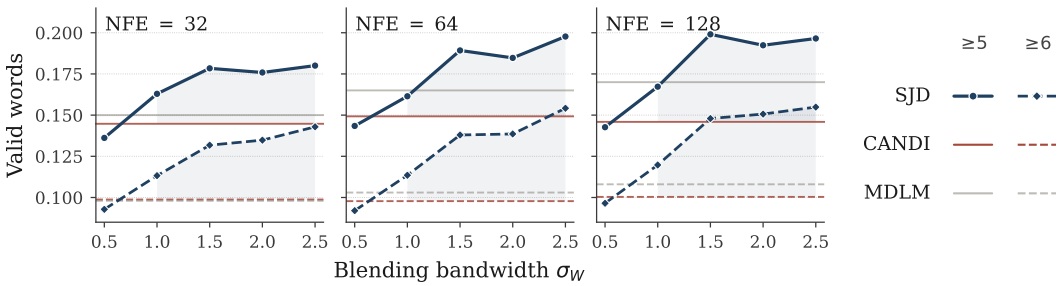

Figure 3: Text8 valid-word counts at length thresholds ≥ 5 and ≥ 6, as a function of the blending-kernel bandwidth σ_W, across NFE budgets. CANDI corresponds to $\sigma_W = 0$ ($W = I$); MDLM is the degenerate-kernel limit.

Table 1: Sudoku board accuracy and convergence. *Acc.* is exact-match board accuracy. *Takeoff* is the average training step at which seeds first begin to solve complete boards.

Method	Acc. (%) ↑	Takeoff ↓
Autoregressive [Shah et al., 2024]		
Left-to-right	9.73	NA
Solver order	87.18	NA
Masked Diffusion [Kim et al., 2025]		
MDM	89.49	NA
Hybrid Diffusion		
CADD [(W,η)=$(I,1)$]	47.12	203k
SJD (σ_W=1.5)	**95.65**	**50k**

Table 2: FID on CIFAR-10 32×32 for unconditional generation; all methods sampled at 256 NFE.

Method	FID ↓
Hybrid Diffusion	
CADD [(W,η)=$(I,1)$]	15.88
SJD (σ_B=2.5)	**14.57**
Masked Diffusion	
MD4 [Shi et al., 2024]	17.43
MDLM [Sahoo et al., 2024]	18.11
Continuous Diffusion	
Bit Diff. [Chen et al., 2023]	19.82
DDPM [Ho et al., 2020]	<u>14.66</u>

et al., 2026] and MDLM [Sahoo et al., 2024], with a larger margin at the ≥ 6 threshold than at ≥ 5 and gains that hold at NFE $= 128$. Figure 5 further emphasizes this finding with quality–diversity frontiers on Text8, by NFE. Full settings are in Appendix E.2. For length ≥ 5, SJD at 32 NFE exceeds both terminal-rounding baselines at 128 NFE (Appendix E.2.1); a likelihood-bound comparison with MD4 appears in Appendix E.2.2.

5.3 Solving sudokus

Sudoku probes the blending axis W when the governing structure is a hard combinatorial constraint graph. The task is to complete a 9×9 grid so that every row, column, and 3×3 box is a permutation of $\{1, \ldots, 9\}$, which requires the model to honor board-level constraints rather than local statistics. We encode those constraints directly into the unsticking kernel. The full construction is given in Appendix E.3. We then ask whether this constraint-aware corruption helps the model learn the board's logic. Table 1 isolates three effects of W. Across 6 seeds CADD is bimodal: several runs collapse to near-chance performance. SJD never collapses. Moreover, by board takeoff (the training step at which a seed first resolves complete boards) SJD takes off roughly $4\times$ sooner; even the CADD seeds that avoid collapse begin solving boards far later. Finally, SJD's mean board accuracy exceeds CADD's and the autoregressive solver granted the oracle solving order.

6 Conclusion

sticky jump diffusions unify masked, continuous, and hybrid diffusion through a single forward process. Flux balance determines the reverse commit rates and destinations, which denoising hazard matching estimates from one classifier trained by simulation-free cross-entropy. Cross-position unsticking kernels encode spatial, sequential, or constraint dependencies and improve generation over the identity-kernel hybrid. Our experiments use VP-matched kernels at on CIFAR-10, Text8, and Sudoku; evaluation at language-model scale and with other kernel families remains open.

Acknowledgments and Disclosure of Funding

This work was supported by the National Science Foundation under grant IIS-2508145 and by an Amazon Research Award.

References

Brian D. O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12, 1982.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, 2021.

Joe Benton, Yuyang Shi, Valentin De Bortoli, George Deligiannidis, and Arnaud Doucet. From denoising diffusions to denoising Markov models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 86, 2024.

Nawaf Bou-Rabee and Miranda C. Holmes-Cerfon. Sticky Brownian motion and its numerical solution. *SIAM Review*, 62, 2020.

Andrew Campbell, William Harvey, Christian Weilbach, Valentin De Bortoli, Tom Rainforth, and Arnaud Doucet. Trans-dimensional generative modeling via jump diffusion models. In *Advances in Neural Information Processing Systems*, 2023.

Chen-Hao Chao, Wei-Fang Sun, Hanwen Liang, Chun-Yi Lee, and Rahul G. Krishnan. Beyond masked and unmasked: Discrete diffusion models via partial masking. In *Advances in Neural Information Processing Systems*, 2025.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. In *International Conference on Learning Representations*, 2023.

Yuxin Chen, Chumeng Liang, Hangke Sui, Ruihan Guo, Chaoran Cheng, Jiaxuan You, and Ge Liu. LangFlow: Continuous diffusion rivals discrete in language modeling. *arXiv preprint arXiv:2604.11748*, 2026.

Giovanni Conforti and Christian Léonard. Time reversal of Markov processes with jumps under a finite entropy condition. *Stochastic Processes and their Applications*, 2022.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems*, 2021.

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H. Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, et al. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022.

Floor Eijkelboom, Grigory Bartosh, Christian A. Naesseth, Max Welling, and Jan-Willem van de Meent. Variational flow matching for graph generation. In *Advances in Neural Information Processing Systems*, 2024.

William Feller. The parabolic differential equations and the associated semi-groups of transformations. *Annals of Mathematics*, 55, 1952.

William Feller. Diffusion processes in one dimension. *Transactions of the American Mathematical Society*, 77, 1954.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.

Peter Holderrieth, Marton Havasi, Jason Yim, Neta Shaul, Itai Gat, Tommi Jaakkola, Brian Karrer, Ricky T. Q. Chen, and Yaron Lipman. Generator matching: Generative modeling with arbitrary Markov processes. In *International Conference on Learning Representations*, 2025.

Keya Hu, Linlu Qiu, Yiyang Lu, Hanhong Zhao, Tianhong Li, Yoon Kim, Jacob Andreas, and Kaiming He. ELF: Embedded language flows. *arXiv preprint arXiv:2605.10938*, 2026.

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6, 2005.

Kiyosi Itô and Henry P. McKean, Jr. Brownian motions on a half line. *Illinois Journal of Mathematics*, 7, 1963.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, 2022.

Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. In *International Conference on Machine Learning*, 2025.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In *Advances in Neural Information Processing Systems*, 2021.

Chanhyuk Lee, Jaehoon Yoo, Manan Agarwal, Sheel Shah, Jerry Huang, Aditi Raghunathan, Seunghoon Hong, Nicholas M. Boffi, and Jinwoo Kim. Flow map language models: One-step language modeling via continuous denoising. *arXiv preprint arXiv:2602.16813*, 2026.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-LM improves controllable text generation. In *Advances in Neural Information Processing Systems*, 2022.

Xingchao Liu, Lemeng Wu, Mao Ye, and Qiang Liu. Learning diffusion bridges on constrained domains. In *International Conference on Learning Representations*, 2023.

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

Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q. Weinberger. Latent diffusion for language generation. In *Advances in Neural Information Processing Systems*, 2023.

Edward Nelson. *Dynamical Theories of Brownian Motion*. Princeton University Press, 1967.

Patrick Pynadath, Jiaxin Shi, and Ruqi Zhang. CANDI: Hybrid discrete-continuous diffusion models. In *International Conference on Learning Representations*, 2026.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander M. Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*, 2024.

Kulin Shah, Nishanth Dikkala, Xin Wang, and Rina Panigrahy. Causal language modeling can elicit search and reasoning capabilities on logic puzzles. In *Advances in Neural Information Processing Systems*, 2024.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data. In *Advances in Neural Information Processing Systems*, 2024.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, 2015.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23, 2011.

Zhihan Yang, Wei Guo, Shuibai Zhang, Subham Sekhar Sahoo, Yongxin Chen, Arash Vahdat, Morteza Mardani, and John Thickstun. Continuous diffusion scales competitively with discrete diffusion for language. *arXiv preprint arXiv:2605.18530*, 2026.

Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Beyond autoregression: Discrete diffusion for complex reasoning and planning. In *International Conference on Learning Representations*, 2025.

Huangjie Zheng, Shansan Gong, Ruixiang Zhang, Tianrong Chen, Jiatao Gu, Mingyuan Zhou, Navdeep Jaitly, and Yizhe Zhang. Continuously augmented discrete diffusion model for categorical generative modeling. In *International Conference on Learning Representations*, 2026.

Cai Zhou, Chenxiao Yang, Yi Hu, Chenyu Wang, Chubin Zhang, Muhan Zhang, Lester Mackey, Tommi Jaakkola, Stephen Bates, and Dinghuai Zhang. Coevolutionary continuous discrete diffusion: Make your diffusion language model a latent reasoner. In *International Conference on Machine Learning*, 2026.

A Notation, assumptions, and vocabulary

We use overhead arrows to disambiguate time direction:

- $\vec{\cdot}$ denotes *forward* (noising) objects, e.g., $\vec{X}_t$ (state), $\vec{b}_t$ (drift), $\vec{J}_t$ (jump kernel), and $\vec{\lambda}_t$ (forward hazard of leaving an anchor).
- $\overleftarrow{\cdot}$ denotes *reverse-time* (denoising) objects, e.g., $\overleftarrow{X}_\tau$, $\overleftarrow{b}_t$, $\overleftarrow{J}_t$, with reverse time $\tau = T - t$.

The state space is $\mathsf{X} = \mathbb{R}^d$ and the (finite or countable) set of anchors is $\mathcal{A} = \{a_k\}_{k=1}^K \subset \mathbb{R}^d$. We write the continuous region as

$$\mathsf{X}_{\mathcal{A}} := \mathbb{R}^d \setminus \mathcal{A}.$$

The forward process $(\vec{X}_t)_{t\in[0,T]}$ is an Itô diffusion on $\mathsf{X}_{\mathcal{A}}$ with scalar diffusion coefficient $g_t > 0$ and drift $\vec{b}_t$, and it can jump off anchors:

$$\mathrm{d}\vec{X}_t = \vec{b}_t(\vec{X}_t)\mathrm{d}t + g_t\mathrm{d}\vec{W}_t \quad \text{if } \vec{X}_t \in \mathsf{X}_{\mathcal{A}},$$

and, if $\vec{X}_t = a \in \mathcal{A}$, then with hazard $\vec{\lambda}_t$ it unsticks to $Y \sim R_a(\cdot)$ where R_a is a fixed probability kernel supported on $\mathsf{X}_{\mathcal{A}}$. There are no forward jumps into anchors: if $\vec{X}_t \in \mathsf{X}_{\mathcal{A}}$ the jump intensity into $\mathcal{A}$ is zero.

The reverse-time process $(\overleftarrow{X}_\tau)_{\tau\in[0,T]}$ with $\tau = T - t$ is Markov on $\mathbb{R}^d$ and (under Assumption C.1) satisfies on $\mathsf{X}_{\mathcal{A}}$:

$$\mathrm{d}\overleftarrow{X}_\tau = \overleftarrow{b}_t(\overleftarrow{X}_\tau)\mathrm{d}\tau + g_t\mathrm{d}\overleftarrow{W}_\tau, \qquad \overleftarrow{b}_t(x) = \vec{b}_t(x) - g_t^2\nabla\log p_t(x).$$

We write $R_a(\mathrm{d}y)$ for the forward unsticking kernel (a probability measure on $\mathsf{X}_{\mathcal{A}}$). Whenever R_a admits a density w.r.t. Lebesgue on $\mathsf{X}_{\mathcal{A}}$, we write $R_a(\mathrm{d}y) = r_a(y)\mathrm{d}y$. Likewise, whenever p_t admits a density on $\mathsf{X}_{\mathcal{A}}$ we write $p_t(\mathrm{d}y) = p_t(y)\mathrm{d}y$. All pointwise formulas involving $p_t(y)$ and $r_a(y,t)$ are understood under these absolute-continuity assumptions.

For sequence tasks, $E : \mathcal{A} \to \mathbb{R}^d$ denotes the embedding map and $W \in \mathbb{R}^{L\times L}$ a blending matrix on positions. The blended mean at position i is $\mu_i(\vec{X}_0) := \big(WE(\vec{X}_0)\big)_i$; we recover the per-anchor case $\mu_i = E(a^{(i)})$ at $W = I$.

B Preliminaries: existing diffusion frameworks

We review the model classes that Section 4 identifies with limits of the SJD construction of Section 2.

B.1 Continuous score-based diffusion

A score-based diffusion model on $\mathbb{R}^d$ [Sohl-Dickstein et al., 2015, Ho et al., 2020, Song et al., 2021] runs the Itô SDE $\mathrm{d}\vec{X}_t = \vec{b}_t(\vec{X}_t)\mathrm{d}t + g_t\mathrm{d}\vec{W}_t$ from $\vec{X}_0 \sim p_{\mathrm{data}}$ to a tractable prior p_T. We focus on the variance-preserving (VP) instance with $\vec{b}_t(x) = -\frac{1}{2}\beta(t)x$ and $g_t = \sqrt{\beta(t)}$, whose conditional marginal kernel is Gaussian,

$$q_t(\cdot \mid x_0) = \mathcal{N}\big(\cdot; \alpha(t)x_0, \sigma^2(t)I_d\big), \qquad \alpha(t) = \exp\Big(-\tfrac{1}{2}\int_0^t \beta(s)\mathrm{d}s\Big), \;\; \sigma^2(t) = 1 - \alpha^2(t),$$

and converges to $\mathcal{N}(0, I_d)$ as $t \to T$. The reverse-time process is the Itô SDE $\mathrm{d}\overleftarrow{X}_\tau = (-\vec{b}_t(\overleftarrow{X}_\tau) + g_t^2\nabla\log p_t(\overleftarrow{X}_\tau))\mathrm{d}\tau + g_t\mathrm{d}\overleftarrow{W}_\tau$ with $\tau = T - t$ [Nelson, 1967, Anderson, 1982, Song et al., 2021]. The score $\nabla\log p_t$ is learned by denoising score matching [Hyvärinen, 2005, Vincent, 2011]: a network $s_\theta(y,t)$ minimizes $\mathbb{E}[\|s_\theta(y,t) - \nabla_y \log q_t(y \mid x_0)\|^2]$, the conditional score being available in closed form for VP. For discrete data, this construction requires an external map to the support like a separately-trained decoder [Li et al., 2022, Dieleman et al., 2022, Lovelace et al., 2023], a quantization step [Chen et al., 2023], or a Doob h-transform [Liu et al., 2023].

B.2 Masked diffusion

Absorbing-state masked diffusion [Austin et al., 2021, Sahoo et al., 2024, Shi et al., 2024] operates on a discrete state space $\mathcal{V} \cup \{\mathsf{m}\}$ with a special mask symbol m. The forward process keeps each clean token with probability α_t and replaces it by m otherwise, where α_t is a smooth survival schedule with $\alpha_0 = 1$ and $\alpha_T = 0$; in continuous time the clean-to-mask hazard is $\beta(t) = -\dot{\alpha}_t/\alpha_t$. Time reversal yields a Markov chain in which masked positions un-mask at total rate $\beta(t)\alpha_t/(1-\alpha_t)$ to a token drawn from the denoising posterior $p_t(v \mid \mathsf{m}, x_t) \propto p_0(v)\Pr(\mathsf{m} \mid v, t)$, while clean positions remain absorbing. A classifier $P_\theta(v \mid x_t, t)$ is trained by cross-entropy on masked positions and substitutes for the posterior at sampling time. The per-position factorization is exact in the forward law and couples positions only through the classifier's context input; once a position commits, the only signal carried about its proximity to alternative tokens is whatever the classifier reconstructs from the surviving context.

B.3 Hybrid discrete-continuous diffusion

Hybrid models [Zheng et al., 2026, Pynadath et al., 2026, Zhou et al., 2026] couple a masked chain with a continuous latent that carries graded information at masked positions. CADD [Zheng et al., 2026] represents the state at time t as a pair $(x_t, z_t) \in (\mathcal{V} \cup \{\mathsf{m}\}) \times \mathbb{R}^d$. The discrete component x_t evolves as the masked chain of Appendix B.2 with survival α_t, and the continuous component, defined for masked sites, follows the VP perturbation kernel of an embedding,

$$z_t \mid x_0 \sim \mathcal{N}\big(\sqrt{\bar{\gamma}_t}E(x_0), (1-\bar{\gamma}_t)I_d\big), \qquad \text{when the site is masked at time } t,$$

with $\bar{\gamma}_t$ a VP-style schedule and E a fixed embedding map. Time reversal couples the discrete un-masking transition with a score-driven update on z_τ; both are predicted from a single network conditioned on (x_t, z_t, t). CANDI [Pynadath et al., 2026] keeps the masked-position augmentation and varies the reverse parameterization. CCDD [Zhou et al., 2026] instead corrupts the continuous component at every position under a factored forward law $q_t(x_t, z_t \mid x_0) = q_t^{\mathrm{disc}}(x_t \mid x_0) q_t^{\mathrm{cont}}(z_t \mid z_0)$, with possibly asynchronous schedules, so its forward lies outside the $(I, 1)$ configuration.

B.4 Framework-level unifications

Generator Matching [Holderrieth et al., 2025] learns the marginal generator of an arbitrary Markov process from conditional generators, and denoising Markov models [Benton et al., 2024] extend score matching to general state spaces; both are abstract recipes that must be instantiated per process class. SJD is such an instantiation, worked out end to end for jump-diffusions whose marginals mix a continuous density with anchor atoms: the flux identity (7) gives the reverse jump kernel in closed form, DHM (Theorem 3.2) turns the reverse hazard into a conditional expectation that a classifier estimates, training is simulation-free, and the unsticking kernel exposes the (W, η) design space our experiments study. Neither framework supplies these objects for the sticky, atom-carrying marginal.

C Theoretical details and proofs

C.1 Proof of Theorem 2.6

Assumption C.1 (Regularity)**.**

(i) $\vec{b}_t$ and g_t are continuous in t, locally Lipschitz in x, with $g_t \in (0, \infty)$.

(ii) For $x \in \mathcal{A}$, $\vec{J}_t(x, \mathrm{d}y) = \vec{\lambda}_t(x) R_t(\mathrm{d}y \mid x)$ on $\mathsf{X}_{\mathcal{A}}$; for $x \in \mathsf{X}_{\mathcal{A}}$, $\vec{J}_t(x, \cdot) \equiv 0$.

(iii) For almost every $t \in (0, T)$, p_t admits a density on $\mathsf{X}_{\mathcal{A}}$ and atoms on $\mathcal{A}$.

Theorem (Time reversal for jump–diffusions, restated)**.** *Under Assumption C.1, define the time-reversed process $\overleftarrow{X}_\tau := \vec{X}_{T-\tau}$, $\tau \in [0, T]$, and denote the corresponding forward time by $t := T - \tau$. Then $(\overleftarrow{X}_\tau)_{\tau \in [0,T]}$ is Markov.*

(Diffusion part on $\mathsf{X}_{\mathcal{A}}$). *On the continuous region $\mathsf{X}_{\mathcal{A}} = \mathbb{R}^d \setminus \mathcal{A}$, the reversed process admits an SDE*

$$\mathrm{d}\overleftarrow{X}_\tau = \tilde{b}_t(\overleftarrow{X}_\tau)\mathrm{d}\tau + g_t \mathrm{d}\overleftarrow{W}_\tau, \qquad \tilde{b}_t(x) = -\vec{b}_t(x) + g_t^2 \nabla \log p_t(x), \quad x \in \mathsf{X}_{\mathcal{A}}.$$

(Jump part: flux identity). *The reverse jump kernel $\overleftarrow{J}_t$ is uniquely determined (for a.e. t) by the flux equation*

$$p_t(\mathrm{d}y)\overleftarrow{J}_t(y,\mathrm{d}x) = p_t(\mathrm{d}x)\vec{J}_t(x,\mathrm{d}y). \tag{23}$$

In particular, since $\vec{J}_t(a,\mathrm{d}y) = \vec{\lambda}_t(a)R_t(\mathrm{d}y\,|\,a)$ for $a \in \mathcal{A}$ and $\vec{J}_t(x,\cdot) \equiv 0$ for $x \in \mathsf{X}_{\mathcal{A}}$, we have for every anchor $a \in \mathcal{A}$ the measure identity on $\mathsf{X}_{\mathcal{A}}$:

$$p_t(\mathrm{d}y)\overleftarrow{J}_t(y,\{a\}) = p_t(\{a\})\vec{\lambda}_t(a)R_t(\mathrm{d}y\,|\,a). \tag{24}$$

If moreover $p_t(\mathrm{d}y) = p_t(y)\mathrm{d}y$ on $\mathsf{X}_{\mathcal{A}}$ and $R_t(\mathrm{d}y\,|\,a) = r_t(y\,|\,a)\mathrm{d}y$, then for $y \in \mathsf{X}_{\mathcal{A}}$,

$$\overleftarrow{J}_t(y,\{a\}) = \frac{p_t(\{a\})}{p_t(y)}\vec{\lambda}_t(a)r_t(y\,|\,a).$$

Moreover $\overleftarrow{J}_t(a,\mathsf{X}_{\mathcal{A}}) = 0$, i.e. anchors are absorbing in reverse time.

Proof. Fix $t \in (0,T)$ for which the marginal p_t admits the decomposition

$$p_t = p_t^{\mathrm{ac}} + p_t^{\mathrm{at}}, \qquad p_t^{\mathrm{at}} = \sum_{a\in\mathcal{A}} m_t(a)\delta_a, \qquad p_t^{\mathrm{ac}}(\mathrm{d}x) = p_t(x)\mathrm{d}x \text{ on } \mathsf{X}_{\mathcal{A}},$$

as in Assumption C.1(iii), and let $\mathcal{U} = C_c^2(\mathbb{R}^d)$.

The proof extracts the backward generator $\overleftarrow{L}_t$ from a single integration-by-parts identity, taken from Conforti and Léonard [2022], by isolating the two pieces of structural information that identity carries. We first manipulate the jump contribution to recover the flux identity (23); specializing to the anchored forward kernel then yields the explicit form of $\overleftarrow{J}_t$ and the absorbing property at anchors. The diffusion contribution determines the drift $\tilde{b}_t$ via distributional integration by parts; the boundary terms generated by the atomic part of p_t cancel against the backward sticky jumps, with the cancellation supplied by the flux identity itself.

The integration-by-parts identity. The forward generator on $u \in \mathcal{U}$ is

$$\vec{L}_t u(x) = \mathbb{1}_{\mathsf{X}_{\mathcal{A}}}(x)\Big[\vec{b}_t(x)\cdot\nabla u(x) + \tfrac{1}{2}g_t^2\Delta u(x)\Big] + \int_{\mathbb{R}^d}\big(u(y)-u(x)\big)\vec{J}_t(x,\mathrm{d}y), \tag{25}$$

with $\vec{J}_t(x,\cdot) \equiv 0$ on $\mathsf{X}_{\mathcal{A}}$ and $\vec{J}_t(a,\mathrm{d}y) = \vec{\lambda}_t(a)R_t(\mathrm{d}y\,|\,a)$ on $\mathcal{A}$; the corresponding carré du champ is

$$\Gamma_t(u,v)(x) = \mathbb{1}_{\mathsf{X}_{\mathcal{A}}}(x)g_t^2\nabla u(x)\cdot\nabla v(x) + \int_{\mathbb{R}^d}\big(u(y)-u(x)\big)\big(v(y)-v(x)\big)\vec{J}_t(x,\mathrm{d}y).$$

Under Assumption C.1, $\mathcal{U}$ is a measure-determining algebra contained in $\mathrm{dom}\vec{L}_t$, and Conforti and Léonard [2022, Theorem 4.4] ensures the existence of a backward generator $\overleftarrow{L}_t$ (defined p_t-a.e.) such that, for all $u,v \in \mathcal{U}$ and a.e. t,

$$\int_{\mathbb{R}^d}\Big[\big(\vec{L}_t u + \overleftarrow{L}_t u\big)(x)v(x) + \Gamma_t(u,v)(x)\Big]p_t(\mathrm{d}x) = 0. \tag{26}$$

Equation (26) is the only analytic input to the proof: the flux identity, the anchored jump formula, and the score-form drift will all be consequences of choosing test functions u, v appropriately. Conservation of the continuous-martingale quadratic variation under time reversal forces the reversed diffusion coefficient to coincide with g_t, so $\overleftarrow{L}_t$ takes the form

$$\overleftarrow{L}_t u(x) = \mathbb{1}_{\mathsf{X}_{\mathcal{A}}}(x)\Big[\tilde{b}_t(x)\cdot\nabla u(x) + \tfrac{1}{2}g_t^2\Delta u(x)\Big] + \int_{\mathbb{R}^d}\big(u(y)-u(x)\big)\overleftarrow{J}_t(x,\mathrm{d}y), \tag{27}$$

for an unknown drift $\tilde{b}_t$ and an unknown backward jump kernel $\overleftarrow{J}_t$, both of which we now identify.

Step 1: flux identity. We first isolate the jump contribution to (26). The diffusion parts of $\vec{L}_t$, $\overleftarrow{L}_t$, and Γ_t satisfy a closed integration-by-parts identity on their own (the standard score-based time-reversal calculation in the absence of jumps; cf. Conforti and Léonard [2022, Theorem 5.7]) so subtracting it from (26) leaves the jump-only identity

$$\int_{\mathbb{R}^d} \Big[\big(\vec{L}_t^{\,\mathrm{jump}} u + \overleftarrow{L}_t^{\,\mathrm{jump}} u\big)(x) v(x) + \Gamma_t^{\mathrm{jump}}(u,v)(x) \Big] p_t(\mathrm{d}x) = 0, \tag{28}$$

where the superscript "jump" picks out the integral term of the corresponding operator. Expanding the three integrals,

$$\begin{aligned} 0 = &\iint v(x)\big(u(y)-u(x)\big)\vec{J}_t(x,\mathrm{d}y)p_t(\mathrm{d}x) + \iint v(x)\big(u(y)-u(x)\big)\overleftarrow{J}_t(x,\mathrm{d}y)p_t(\mathrm{d}x) \\ &+ \iint \big(u(y)-u(x)\big)\big(v(y)-v(x)\big)\vec{J}_t(x,\mathrm{d}y)p_t(\mathrm{d}x). \end{aligned}$$

The algebraic identity

$$v(x)\big(u(y)-u(x)\big) + \big(u(y)-u(x)\big)\big(v(y)-v(x)\big) = \big(u(y)-u(x)\big)v(y)$$

collapses the first and third terms of the right-hand side, leaving

$$\iint v(x)\big(u(y)-u(x)\big)\overleftarrow{J}_t(x,\mathrm{d}y)p_t(\mathrm{d}x) = -\iint \big(u(y)-u(x)\big)v(y)\vec{J}_t(x,\mathrm{d}y)p_t(\mathrm{d}x).$$

Swap the dummy variables $x \leftrightarrow y$ on the right and use $u(x)-u(y) = -\big(u(y)-u(x)\big)$:

$$\iint v(x)\big(u(y)-u(x)\big)\overleftarrow{J}_t(x,\mathrm{d}y)p_t(\mathrm{d}x) = \iint v(x)\big(u(y)-u(x)\big)\vec{J}_t(y,\mathrm{d}x)p_t(\mathrm{d}y).$$

The class of test functions $(x,y) \mapsto v(x)\big(u(y)-u(x)\big)$ for $u, v \in \mathcal{U}$ is measure-determining off the diagonal, so the two measures on $\mathbb{R}^d \times \mathbb{R}^d$ coincide:

$$p_t(\mathrm{d}x)\overleftarrow{J}_t(x,\mathrm{d}y) = p_t(\mathrm{d}y)\vec{J}_t(y,\mathrm{d}x), \tag{29}$$

which is exactly the flux identity (23).

Step 2: anchored jumps. We now specialize (29) to the structure of the SJD forward kernel. By Assumption C.1(ii), $\vec{J}_t(x,\cdot) \equiv 0$ for $x \in \mathsf{X}_{\mathcal{A}}$ and $\vec{J}_t(a,\mathrm{d}y) = \vec{\lambda}_t(a)R_t(\mathrm{d}y \,|\, a)$ for $a \in \mathcal{A}$. Fix $a \in \mathcal{A}$ and apply (29) with $\mathrm{d}x = \{a\}$ and $\mathrm{d}y \subset \mathsf{X}_{\mathcal{A}}$:

$$p_t(\mathrm{d}y)\overleftarrow{J}_t(y,\{a\}) = p_t(\{a\})\vec{J}_t(a,\mathrm{d}y) = m_t(a)\vec{\lambda}_t(a)R_t(\mathrm{d}y \,|\, a), \tag{30}$$

which is exactly (24). If p_t admits density $p_t(y)$ on $\mathsf{X}_{\mathcal{A}}$ and $R_t(\mathrm{d}y \,|\, a) = r_t(y \,|\, a)\mathrm{d}y$, dividing both sides of (30) by $p_t(y)\mathrm{d}y$ gives the closed form

$$\overleftarrow{J}_t(y,\{a\}) = \frac{m_t(a)}{p_t(y)}\vec{\lambda}_t(a)r_t(y \,|\, a), \qquad y \in \mathsf{X}_{\mathcal{A}},\ a \in \mathcal{A}.$$

Conversely, applying (29) with $\mathrm{d}y = \{a\}$ and any $\mathrm{d}x \subset \mathsf{X}_{\mathcal{A}}$,

$$p_t(\{a\})\overleftarrow{J}_t(a,\mathrm{d}x) = p_t(\mathrm{d}x)\vec{J}_t(x,\{a\}) = 0,$$

since the forward kernel does not jump out of $\mathsf{X}_{\mathcal{A}}$. Hence $\overleftarrow{J}_t(a,\mathsf{X}_{\mathcal{A}}) = 0$: anchors are absorbing in reverse time.

The total backward sticky-jump rate at $y \in \mathsf{X}_{\mathcal{A}}$ is $\overleftarrow{\lambda}^{\star}_t(y) := \sum_{a\in\mathcal{A}} \overleftarrow{J}_t(y,\{a\})$. Summing (30) over a yields the identity

$$\sum_{a\in\mathcal{A}} m_t(a)\vec{\lambda}_t(a)r_t(y \,|\, a) = p_t(y)\overleftarrow{\lambda}^{\star}_t(y), \qquad y \in \mathsf{X}_{\mathcal{A}}, \tag{31}$$

which we will use in Step 3.

Step 3: diffusion drift. Take $u, v \in \mathcal{U}$ with supports contained in $\mathsf{X}_{\mathcal{A}}$, so that $u(a) = v(a) = 0$ for every $a \in \mathcal{A}$. We catalogue the contributions of (26) term by term.

Forward generator. On $\mathsf{X}_{\mathcal{A}}$, $\vec{L}_t u(x) = \vec{b}_t(x) \cdot \nabla u(x) + \frac{1}{2} g_t^2 \Delta u(x)$ since $\vec{J}_t(x, \cdot) \equiv 0$. At $x = a \in \mathcal{A}$, the contribution $\vec{L}_t u(a) v(a)$ vanishes because $v(a) = 0$.

Backward generator. On $\mathsf{X}_{\mathcal{A}}$, the ansatz (27), the result of Step 2, and $u(a) = 0$ give

$$\begin{aligned} \overleftarrow{L}_t u(x) &= \tilde{b}_t(x) \cdot \nabla u(x) + \tfrac{1}{2} g_t^2 \Delta u(x) + \sum_{a \in \mathcal{A}} \big(u(a) - u(x)\big) \overleftarrow{J}_t(x, \{a\}) \\ &= \tilde{b}_t \cdot \nabla u + \tfrac{1}{2} g_t^2 \Delta u - u \overleftarrow{\lambda}_t^\star. \end{aligned} \tag{32}$$

The sticky jumps to anchors thus contribute the additional term $-u(x)\overleftarrow{\lambda}_t^\star(x)$ to $\overleftarrow{L}_t u$ on $\mathsf{X}_{\mathcal{A}}$, even though u is supported off the anchors. At $x = a$, $\overleftarrow{L}_t u(a) v(a) = 0$ since $v(a) = 0$.

Carré du champ. On $\mathsf{X}_{\mathcal{A}}$, $\Gamma_t(u, v)(x) = g_t^2 \nabla u \cdot \nabla v$ since $\vec{J}_t(x, \cdot) \equiv 0$. At $x = a \in \mathcal{A}$, $u(a) = v(a) = 0$ leaves the nontrivial contribution

$$\Gamma_t(u, v)(a) = \int_{\mathbb{R}^d} u(y) v(y) \vec{\lambda}_t(a) R_t(\mathrm{d}y \,|\, a).$$

Combining the three contributions in (26),

$$\begin{aligned} 0 = \int_{\mathsf{X}_{\mathcal{A}}} \Big[(\vec{b}_t + \tilde{b}_t) \cdot \nabla u v + g_t^2 \Delta u v + g_t^2 \nabla u \cdot \nabla v - u v \overleftarrow{\lambda}_t^\star \Big] p_t(x) \mathrm{d}x \\ + \sum_{a \in \mathcal{A}} m_t(a) \vec{\lambda}_t(a) \int_{\mathsf{X}_{\mathcal{A}}} u(y) v(y) r_t(y \,|\, a) \mathrm{d}y. \end{aligned} \tag{33}$$

The atomic carré-du-champ contribution and the backward sticky-jump contribution cancel exactly: by (31),

$$\sum_{a \in \mathcal{A}} m_t(a) \vec{\lambda}_t(a) \int_{\mathsf{X}_{\mathcal{A}}} u v r_t(y \,|\, a) \mathrm{d}y = \int_{\mathsf{X}_{\mathcal{A}}} u(y) v(y) \overleftarrow{\lambda}_t^\star(y) p_t(y) \mathrm{d}y,$$

so the second line of (33) kills the $-uv\overleftarrow{\lambda}_t^\star$ term in the bulk integral. What remains is the standard diffusion-IbP identity

$$\int_{\mathsf{X}_{\mathcal{A}}} \Big[(\vec{b}_t + \tilde{b}_t) \cdot \nabla u v + g_t^2 \Delta u v + g_t^2 \nabla u \cdot \nabla v \Big] p_t(x) \mathrm{d}x = 0. \tag{34}$$

Integrating by parts in space (boundary terms vanish since u, v are compactly supported in $\mathsf{X}_{\mathcal{A}}$),

$$\int v \Delta u p_t \mathrm{d}x = -\int \nabla u \cdot \nabla (v p_t) \mathrm{d}x = -\int \nabla u \cdot \big(p_t \nabla v + v \nabla p_t\big) \mathrm{d}x,$$

so the bracketed combination simplifies:

$$\int g_t^2 \big(v \Delta u + \nabla u \cdot \nabla v\big) p_t \mathrm{d}x = -g_t^2 \int v \nabla u \cdot \nabla p_t \mathrm{d}x.$$

Therefore (34) reduces to

$$0 = \int_{\mathsf{X}_{\mathcal{A}}} v(x) A(x) \cdot \nabla u(x) \mathrm{d}x, \qquad A(x) := p_t(x) \big(\vec{b}_t(x) + \tilde{b}_t(x)\big) - g_t^2 \nabla p_t(x).$$

A standard distributional argument forces $A \equiv 0$ on $\mathsf{X}_{\mathcal{A}}$: for any $\varphi \in C_c^\infty(\mathsf{X}_{\mathcal{A}})$ and any cutoff $\chi \in C_c^\infty(\mathsf{X}_{\mathcal{A}})$ with $\chi \equiv 1$ on $\mathrm{supp}(\varphi)$, set $u(x) := x_i \chi(x)$ and $v(x) := \varphi(x)$, so that $\nabla u = e_i$ on $\mathrm{supp}(\varphi)$ and the identity above reads $0 = \int_{\mathsf{X}_{\mathcal{A}}} \varphi A_i \mathrm{d}x$. As φ ranges over $C_c^\infty(\mathsf{X}_{\mathcal{A}})$, $A_i = 0$ a.e. on $\mathsf{X}_{\mathcal{A}}$; ranging $i \in \{1, \dots, d\}$ yields $A \equiv 0$ a.e. Dividing by $p_t(x) > 0$,

$$\tilde{b}_t(x) = -\vec{b}_t(x) + g_t^2 \nabla \log p_t(x), \qquad x \in \mathsf{X}_{\mathcal{A}}, \tag{35}$$

which is the claimed reverse drift, recovering the score-based time-reversal of Song et al. [2021] on the off-anchor region. □

Corollary C.2 (Probability-flow dynamics with sticky jumps)**.** *Let Assumption C.1 hold and let $\overleftarrow{J}_t$ be the reverse jump kernel of Theorem 2.6. Write*

$$\overleftarrow{b}_t^{\mathrm{PF}}(x) := -\vec{b}_t(x) + \tfrac{1}{2} g_t^2 \nabla \log p_t(x), \qquad x \in \mathsf{X}_{\mathcal{A}},$$

and assume in addition that $\overleftarrow{b}_t^{\mathrm{PF}}$ is locally Lipschitz in x on $\mathsf{X}_{\mathcal{A}}$, that solutions of (36) *below exist on $[0,T]$ and remain in $\mathsf{X}_{\mathcal{A}}$ for p_T^{ac}-a.e. initial point, and that $\overleftarrow{\lambda}_t^\star$ is locally bounded on $\mathsf{X}_{\mathcal{A}}$. Define the probability-flow process $(\widetilde{X}_\tau)_{\tau\in[0,T]}$, with $t = T - \tau$ and $\widetilde{X}_0 \sim p_T$, evolving on $\mathsf{X}_{\mathcal{A}}$ by the ordinary differential equation*

$$\mathrm{d}\widetilde{X}_\tau = \overleftarrow{b}_t^{\mathrm{PF}}(\widetilde{X}_\tau)\,\mathrm{d}\tau, \tag{36}$$

and committing to anchors with the sticky-jump intensities of the exact reversal: from $y \in \mathsf{X}_{\mathcal{A}}$, jumps into $a \in \mathcal{A}$ occur at rate $\overleftarrow{J}_t(y,\{a\})$ given by (8)*, and anchors are absorbing. Then $\widetilde{X}_\tau \sim p_{T-\tau}$ for every $\tau \in [0,T]$; in particular, the probability-flow process and the exact reversal of Theorem 2.6 share all time marginals.*

Proof. Throughout, $t = T - \tau$. Set $\rho_\tau := p_{T-\tau}$, so that

$$\rho_\tau = \rho_\tau^{\mathrm{ac}} + \sum_{a\in\mathcal{A}} m_t(a)\,\delta_a,$$

and write $\rho_\tau(x) = p_t(x)$ for the density of ρ_τ^{ac} on $\mathsf{X}_{\mathcal{A}}$. Let $\overleftarrow{b}_t := -\vec{b}_t + g_t^2 \nabla \log p_t$ denote the reverse drift (35). Spatial derivatives of ρ_τ are understood in the sense of distributions on $\mathsf{X}_{\mathcal{A}}$.

Step 1: Fokker–Planck equation of the reversal on $\mathsf{X}_{\mathcal{A}}$. Let $u \in C_c^2(\mathsf{X}_{\mathcal{A}})$. Since u vanishes on $\mathcal{A}$, (32) gives, for $x \in \mathsf{X}_{\mathcal{A}}$,

$$\overleftarrow{L}_t u(x) = \overleftarrow{b}_t(x)\cdot\nabla u(x) + \tfrac{1}{2} g_t^2 \Delta u(x) + \sum_{a\in\mathcal{A}} \big(u(a) - u(x)\big)\overleftarrow{J}_t(x,\{a\}) = \overleftarrow{b}_t \cdot \nabla u + \tfrac{1}{2} g_t^2 \Delta u - \overleftarrow{\lambda}_t^\star\, u,$$

while $\overleftarrow{L}_t u(a) = 0$ for $a \in \mathcal{A}$, since the diffusion part vanishes off $\mathsf{X}_{\mathcal{A}}$ and $\overleftarrow{J}_t(a,\cdot) \equiv 0$ (anchors are absorbing). By Theorem 2.6, $\overleftarrow{X}$ is Markov with generator $\overleftarrow{L}_t$ and $\overleftarrow{X}_\tau \sim \rho_\tau$, so

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\,\mathbb{E}\big[u(\overleftarrow{X}_\tau)\big] = \mathbb{E}\big[(\overleftarrow{L}_t u)(\overleftarrow{X}_\tau)\big].$$

Because $u(a) = 0$ and $\overleftarrow{L}_t u(a) = 0$, the atoms of ρ_τ contribute to neither side, and

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\int_{\mathsf{X}_{\mathcal{A}}} u\,\rho_\tau\,\mathrm{d}x = \int_{\mathsf{X}_{\mathcal{A}}} (\overleftarrow{L}_t u)\,\rho_\tau\,\mathrm{d}x. \tag{37}$$

Integrating by parts in the drift term and applying Green's second identity to the Laplacian term (there are no boundary terms, since $\operatorname{supp} u$ is compact in $\mathsf{X}_{\mathcal{A}}$),

$$\int_{\mathsf{X}_{\mathcal{A}}} (\overleftarrow{L}_t u)\,\rho_\tau\,\mathrm{d}x = \int_{\mathsf{X}_{\mathcal{A}}} u\,\overleftarrow{L}_t^\star \rho_\tau\,\mathrm{d}x, \qquad \overleftarrow{L}_t^\star v := -\nabla\cdot\big(\overleftarrow{b}_t\, v\big) + \tfrac{1}{2} g_t^2 \Delta v - \overleftarrow{\lambda}_t^\star\, v,$$

where $\overleftarrow{L}_t^\star$ is the $L^2(\mathrm{d}x)$ adjoint of $\overleftarrow{L}_t$ on $C_c^2(\mathsf{X}_{\mathcal{A}})$. As (37) holds for every $u \in C_c^2(\mathsf{X}_{\mathcal{A}})$,

$$\partial_\tau \rho_\tau = \overleftarrow{L}_t^\star \rho_\tau = -\nabla\cdot\big(\overleftarrow{b}_t\,\rho_\tau\big) + \tfrac{1}{2} g_t^2 \Delta \rho_\tau - \overleftarrow{\lambda}_t^\star\,\rho_\tau \quad \text{on } \mathsf{X}_{\mathcal{A}}. \tag{38}$$

Step 2: Continuity-equation form. Since $\rho_\tau \nabla \log p_t = \nabla \rho_\tau$, we have $\nabla\cdot\big(g_t^2 \rho_\tau \nabla \log p_t\big) = g_t^2 \Delta \rho_\tau$. Substituting $\overleftarrow{b}_t = \overleftarrow{b}_t^{\mathrm{PF}} + \frac{1}{2} g_t^2 \nabla \log p_t$ into (38) therefore gives

$$\partial_\tau \rho_\tau = -\nabla\cdot\big(\overleftarrow{b}_t^{\mathrm{PF}}\,\rho_\tau\big) - \overleftarrow{\lambda}_t^\star\,\rho_\tau \quad \text{on } \mathsf{X}_{\mathcal{A}}. \tag{39}$$

Step 3: Anchor masses. Since the forward process does not re-attach to anchors, $m_t(a) = p_0(a) S_a(t)$, so $\partial_t m_t(a) = -\vec{\lambda}_t(a)\, m_t(a)$. With $\rho_\tau(\{a\}) = m_{T-\tau}(a)$,

$$\partial_\tau \rho_\tau(\{a\}) = \vec{\lambda}_t(a)\, m_t(a) = \int_{\mathsf{X}_{\mathcal{A}}} \overleftarrow{J}_t(y,\{a\})\,\rho_\tau(y)\,\mathrm{d}y, \tag{40}$$

where the last equality uses (8) with $\rho_\tau(y) = p_t(y)$, together with $\int_{\mathsf{X}_{\mathcal{A}}} r_t(y \mid a)\,\mathrm{d}y = 1$.

Step 4: Law of the probability-flow process. Let q_τ denote the law of $\widetilde{X}_\tau$, with absolutely continuous part q_τ^{ac} on $\mathsf{X}_{\mathcal{A}}$ and atoms $q_\tau(\{a\})$. Between jumps, $\widetilde{X}$ follows (36); from $y \in \mathsf{X}_{\mathcal{A}}$ it jumps to a at rate $\overleftarrow{J}_t(y, \{a\})$, hence leaves $\mathsf{X}_{\mathcal{A}}$ at total rate $\overleftarrow{\lambda}_t^\star(y)$; and anchors are absorbing. Its generator is therefore

$$L_t^{\mathrm{PF}} u(x) = \mathbf{1}_{\mathsf{X}_{\mathcal{A}}}(x) \Big[\overleftarrow{b}_t^{\mathrm{PF}}(x) \cdot \nabla u(x) + \sum_{a \in \mathcal{A}} \big(u(a) - u(x)\big) \overleftarrow{J}_t(x, \{a\}) \Big].$$

Repeating Step 1 with L_t^{PF} in place of $\overleftarrow{L}_t$ (there is no Laplacian term, as (36) carries no noise) shows that q^{ac} satisfies (39); taking $u = \mathbf{1}_{\{a\}}$, for which $L_t^{\mathrm{PF}} u = \overleftarrow{J}_t(\cdot, \{a\})$ on $\mathsf{X}_{\mathcal{A}}$ and $L_t^{\mathrm{PF}} u(a) = 0$, shows that q satisfies (40). Moreover $q_0 = p_T = \rho_0$.

Step 5: Uniqueness. The coefficients $\overleftarrow{b}_t^{\mathrm{PF}}$ and $\overleftarrow{\lambda}_t^\star$ of (39) are fixed functions determined by p_t, so (39) is a linear transport equation with a killing term. Under the stated regularity it admits a unique nonnegative solution for a given initial datum, namely the one given by the method of characteristics along the flow of (36). Since ρ^{ac} and q^{ac} both solve (39) with initial datum p_T^{ac}, we get $q_\tau^{\mathrm{ac}} = \rho_\tau^{\mathrm{ac}}$ for all $\tau \in [0, T]$. The right-hand sides of (40) then coincide for q and ρ, and since $q_0(\{a\}) = \rho_0(\{a\}) = m_T(a)$, integrating in τ gives $q_\tau(\{a\}) = \rho_\tau(\{a\})$ for every $a \in \mathcal{A}$. Hence $q_\tau = \rho_\tau = p_{T-\tau}$ for all $\tau \in [0, T]$.

Equivalently, in forward time (39) reads

$$\partial_t p_t = -\nabla \cdot \Big[\Big(\vec{b}_t - \tfrac{1}{2} g_t^2 \nabla \log p_t \Big) p_t \Big] + \overleftarrow{\lambda}_t^\star \, p_t \quad \text{on } \mathsf{X}_{\mathcal{A}}.$$

The term $+\overleftarrow{\lambda}_t^\star p_t$ is the source created by unsticking: mass leaves the anchors and enters $\mathsf{X}_{\mathcal{A}}$. In reverse time it becomes the sink $-\overleftarrow{\lambda}_t^\star \rho_\tau$: mass leaves $\mathsf{X}_{\mathcal{A}}$ through sticky jumps, and (40) deposits it back on the anchors. □

Remark C.3 (Guidance). Because every reverse quantity is an analytic functional of the single classifier P_θ, conditional guidance requires no additional machinery. For a conditioning signal c and a guidance weight w, set

$$P_\theta^w(\cdot \mid y, t) \propto P_\theta(\cdot \mid y, t, c)^w P_\theta(\cdot \mid y, t, \varnothing)^{1-w},$$

a mixture of logits, and substitute P_θ^w for P_θ in the induced score ((63) and (64)) and in the plug-ins (18). One guided posterior then steers the drift, the commit rate, and the destination jointly. Dieleman et al. [2022] demonstrate the continuous half of this construction. Our experiments are unconditional, so we state the mechanism without an empirical claim.

C.2 Proof of Theorem 3.2

We restate Theorem 3.2 with the relevant SJD-specific objects in scope, then give the proof.

Theorem (denoising hazard matching, restated)**.** *Let $\vec{X}_0 \sim p_0$ on $\mathcal{A}$ and, conditional on $\vec{X}_0$, let $\vec{X}_t$ be the SJD state at time t restricted to the continuous region $\mathsf{X}_{\mathcal{A}}$, with conditional density $p_t^{\mathrm{ac}}(\cdot \mid \vec{X}_0)$ given by the unstick-time mixture of Remark 2.4. For $y \in \mathsf{X}_{\mathcal{A}}$ and $x_0 \in \mathcal{A}$, define the DHM target*

$$\widehat{\lambda}_t(y, x_0) := \frac{\vec{\lambda}_t(x_0) S_{x_0}(t) r_{x_0}(y, t)}{p_t^{\mathrm{ac}}(y \mid x_0)}, \tag{41}$$

with $S_{x_0}(t) = \exp\big(-\int_0^t \vec{\lambda}_s(x_0) \mathrm{d}s\big)$. Then for almost every $y \in \mathsf{X}_{\mathcal{A}}$,

$$\mathbb{E}\big[\widehat{\lambda}_t(\vec{X}_t, \vec{X}_0) \;\big|\; \vec{X}_t = y\big] = \overleftarrow{\lambda}_t^\star(y), \tag{42}$$

where $\overleftarrow{\lambda}_t^\star(y)$ is the total reverse jump rate of (10)*.*

Proof. The proof is a direct application of Bayes' rule for the SJD posterior. The crucial point is that the $p_t^{\mathrm{ac}}(y \mid a)$ in the denominator of (41) is the actual conditional density of the SJD continuous part and not an auxiliary corruption kernel. The same object therefore reappears in the denoising posterior (13), and the two factors cancel.

Because the SJD forward dynamics never re-attach to anchors, the only way for a particle to be at anchor a at time t is to have started at a and never unstuck. Hence

$$m_t(a) = p_0(a)S_a(t), \qquad a \in \mathcal{A}. \tag{43}$$

Let $y \in \mathsf{X}_{\mathcal{A}}$. Since $\vec{X}_0$ takes values in the finite set $\mathcal{A}$, the joint density of $(\vec{X}_0, \vec{X}_t)$ on $\mathcal{A} \times \mathsf{X}_{\mathcal{A}}$ is $p_0(a)p_t^{\mathrm{ac}}(y \,|\, a)$, and the marginal continuous density of $\vec{X}_t$ is

$$p_t(y) = \sum_{a \in \mathcal{A}} p_0(a)p_t^{\mathrm{ac}}(y \,|\, a), \qquad y \in \mathsf{X}_{\mathcal{A}}.$$

Bayes' rule gives the denoising posterior (13),

$$p_t(a \,|\, y) = \Pr\left(\vec{X}_0 = a \;\middle|\; \vec{X}_t = y\right) = \frac{p_0(a)p_t^{\mathrm{ac}}(y \,|\, a)}{p_t(y)}. \tag{44}$$

Because $\vec{X}_0$ is supported on $\mathcal{A}$, the conditional expectation in (42) is a finite sum:

$$\mathbb{E}\big[\widehat{\lambda}_t(\vec{X}_t, \vec{X}_0) \;\big|\; \vec{X}_t = y\big] = \sum_{a \in \mathcal{A}} \widehat{\lambda}_t(y, a)p_t(a \,|\, y). \tag{45}$$

Substituting (41) and (44) into (45),

$$\sum_{a \in \mathcal{A}} \widehat{\lambda}_t(y, a)p_t(a \,|\, y) = \sum_{a \in \mathcal{A}} \frac{\vec{\lambda}_t(a)S_a(t)r_a(y, t)}{p_t^{\mathrm{ac}}(y \,|\, a)} \cdot \frac{p_0(a)p_t^{\mathrm{ac}}(y \,|\, a)}{p_t(y)}.$$

The factor $p_t^{\mathrm{ac}}(y \,|\, a)$ cancels exactly:

$$= \frac{1}{p_t(y)} \sum_{a \in \mathcal{A}} p_0(a)S_a(t)\vec{\lambda}_t(a)r_a(y, t).$$

Applying the mass identity (43),

$$= \frac{1}{p_t(y)} \sum_{a \in \mathcal{A}} m_t(a)\vec{\lambda}_t(a)r_a(y, t) = \overleftarrow{\lambda}_t^{\star}(y),$$

where the last equality is the definition of the total reverse jump rate (10). This proves (42). □

C.3 Rao–Blackwellized plug-in

Theorem 3.2 expresses the total reverse hazard $\overleftarrow{\lambda}_t^{\star}(y)$ as a conditional expectation under the SJD forward pairing. We now record three properties that justify the plug-in estimators (18): a per-anchor flux identity, consistency of the plug-in when the classifier matches the true posterior, and the Rao–Blackwell variance reduction.

Setup. Throughout this subsection we fix $t \in (0, T]$ and $y \in \mathsf{X}_{\mathcal{A}}$, and write $\widehat{\lambda}_t(y, a)$ for the DHM target (41) and $p_t(a \,|\, y)$ for the SJD denoising posterior (44). We assume the support condition $r_a(y, t) > 0 \Rightarrow p_t^{\mathrm{ac}}(y \,|\, a) > 0$, which is automatic whenever the convolution semigroup $r_\tau * K_{\tau \to t}$ has full support on $\mathsf{X}_{\mathcal{A}}$ (in particular for the VP-matched family of Example 2.2). Given any probability vector $P_\theta(\cdot \,|\, y, t)$ on $\mathcal{A}$, recall the plug-in estimators

$$\Lambda_{\mathrm{plug},\theta}(a \,|\, y, t) := \widehat{\lambda}_t(y, a)P_\theta(a \,|\, y, t), \qquad \lambda_{\mathrm{plug},\theta}(y, t) := \sum_{a \in \mathcal{A}} \Lambda_{\mathrm{plug},\theta}(a \,|\, y, t). \tag{46}$$

Per-anchor flux identity. The following identity is the per-anchor analogue of Theorem 3.2: the true reverse *per-anchor* intensity factors exactly as the DHM target times the denoising posterior.

Lemma C.4 (Per-anchor identity)**.** *For every $a \in \mathcal{A}$ and almost every $y \in \mathsf{X}_{\mathcal{A}}$,*

$$\Lambda_t^{\star}(a \,|\, y) = \widehat{\lambda}_t(y, a)p_t(a \,|\, y). \tag{47}$$

Proof. Substitute (41) for $\widehat{\lambda}_t(y,a)$ and (44) for $p_t(a \mid y)$:

$$\widehat{\lambda}_t(y,a)p_t(a \mid y) = \frac{\vec{\lambda}_t(a)S_a(t)r_a(y,t)}{p_t^{\mathrm{ac}}(y \mid a)} \cdot \frac{p_0(a)p_t^{\mathrm{ac}}(y \mid a)}{p_t(y)} = \frac{p_0(a)S_a(t)\vec{\lambda}_t(a)r_a(y,t)}{p_t(y)}.$$

Using $m_t(a) = p_0(a)S_a(t)$ from (43), the right-hand side is exactly $\Lambda_t^\star(a \mid y)$ as defined in (9). □

Lemma C.4 is the structural fact that makes a single classifier sufficient: the DHM target $\widehat{\lambda}_t(y,a)$ is precisely the Radon–Nikodym factor that converts a probability mass $p_t(a \mid y)$ (a posterior over data labels) into a per-anchor jump rate $\Lambda_t^\star(a \mid y)$ (jumps per unit time into anchor a).

Consistency of the plug-in. Lemma C.4 immediately yields consistency: substituting the true posterior into the plug-in recovers the true reverse intensities.

Proposition C.5 (Consistency)**.** *If $P_\theta(a \mid y,t) = p_t(a \mid y)$ for every $a \in \mathcal{A}$ and almost every $y \in \mathsf{X}_{\mathcal{A}}$, then*

$$\Lambda_{\mathrm{plug},\theta}(a \mid y,t) = \Lambda_t^\star(a \mid y), \qquad \lambda_{\mathrm{plug},\theta}(y,t) = \overleftarrow{\lambda}_t^\star(y), \qquad \pi_{\mathrm{plug},\theta}(a \mid y,t) = \pi_t^\star(a \mid y). \tag{48}$$

Proof. The per-anchor equality is Lemma C.4. The total-rate equality follows by summing over a, and the allocation equality by normalizing. □

Proposition C.5 pins down what the classifier needs to learn: the cross-entropy loss (14) is a strictly proper scoring rule whose Bayes optimum is exactly the posterior $p_t(\cdot \mid y)$, and the plug-in inherits all three identities of (48) as soon as P_θ matches that posterior.

Rao–Blackwellization. The DHM identity (42) also yields an unbiased Monte Carlo estimator of $\overleftarrow{\lambda}_t^\star(y)$ from a single $(\vec{X}_0, \vec{X}_t)$ pair: the random variable

$$Z := \widehat{\lambda}_t(\vec{X}_t, \vec{X}_0). \tag{49}$$

By Theorem 3.2, $\mathbb{E}[Z \mid \vec{X}_t = y] = \overleftarrow{\lambda}_t^\star(y)$. The plug-in (46) replaces Z by its conditional expectation given $(\vec{X}_t, t)$, with the unknown posterior approximated by P_θ. The next proposition makes this Rao–Blackwell relationship precise.

Proposition C.6 (Rao–Blackwellization)**.** *Let $Z = \widehat{\lambda}_t(\vec{X}_t, \vec{X}_0)$ as in* (49)*, with $(\vec{X}_0, \vec{X}_t)$ paired by the SJD forward law. Assume $\mathbb{E}[Z^2 \mid t] < \infty$. Then:*

(i) (Conditional mean.) $\mathbb{E}[Z \mid \vec{X}_t, t] = \overleftarrow{\lambda}_t^\star(\vec{X}_t)$ *almost surely.*

(ii) (Variance decomposition.) *The law of total variance gives*

$$\mathrm{Var}(Z \mid t) = \mathrm{Var}\left(\overleftarrow{\lambda}_t^\star(\vec{X}_t) \;\middle|\; t\right) + \mathbb{E}\left[\,\mathrm{Var}(Z \mid \vec{X}_t, t) \;\middle|\; t\right]. \tag{50}$$

(iii) (Variance reduction.) *Consequently,*

$$\mathrm{Var}\left(\overleftarrow{\lambda}_t^\star(\vec{X}_t) \;\middle|\; t\right) \le \mathrm{Var}(Z \mid t), \tag{51}$$

with equality if and only if Z is almost surely a function of $(\vec{X}_t, t)$, i.e. $\mathrm{Var}(Z \mid \vec{X}_t, t) = 0$ a.s.

Proof. For (i), apply the discrete expansion (45) with $y = \vec{X}_t$:

$$\mathbb{E}[Z \mid \vec{X}_t, t] = \sum_{a \in \mathcal{A}} \widehat{\lambda}_t(\vec{X}_t, a)p_t(a \mid \vec{X}_t) = \overleftarrow{\lambda}_t^\star(\vec{X}_t),$$

where the second equality is Theorem 3.2. Statement (ii) is the law of total variance, conditional on t, with conditioning σ-field $\sigma(\vec{X}_t, t)$ and the substitution $\mathbb{E}[Z \mid \vec{X}_t, t] = \overleftarrow{\lambda}_t^\star(\vec{X}_t)$ from (i). Statement (iii) follows from (50) since the second term is nonnegative, with equality iff that term vanishes. □

Interpretation. The single-sample target Z in (49) is unbiased for $\overleftarrow{\lambda}^\star_t(\vec{X}_t)$ but carries the within-cell noise $\mathrm{Var}(Z \mid \vec{X}_t, t)$: at a fixed $\vec{X}_t = y$, different latent draws of $\vec{X}_0 \sim p_t(\cdot \mid y)$ produce different values $\widehat{\lambda}_t(y, \vec{X}_0)$. The Rao–Blackwellized estimator $\sum_a \widehat{\lambda}_t(y,a) p_t(a \mid y)$ averages this noise out exactly, replacing Z by its conditional mean and saving the second term in (50). The plug-in (46) performs this Rao–Blackwellization with the true posterior replaced by the trained classifier P_θ:

$$\lambda_{\mathrm{plug},\theta}(y,t) = \underbrace{\sum_{a \in \mathcal{A}} \widehat{\lambda}_t(y,a) p_t(a \mid y)}_{=\overleftarrow{\lambda}^\star_t(y) \text{ by Theorem 3.2}} + \underbrace{\sum_{a \in \mathcal{A}} \widehat{\lambda}_t(y,a) \big(P_\theta(a \mid y, t) - p_t(a \mid y)\big)}_{\text{classifier error}}. \tag{52}$$

The first term is exact by Rao–Blackwell; the residual is controlled by the gap between P_θ and the Bayes-optimal posterior, which the cross-entropy loss (14) drives to zero in the well-specified, infinite-data limit. In particular, by Cauchy–Schwarz,

$$\big|\lambda_{\mathrm{plug},\theta}(y,t) - \overleftarrow{\lambda}^\star_t(y)\big| \le \Big(\sum_{a \in \mathcal{A}} \widehat{\lambda}_t(y,a)^2 \Big)^{1/2} \Big(\sum_{a \in \mathcal{A}} \big(P_\theta(a \mid y, t) - p_t(a \mid y)\big)^2 \Big)^{1/2},$$

so any consistent classifier yields a consistent plug-in hazard, and the conversion from posterior error to hazard error is mediated by the analytic factor $\widehat{\lambda}_t(y, \cdot)$.

Two practical consequences follow. First, training reduces to a single cross-entropy objective on a discrete classifier, with no auxiliary regression head for the (unbounded, heavy-tailed) hazard target Z. Second, all SJD-specific structure—the forward hazard, the unsticking density, and the unstick-time mixture $p_t^{\mathrm{ac}}(y \mid a)$—enters only through the analytic reweighting $\widehat{\lambda}_t(y,a)$, which is computed once per reverse step using the closed-form quadrature of Appendix D.2. The classifier never sees these quantities directly; it learns only the discrete posterior $p_t(\cdot \mid y)$, and the analytic reweighting converts that into the rates and intensities required by the reverse SDE.

C.4 Proof of Proposition 4.1

Proposition (The plain hybrid coincides with CADD, restated)**.** *Consider the VP-matched SJD of Example 2.2 at $(W, \eta) = (I, 1)$ with anchor-agnostic hazard $\vec{\lambda}(t)$ and survival $S(t)$, and CADD with masking survival α_t, continuous kernel $\mathcal{N}(\sqrt{\bar\gamma_t} E(x_0), (1-\bar\gamma_t) I_d)$ at masked positions, and a common embedding map E. Identify $S(t) \leftrightarrow \alpha_t$ and $(\alpha(t), \sigma^2(t)) \leftrightarrow (\sqrt{\bar\gamma_t}, 1-\bar\gamma_t)$. Then:*

- *(i) the time-t laws of the corrupted state given the data coincide for every t, and the SJD cross-entropy objective of Algorithm 1 equals CADD's cross-entropy objective;*
- *(ii) the reverse process of Theorem 2.6 reduces to a commit rate $\vec{\lambda}(t) S(t)/(1-S(t))$ shared across anchors, an allocation equal to the classifier posterior, and a reverse SDE whose score is the residual to the posterior-mean embedding;*
- *(iii) CADD's sampler is a first-order discretization of (ii), so the two generative laws coincide in the small-step limit.*

(i) Forward laws and objectives. Fix a position and condition on $\vec{X}_0$. Under $S(t) \leftrightarrow \alpha_t$ the stuck probability equals the survival of CADD's absorbing chain. Conditionally on having unstuck, (22) gives the off-anchor location the law $\mathcal{N}(\alpha(t) a, \sigma^2(t) I_d)$, which is CADD's masked-position latent $\mathcal{N}(\sqrt{\bar\gamma_t} E(a), (1-\bar\gamma_t) I_d)$ under $(\alpha(t), \sigma^2(t)) \leftrightarrow (\sqrt{\bar\gamma_t}, 1-\bar\gamma_t)$. Positions are conditionally independent given $\vec{X}_0$ in both constructions, so the time-t joint laws agree. The state map is a measurable bijection (a stuck coordinate sits exactly on its anchor and thus carries the token identity, as does an unmasked token; an unstuck coordinate carries the latent), so the Bayes posteriors over the clean sequence agree, and both objectives draw t uniformly and apply unweighted cross-entropy at unstuck (masked) positions: Algorithm 1 coincides with CADD's loss.

(ii) The reverse process at $(I, 1)$. At $\eta = 1$, $v_t(\tau) = \sigma^2(t)$ by (20), so the spatial factor of the DHM target equals $r_t / p_t^{\mathrm{ac}} = 1/(1-S(t))$ by (22), giving $\widehat{\lambda}_t(y,a) = \vec{\lambda}(t) S(t)/(1-S(t))$ for every a and y. The plug-in total rate (18) is therefore anchor-independent and the allocation (19) reduces to $P_\theta(\cdot \mid y, t)$. Since $v_t(\tau)$ no longer depends on τ, the time-posterior weight in (63) integrates to one

and the classifier-induced score reduces to

$$s_\theta(y,t) = -\frac{y - \alpha(t) m_\theta(y,t)}{\sigma^2(t)}, \qquad m_\theta(y,t) := \sum_{a \in \mathcal{A}} P_\theta(a \,|\, y, t) a,$$

the residual to the posterior-mean anchor, exact at the cross-entropy optimum.

(iii) CADD's sampler as a discretization. Over a reverse step from t to $t-\Delta$, the exponential clock of Algorithm 2 commits with probability $1 - \exp(-\widehat{\lambda}_t \Delta) = \vec{\lambda}(t) S(t) \Delta / (1 - S(t)) + O(\Delta^2)$; since $\dot{S} = -\vec{\lambda} S$, this matches CADD's unmasking probability $(\alpha_{t-\Delta} - \alpha_t)/(1 - \alpha_t)$ to first order under $S \leftrightarrow \alpha$. The committed token is drawn from the classifier posterior in both. While uncommitted, CADD's update $z_{t-\Delta} \sim \mathcal{N}(\tilde{\mu}_t(\hat{z}_{0,\theta}, z_t), \tilde{\beta}_t I_d)$ with $\hat{z}_{0,\theta} = \sum_a P_\theta(a \,|\, \cdot)\, E(a) = m_\theta$ is the standard DDPM ancestral step, which agrees to first order in Δ with the Euler–Maruyama step of Algorithm 2 on the reverse SDE (5) with the score above [Ho et al., 2020, Song et al., 2021]. Upon commitment both processes place the coordinate at the committed anchor and freeze it. The two samplers are therefore first-order discretizations of the same jump-diffusion, and their laws coincide as $\Delta \to 0$. CADD's alternative latent estimators, such as resampling z_t from the forward kernel centered at $\hat{z}_{0,\theta}$, are further approximations of the same reverse law, trading fidelity for mode coverage as Zheng et al. [2026] discuss. □

C.5 Learning the forward hazard, and why we fix it

Section 4.5 identifies the per-anchor forward hazard as a further design axis. Here we derive a variational objective and analyze learning the hazard jointly with the denoising classifier. The matched MD4 comparison is reported in Appendix E.2.2, and learned and fixed-hazard experiments are reported in Appendix E.2.3.

Objective. Parametrize $\vec{\lambda}_t(a) = \beta(t) w(a)$ with $w : \mathcal{A} \to \mathbb{R}_{>0}$ learned jointly with θ (the anchor-agnostic baseline is $w \equiv 1$). The plain cross-entropy (14) is not a bound on $\log p(\vec{X}_0)$, so its w-gradient does not target likelihood; we replace it with the negative ELBO. At $\eta = 1$ the unstick-time mixture collapses (22), making the DHM target (15) the closed-form scalar $\widehat{\lambda}_t(a) = \beta(t) w(a) S_a(t)/(1 - S_a(t))$ and the conditional score target w-free.

Proposition C.7 (SJD negative ELBO at $\eta = 1$)**.** *Let $S_a(t) = e^{-w(a)B(t)}$ with $B(t) = \int_0^t \beta(s)\mathrm{d}s$, $m_t(a) = p_0(a) S_a(t)$, and assume $S_a(T) = 0$ for every a together with the terminal conditions $\alpha(T) = 0$, $\sigma(T) = 1$. Then the negative ELBO decomposes as $\mathcal{L}(\theta, w) = \mathcal{L}_{\mathrm{CE}} + \mathcal{L}_{\mathrm{RB}} + \mathcal{L}_{\mathrm{score}}$, where*

$$\mathcal{L}_{\mathrm{CE}} = \int_0^T \sum_{a \in \mathcal{A}} \big(- \dot{m}_t(a) \big) \mathbb{E}_{Y \sim q_t(\cdot \,|\, a)} \big[- \log P_\theta(a \,|\, Y, t) \big] \mathrm{d}t, \tag{53}$$

$$\mathcal{L}_{\mathrm{RB}} = \int_0^T \mathbb{E}_{Y \sim p_t^{\mathrm{ac}}} \Big[\textstyle\sum_{a \in \mathcal{A}} \widehat{\lambda}_t(a) \big(P_\theta(a \,|\, Y, t) - p_t(a \,|\, Y) \big) \Big] \mathrm{d}t, \tag{54}$$

$$\mathcal{L}_{\mathrm{score}} = \int_0^T \sum_{a \in \mathcal{A}} p_0(a) \big(1 - S_a(t)\big) \mathbb{E}_{Y \sim q_t(\cdot \,|\, a)} \Big[\tfrac{g_t^2}{2} \big\| s_\theta(Y, t) + \tfrac{Y - \alpha(t) a}{\sigma^2(t)} \big\|^2 \Big] \mathrm{d}t, \tag{55}$$

$-\dot{m}_t(a) = p_0(a) \beta(t) w(a) S_a(t)$ is the forward rate of mass leaving anchor a, and $\mathbb{E}_{Y \sim p_t^{\mathrm{ac}}}$ denotes integration against p_t^{ac} on $\mathsf{X}_{\mathcal{A}}$.

Proof. Condition on $\vec{X}_0 = a$. At $\eta = 1$ the conditional law at time t is the atom $S_a(t)\delta_a$ plus the absolutely continuous part $(1 - S_a(t)) q_t(\cdot \,|\, a)$ (22). The conditional process is itself an SJD, so the flux identity of Theorem 2.6 applies to it: the conditional reverse intensity into a is the state-free scalar $S_a(t) \vec{\lambda}_t(a) q_t(y \,|\, a) \big/ \big[(1 - S_a(t)) q_t(y \,|\, a) \big] = \widehat{\lambda}_t(a)$ (the DHM target (15) is precisely the $\vec{X}_0$-conditional reverse intensity), and the conditional reverse drift uses the conditional score $-(y - \alpha(t)a)/\sigma^2(t)$. The model reverses with score s_θ and plug-in rates $\Lambda_\theta(b \,|\, y, t) = \widehat{\lambda}_t(b) P_\theta(b \,|\, y, t)$ (18); anchors are absorbing on both sides, so committed segments contribute nothing. The pathwise KL between the two reverse laws is the diffusion

quadratic term plus the jump term in Bregman form, following Campbell et al. [2023, Prop. 2], $\sum_b \Lambda_\theta(b) - \widehat{\lambda}_t(a)\log\Lambda_\theta(a) + \widehat{\lambda}_t(a)\log\widehat{\lambda}_t(a) - \widehat{\lambda}_t(a)$, integrated over the off-anchor event, plus a terminal KL that vanishes under the stated assumptions. Substituting Λ_θ, the $\log\widehat{\lambda}_t(a)$ terms cancel, leaving $\sum_b \widehat{\lambda}_t(b)P_\theta(b\,|\,Y,t) - \widehat{\lambda}_t(a)\log P_\theta(a\,|\,Y,t) - \widehat{\lambda}_t(a)$. Average over $a \sim p_0$ and the off-anchor event (weight $1 - S_a(t)$, $Y \sim q_t(\cdot\,|\,a)$). The middle term integrates to $\mathcal{L}_{\mathrm{CE}}$ via $p_0(a)(1-S_a(t))\widehat{\lambda}_t(a) = -\dot{m}_t(a)$. The first term equals $\mathbb{E}_{Y\sim p_t^{\mathrm{ac}}}[\sum_b \widehat{\lambda}_t(b)P_\theta(b\,|\,Y,t)]$ since $\sum_a p_0(a)(1-S_a(t))q_t(y\,|\,a) = p_t^{\mathrm{ac}}(y)$; the third sums to $-\sum_b(-\dot{m}_t(b))$, and the tower identity $\mathbb{E}_{Y\sim p_t^{\mathrm{ac}}}[\sum_b \widehat{\lambda}_t(b)p_t(b\,|\,Y)] = \sum_b(-\dot{m}_t(b))$ combines the two into $\mathcal{L}_{\mathrm{RB}}$. The diffusion term is $\mathcal{L}_{\mathrm{score}}$. □

$\mathcal{L}_{\mathrm{CE}}$ is the SJD analogue of the MDLM cross-entropy bound, with the absorbing point replaced by a VP-Gaussian draw and the time-only weight by the per-anchor $-\dot{m}_t(a)$. $\mathcal{L}_{\mathrm{RB}}$ is the rate-balancing part of the jump term, and its role is sharper than self-consistency: $\mathcal{L}_{\mathrm{CE}}$ alone is minimized by the allocation $\pi_t^\star(a\,|\,y) \propto \widehat{\lambda}_t(a)p_t(a\,|\,y)$ of (19) rather than by the posterior, and $\mathcal{L}_{\mathrm{RB}}$, linear in P_θ, is exactly the correction that restores the posterior as the joint minimizer. $\mathcal{L}_{\mathrm{score}}$ is conditional denoising score matching; because the induced score of Appendix D.3 is exact at the cross-entropy optimum, we train no score head and optimize $\mathcal{L}_{\mathrm{CE}} + \mathcal{L}_{\mathrm{RB}}$ alone. Every w-dependence is reparametrizable: unlike GenMD4, whose discrete forward pass forces a REINFORCE estimator [Shi et al., 2024, App. I.2], the continuous off-anchor state yields an exact ∂_w from one backward pass. In the degenerate-kernel limit of Section 4.1, the score term vanishes and the jump contribution $\mathcal{L}_{\mathrm{CE}} + \mathcal{L}_{\mathrm{RB}}$ reduces to the GenMD4 negative ELBO under $\alpha_{t,a} \leftrightarrow S_a(t)$. For an anchor-agnostic schedule, $\mathcal{L}_{\mathrm{RB}} \equiv 0$.

The objective cannot identify a useful hazard. First, the only term that couples P_θ to the choice of hazard carries no signal at the optimum:

Lemma C.8 (Rate-balance vanishes at the optimum)**.** *If $P_\theta(\cdot\,|\,y,t) = p_t(\cdot\,|\,y)$ for almost every (y,t), then $\mathcal{L}_{\mathrm{RB}}(\theta, w) = 0$ for every w.*

Proof. At the Bayes classifier the integrand of (54) is zero pointwise, for every w. □

At $w \equiv 1$ it moreover vanishes identically in θ, since $\widehat{\lambda}_t$ is anchor-agnostic. Second, the full bound is flat in w at the per-w optimum. At the Bayes classifier the plug-in rates are exact (Proposition C.5) and the induced score is exact (Appendix D.3), so the model is the exact time reversal of the forward law for *every* w: the bound is tight at the data entropy, $\mathcal{L}(\theta^\star(w), w) = \mathrm{H}(p_0)$ in the well-specified limit—the operational form of the schedule non-identifiability of Kingma et al. [2021]. Since $\theta^\star$ is also a stationary point of $\mathcal{L}$ in θ, the envelope theorem gives $\partial_w\mathcal{L}\big|_{\theta^\star} = 0$, hence

$$\partial_w\big(\mathcal{L}_{\mathrm{CE}} + \mathcal{L}_{\mathrm{RB}}\big)\big|_{\theta^\star} = -\partial_w\mathcal{L}_{\mathrm{score}}\big|_{\theta^\star} :$$

at convergence, the trained objective's hazard gradient is exactly the negative of the omitted score term's. Descending it inflates w fastest on anchors carrying the most p_0-weighted score residual (the prefactor $p_0(a)(1-S_a(t))$ grows with $w(a)$ and is largest for common anchors), i.e. a frequency-ordered schedule; away from the optimum, the residual w-signal in $\mathcal{L}_{\mathrm{RB}}$ supplies the opposite pull. Both directions materialize below, and both degrade generation when deployed.

D Implementation details

D.1 Convolution closure for the VP-matched SJD

The DHM target (15) and the plug-in quadrature (65) both require the off-anchor conditional $p_t^{\mathrm{ac}}(y\,|\,\vec{X}_0, i)$ at each position i, which is built from the convolution

$$(r_\tau * K_{\tau\to t})(y\,|\,\vec{X}_0, i) = \int_{\mathbb{R}^d} r_\tau(z\,|\,\vec{X}_0, i)K_{\tau\to t}(y\,|\,z)\mathrm{d}z,$$

of the unsticking kernel at time τ with the forward diffusion semigroup from τ to t. We work out this convolution explicitly for the VP-matched family of Example 2.2, where both factors are Gaussian and the convolution stays Gaussian. The per-anchor case is recovered at $W = I$, where $\mu_i(\vec{X}_0) = E(a^{(i)})$ depends only on the local anchor and the conditioning collapses to $p_t^{\mathrm{ac}}(y\,|\,a^{(i)})$.

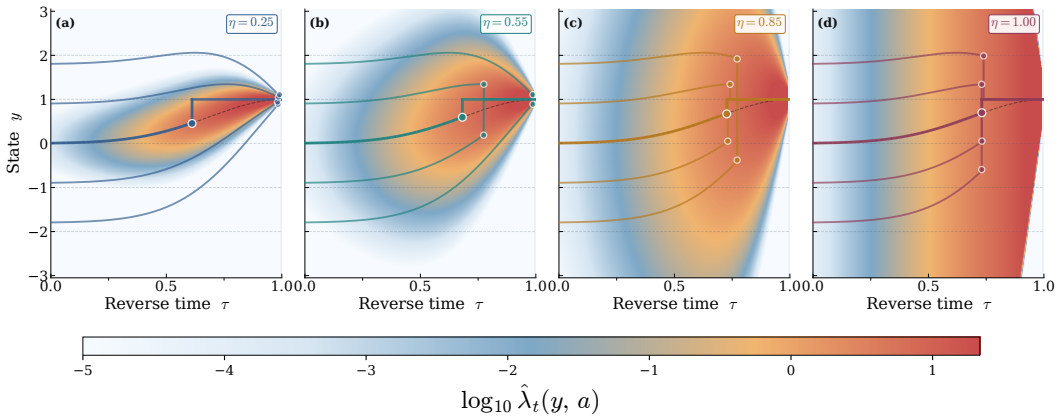

Figure 4: Effect of η on the state-dependent hazard target $\widehat{\lambda}_t(y, a)$ under a single anchor $a = 1$ for reverse trajectories (circles mark commitment events). As $\eta \to 1$, the width of the intensity surface widens. As η decreases from 1.00 (d) to 0.25 (a), the variance of the unsticking kernel sharpens relative to the mixture marginal p_t^{ac}. High intensity regions contract toward the anchor mean, and trajectories in low intensity regions tend to later commitments.

The VP semigroup. The VP-SDE $\mathrm{d}\vec{X}_t = -\frac{1}{2}\beta(t)\vec{X}_t\mathrm{d}t + \sqrt{\beta(t)}\mathrm{d}\vec{W}_t$ has the linear-Gaussian transition kernel

$$K_{\tau\to t}(\cdot \,|\, z) = \mathcal{N}\left(\cdot;\ \tfrac{\alpha(t)}{\alpha(\tau)}z,\ \left(\sigma^2(t) - \tfrac{\alpha^2(t)}{\alpha^2(\tau)}\sigma^2(\tau)\right)I_d\right),$$

where $\alpha(t) = \exp\left(-\frac{1}{2}\int_0^t \beta(s)\mathrm{d}s\right)$ and $\sigma^2(t) = 1 - \alpha^2(t)$. The VP-matched unsticking kernel from Example 2.2 is

$$r_\tau(\cdot \,|\, \vec{X}_0, i) = \mathcal{N}\left(\cdot;\ \alpha(\tau)\mu_i(\vec{X}_0),\ \eta^2\sigma^2(\tau)I_d\right), \qquad \mu_i(\vec{X}_0) = \left(WE(\vec{X}_0)\right)_i.$$

The convolution. Both factors are isotropic Gaussian, so the convolution is again isotropic Gaussian with mean and variance given by linearity. Writing $m := \mu_i(\vec{X}_0)$ for the blended mean at position i,

$$\text{mean}: \quad \tfrac{\alpha(t)}{\alpha(\tau)} \cdot \alpha(\tau)m = \alpha(t)m,$$

$$\text{variance}: \quad \left(\tfrac{\alpha(t)}{\alpha(\tau)}\right)^2 \cdot \eta^2\sigma^2(\tau) + \left(\sigma^2(t) - \tfrac{\alpha^2(t)}{\alpha^2(\tau)}\sigma^2(\tau)\right) = \sigma^2(t) - (1-\eta^2)\tfrac{\alpha^2(t)}{\alpha^2(\tau)}\sigma^2(\tau).$$

Writing $v_t(\tau) := \sigma^2(t) - (1-\eta^2)\alpha^2(t)\sigma^2(\tau)/\alpha^2(\tau)$, we recover (20):

$$(r_\tau * K_{\tau\to t})(\cdot \,|\, \vec{X}_0, i) = \mathcal{N}\big(\cdot;\ \alpha(t)\mu_i(\vec{X}_0),\ v_t(\tau)I_d\big). \tag{56}$$

The variance $v_t(\tau)$ is identical to the per-anchor case because the variance derivation depends only on the unsticking variance $\eta^2\sigma^2(\tau)$ and the VP semigroup; the cross-position structure W enters only through the mean.

Sanity checks. Three limits confirm that (56) is well-behaved.

Recovery of the diffusion kernel at $\eta = 1$. When $\eta = 1$ the unsticking kernel coincides with $q_\tau(\cdot \,|\, a) = K_{0\to\tau}(\cdot \,|\, a)$, and the semigroup property of the VP transition kernel gives $q_\tau * K_{\tau\to t} = q_t$. Consistently, $v_t(\tau)\big|_{\eta=1} = \sigma^2(t)$ for every τ, recovering the VP perturbation kernel.

Variance is positive and decreasing in τ for $\eta < 1$. For fixed t and $\eta \in (0, 1]$, $v_t(\tau)$ is strictly decreasing in $\tau \in (0, t)$, with $v_t(\tau) \to \eta^2\sigma^2(t)$ as $\tau \to t^-$ (a particle that just unstuck) and $v_t(\tau) \to \sigma^2(t)$ as $\tau \to 0^+$ (an early unstick has had the full VP semigroup to wash out the variance deficit). In particular $v_t(\tau) > 0$ throughout.

Terminal Gaussian prior. As $t \to T$ with $\alpha(T) \to 0$, every mixture component converges to $\mathcal{N}(0, I_d)$ regardless of τ and η, recovering Remark 2.5.

D.2 Efficient evaluation of the plug-in hazard

We give the implementation details behind the cost estimate of the plugin hazard: a one-dimensional quadrature for $p_t^{\mathrm{ac}}(y \mid a)$ and the resulting plug-in $\widehat{\lambda}_t(y, a)$, with a vectorized form that shares structure across anchors. We focus throughout on the VP-matched family of Example 2.2, where the convolution (56) is closed-form.

Discretizing the quadrature. Recall that

$$p_t^{\mathrm{ac}}(y \mid a) = \int_0^t \vec{\lambda}_\tau(a) S_a(\tau)(r_\tau * K_{\tau \to t})(y \mid a) \mathrm{d}\tau. \tag{57}$$

We discretize τ on a fixed grid $0 < \tau_1 < \cdots < \tau_{N_\tau} \le t$ with quadrature weights $\{w_i\}_{i=1}^{N_\tau}$ (we use Simpson's rule with $N_\tau \in [16, 64]$ in our experiments). Substituting the closed-form convolution (56),

$$p_t^{\mathrm{ac}}(y \mid a) \approx \sum_{i=1}^{N_\tau} w_i \vec{\lambda}_{\tau_i}(a) S_a(\tau_i) \mathcal{N}\big(y; \alpha(t)a, v_t(\tau_i) I_d\big). \tag{58}$$

Log-domain stability. The Gaussian densities in (58) can underflow when y is far from $\alpha(t)a$. We therefore evaluate the log-density by log-sum-exp:

$$\log p_t^{\mathrm{ac}}(y \mid a) \approx \mathrm{logsumexp}_i \Big(\log w_i + \log \vec{\lambda}_{\tau_i}(a) + \log S_a(\tau_i) + \log \mathcal{N}\big(y; \alpha(t)a, v_t(\tau_i) I_d\big) \Big). \tag{59}$$

The DHM target is then $\log \widehat{\lambda}_t(y, a) = \log \vec{\lambda}_t(a) + \log S_a(t) + \log r_a(y, t) - \log p_t^{\mathrm{ac}}(y \mid a)$, and the reweighted logits in (19) become

$$\ell_\theta(a; y, t) + \log \widehat{\lambda}_t(y, a),$$

fed directly into a softmax to sample the destination anchor.

Sharing structure across anchors. For the common case of an anchor-agnostic forward hazard $\vec{\lambda}_\tau(a) \equiv \vec{\lambda}(\tau)$, the prefactor $w_i \vec{\lambda}(\tau_i) S(\tau_i)$ in (58) is independent of a and can be precomputed once per reverse step. The only a-dependence in the integrand is in the Gaussian mean $\alpha(t)a$ (recall that the variance $v_t(\tau_i)$ depends only on τ_i and t, not on a; a consequence of VP-matching). The quadrature therefore reduces to evaluating $|\mathcal{A}| \cdot N_\tau$ Gaussian log-densities and one log-sum-exp per anchor. More generally, when the hazard factors as $\vec{\lambda}_\tau(a) = \beta(\tau) w(a)$, the τ-weights $w_i \beta(\tau_i) S(\tau_i)^{w(a)}$ can be precomputed for each anchor.

Cost. Per reverse step and per site y, evaluating the full plug-in has three components:

(i) Precomputing the τ-weights and variances: $\mathcal{O}(N_\tau)$.
(ii) Evaluating $\mathcal{N}(y; \alpha(t)a, v_t(\tau_i) I_d)$ for all (a, i): $\mathcal{O}(N_\tau |\mathcal{A}| d)$.
(iii) Log-sum-exp across τ_i and softmax across a: $\mathcal{O}(N_\tau |\mathcal{A}|)$.

The dominant term is step 2. With $N_\tau \in [16, 64]$, $|\mathcal{A}|$ on the order of a vocabulary size, and d on the order of a feature dimension, this is one to two orders of magnitude cheaper than a single forward pass through P_θ in our experiments. The quadrature is fully vectorizable and runs alongside the classifier on the same accelerator.

Cost of the blending step. For the non-local kernel of Example 2.2, computing the per-position blended mean $\mu = W E(\vec{X}_0)$ adds an $L \times L$ matrix multiply with cost $\mathcal{O}(L^2 d)$ for dense W and $\mathcal{O}(Ldw)$ for banded W of bandwidth w. Both are negligible against one classifier forward pass, and the matmul is computed once per minibatch (training) or once per reverse step (sampling).

D.3 Classifier-induced score

The reverse drift (6) requires the score $\nabla \log p_t(y)$ on $\mathsf{X}_{\mathcal{A}}$. We show how to read it off from the trained classifier P_θ and the same plug-in quadrature used for the hazard, without training a second network Dieleman et al. [2022].

Setup. The marginal density on $\mathsf{X}_{\mathcal{A}}$ factors as $p_t(y) = \sum_{a\in\mathcal{A}} p_0(a)p_t^{\mathrm{ac}}(y \mid a)$, so by the standard mixture-score identity

$$\nabla \log p_t(y) = \sum_{a\in\mathcal{A}} p_t(a \mid y)\nabla_y \log p_t^{\mathrm{ac}}(y \mid a). \tag{60}$$

The classifier P_θ approximates $p_t(a \mid y)$ from (13), so the only remaining piece is $\nabla_y \log p_t^{\mathrm{ac}}(y \mid a)$.

Score of the mixture conditional. Differentiating the integral form of $p_t^{\mathrm{ac}}(y \mid a)$ in (57) under the integral sign,

$$\nabla_y p_t^{\mathrm{ac}}(y \mid a) = \int_0^t \vec{\lambda}_\tau(a)S_a(\tau)\nabla_y(r_\tau * K_{\tau\to t})(y \mid a)\mathrm{d}\tau,$$

and dividing by $p_t^{\mathrm{ac}}(y \mid a)$,

$$\nabla_y \log p_t^{\mathrm{ac}}(y \mid a) = \int_0^t \mathrm{w}_t(\tau \mid y, a)\nabla_y \log(r_\tau * K_{\tau\to t})(y \mid a)\mathrm{d}\tau, \tag{61}$$

where the time-posterior weight is

$$\mathrm{w}_t(\tau \mid y, a) := \frac{\vec{\lambda}_\tau(a)S_a(\tau)(r_\tau * K_{\tau\to t})(y \mid a)}{p_t^{\mathrm{ac}}(y \mid a)}. \tag{62}$$

This is the conditional density of the unstick time given $(\vec{X}_0 = a, \vec{X}_t = y)$; (61) expresses the score of the mixture as the average of the component scores, weighted by the time-posterior.

VP-matched closed form. Under Example 2.2, the convolution (56) is Gaussian, so its score is

$$\nabla_y \log(r_\tau * K_{\tau\to t})(y \mid a) = -\frac{y - \alpha(t)a}{v_t(\tau)}.$$

Substituting into (61) and (60),

$$s_\theta(y, t) = -\sum_{a\in\mathcal{A}} P_\theta(a \mid y, t) \int_0^t \mathrm{w}_t(\tau \mid y, a)\frac{y - \alpha(t)a}{v_t(\tau)}\mathrm{d}\tau, \tag{63}$$

where we have substituted P_θ for the true posterior $p_t(a \mid y)$. Equation (63) is the classifier-induced score: it depends on P_θ, the precomputed τ-grid quadrature for w_t, and the affine residual $y - \alpha(t)a$.

Sharing computation with the plug-in hazard. The time-posterior weight $\mathrm{w}_t(\tau_i \mid y, a)$ is the same Gaussian-density times hazard-survival product that appears in the discretized quadrature (58), normalized by $p_t^{\mathrm{ac}}(y \mid a)$. In implementation, the score and the hazard reuse the same set of evaluations: at each (y, t) we compute $\log \mathcal{N}(y; \alpha(t)a, v_t(\tau_i)I_d)$ once for each (a, i) pair, log-sum-exp over i to obtain $\log p_t^{\mathrm{ac}}(y \mid a)$ for the hazard, and softmax over i within each a to obtain $\mathrm{w}_t(\tau_i \mid y, a)$ for the score. The marginal cost of the score on top of the hazard is one extra weighted sum per anchor: still negligible compared to a forward pass through P_θ.

Limit at $\eta = 1$. For $\eta = 1$, the convolution (56) collapses to the VP perturbation kernel $q_t(\cdot \mid a)$ for every τ, so the time-posterior $\mathrm{w}_t(\tau \mid y, a)$ becomes uniform and integrates out. Equation (63) reduces to

$$s_\theta(y, t)\big|_{\eta=1} = -\sum_{a\in\mathcal{A}} P_\theta(a \mid y, t)\frac{y - \alpha(t)a}{\sigma^2(t)},$$

which is the standard mixture-of-Gaussians score familiar from masked diffusion with continuous augmentation Zheng et al. [2026]. The general $\eta < 1$ formula (63) interpolates between this masked-diffusion-style score and the per-component VP score by averaging over the unstick time.

This $(W, \eta) = (I, 1)$ limit is exactly the score-interpolation estimator of Dieleman et al. [2022], and Eijkelboom et al. [2024] uses the same construction. SJD contributes nothing new on the drift side at this point. What SJD adds is the jump side: Theorem 2.6 fixes the commit rate and the destination by flux balance, and Theorem 3.2 together with Lemma C.4 makes both estimable from the same classifier. Neither construction has an analogue in the continuous setting, where the process has no jumps. Equation (64) below extends the estimator to $W \neq I$.

Blended kernels ($W \neq I$). For the non-local kernel of Example 2.2 the conditioning object is the whole clean sequence, and (63) does not apply as written: the mixture component at position i is centered at the blended $\alpha(t)\mu_i(\vec{X}_0)$ of (3), not at $\alpha(t)a^{(i)}$. At $\eta = 1$ the correction is exact and closed-form. Write $y = (y_1, \dots, y_L)$ for the sequence state. Conditionally on $\vec{X}_0$, the active (off-anchor) coordinates are independent with $y_i \sim \mathcal{N}\big(\alpha(t)\mu_i(\vec{X}_0), \sigma^2(t)I_d\big)$, by (56) at $v_t(\tau) = \sigma^2(t)$, and a committed position reveals its anchor exactly, since anchors are absorbing. The tower argument of (60), now conditioning on the full state (active coordinates and committed anchors), gives

$$\nabla_{y_i} \log p_t(y) = -\frac{y_i - \alpha(t)\mathbb{E}\big[\mu_i(\vec{X}_0) \,|\, y\big]}{\sigma^2(t)},$$

and because μ_i is linear in the embedded sequence (3), the conditional mean requires only the *per-position marginal* posteriors: $\mathbb{E}[\mu_i(\vec{X}_0) \,|\, y] = \sum_j W_{ij} m_j(y)$, where $m_j(y) := \sum_{a \in \mathcal{A}} p_t^{(j)}(a \,|\, y) E(a)$ is the posterior-mean embedding at position j and $p_t^{(j)}(\cdot \,|\, y)$ the marginal posterior of that position's anchor given the full state; at committed and clamped positions, $m_j = E(a^{(j)})$ exactly. Substituting the classifier yields the blended-mean induced score

$$s_{\theta,i}(y,t)\big|_{\eta=1} = -\frac{y_i - \alpha(t)\big(W m_\theta(y,t)\big)_i}{\sigma^2(t)}, \qquad m_{\theta,j}(y,t) = \sum_{a \in \mathcal{A}} P_\theta^{(j)}(a \,|\, y, t) E(a). \tag{64}$$

Three remarks. First, (64) is exact, not a mean-field approximation: the conditional score is linear in μ_i, so the joint posterior over sequences is never needed, and the single per-position classifier remains sufficient at $W \neq I$. Second, the jump side needs no modification: at $\eta = 1$ the spatial factor of the DHM target collapses to $1/(1 - S_a(t))$ for *any* W so the commit rates and allocation are unchanged, and (64) is the only place W enters the sampler: one blending product $W m_\theta$ per reverse step, the same product as in training (Appendix D.2). Third, at $W = I$ it reduces to the $\eta = 1$ limit above. We use (64) whenever $W \neq I$; for $\eta < 1$ the time-posterior (62) depends on $\mu_i(\vec{X}_0)$ inside the conditional expectation, per-position marginals no longer suffice, and we do not combine $W \neq I$ with $\eta < 1$.

D.4 Sampling

A trained classifier P_θ from Section 3 fully specifies an approximation to the reverse-time SJD of Theorem 2.6: the score and the per-anchor reverse intensities are both available through the plug-in (18). We assemble these into a sampler that runs from the prior p_T at $\tau = 0$ to data at $\tau = T$.

The classifier also receives no explicit commitment indicator. Its input is the state alone. A committed coordinate is clamped exactly at its anchor, and a diffusing coordinate hits an anchor with probability zero, so the state determines the commitment pattern almost surely. At $(W, \eta) = (I, 1)$ the state is a measurable bijection of the explicit pair (x_t, z_t) used by hybrid models with a separate discrete channel (Proposition 4.1(i)), so no information is lost by dropping that channel.

Predictor: one reverse step. At reverse time τ (forward time $t = T - \tau$) and current state $y \in \mathsf{X}_{\mathcal{A}}$, one Euler step of size $\Delta\tau$ does two things in parallel:

(i) *Drift and diffuse along the score.* Compute the classifier-induced score $s_\theta(y,t)$ (Appendix D.3) and advance the continuous SDE component via a standard Euler–Maruyama step.

(ii) *Sample a sticky jump.* With probability $1 - \exp\big(-\lambda_{\mathrm{plug},\theta}(y,t)\Delta\tau\big)$, commit to an anchor drawn from $\pi_{\mathrm{plug},\theta}(\cdot \,|\, y, t)$ (19) and treat that anchor as absorbing for the rest of the reverse trajectory.

Both steps reuse the same forward pass through P_θ: the classifier produces logits over anchors, from which the score and the reweighted jump distribution are read off. Algorithm 2 states the full sampler.

Evaluating the plug-in hazard. The plug-in (18) requires the mixture density $p_t^{\mathrm{ac}}(y \,|\, a)$ pointwise, evaluated by

$$p_t^{\mathrm{ac}}(y \,|\, a) = \int_0^t \vec{\lambda}_\tau(a) S_a(\tau) (r_\tau * K_{\tau \to t})(y \,|\, a) \mathrm{d}\tau, \tag{65}$$

which is one-dimensional and admits a closed-form integrand whenever the simulation-free conditions of Section 3.5 hold. We discretize (65) on a τ-grid of N_τ points and evaluate the log of the integrand by log-sum-exp for numerical stability. The cost is negligible compared to a forward pass through P_θ. See Appendix D.2 for details.

E Experimental details

E.1 CIFAR-10

Value-space blending kernel. The blended arms of Figure 2 replace the linear embedding blend (3) with a blend applied to raw pixel values before embedding. Flatten the 32×32 pixel grid row-major into $N = 1024$ sites, indexed by $i \mapsto (r_i, c_i)$ with $r_i = \lfloor i/32 \rfloor$ and $c_i = i \bmod 32$, and write the squared grid distance

$$d_{ij}^2 = (r_i - r_j)^2 + (c_i - c_j)^2. \tag{66}$$

The blending matrix $B^{(\sigma_B)} \in \mathbb{R}^{N \times N}$ is the row-stochastic Gaussian position kernel

$$B_{ij}^{(\sigma_B)} = \frac{\exp\big(-d_{ij}^2/2\sigma_B^2\big)}{\sum_{k=1}^{N} \exp\big(-d_{ik}^2/2\sigma_B^2\big)}, \qquad \sigma_B \in \{1.5, 2.0, 2.5\}, \tag{67}$$

equivalently the row-wise softmax of $-D/2\sigma_B^2$ with $D = [d_{ij}^2]$. Row-stochasticity keeps blended values on the pixel scale, and $B^{(\sigma_B)} \to I$ as $\sigma_B \to 0$, recovering the identity-kernel hybrid. With $v(\vec{X}_0) \in [0, 255]^{N \times 3}$ the raw pixel values and $\mathcal{E}$ the fixed sinusoidal embedding of the value scale,

$$\mathcal{E}(v) = \Big[\sin\big(2\pi f v/P\big), \cos\big(2\pi f v/P\big)\Big]_{f=1}^{F}, \qquad F = 4, \quad P = 255, \quad \|\mathcal{E}(v)\|_2 = \sqrt{F} = 2, \tag{68}$$

the forward mean blurs the raw values and embeds the blurred result,

$$\mu(\vec{X}_0) = \mathcal{E}\big(B^{(\sigma_B)} v(\vec{X}_0)\big), \tag{69}$$

with $B^{(\sigma_B)}$ acting over the flattened spatial grid independently for each of the $C = 3$ colour channels; the token sequence has length $L = 3N = 3072$, one `uint8` token per channel per site. The unsticking kernel (4) uses this mean at $\eta = 1$. The composition $\mathcal{E} \circ B^{(\sigma_B)}$ is smooth in the pixel values, which is what makes natural images low-pass under this corruption; the same blur applied to arbitrary learned embeddings does not preserve that structure.

At sampling, the induced score and plug-in weights require the forward mean under the denoising posterior. We form the per-site posterior-mean value $\hat{v}_i = \sum_{v=0}^{255} P_\theta(v \mid \cdot) v$, blur it through $B^{(\sigma_B)}$, and embed the blended result, $\mathcal{E}\big(B^{(\sigma_B)} \hat{v}\big)$; the embedding step is the mean-field substitution of Appendix D.3.

Setup and common hyperparameters. All CIFAR-10 experiments used unconditional generation on images represented as discrete `uint8` tokens of shape $32 \times 32 \times 3$ with vocabulary size 256 [Shi et al., 2024, Zheng et al., 2026]. We used no data augmentation. Unless otherwise noted, training and validation used batch size 512, AdamW with learning rate 10^{-4}, $\beta_2 = 0.999$, weight decay 0, and a 1,000-step warmup followed by a constant learning-rate schedule. All runs used seed 0, were trained for 200,000 steps, used EMA decay 0.9999, and saved checkpoints every 10,000 steps; the best checkpoint was selected by validation FID. We report FID-50k against CIFAR-10 train statistics, using FID evaluation batch size 128. All samplers used 256 NFEs.

All CIFAR-10 models used ADM-family U-Net backbones [Dhariwal and Nichol, 2021]. SJD, MD4, MDLM, and Bit Diffusion used the 5D image-token ADM variant, while DDPM used the 2D ADM variant. Shared ADM hyperparameters were feature dimension 96, 32 layers, dropout 0.1, channel multipliers $[3, 4, 4]$, 4 residual blocks per level, and attention at resolutions $[2, 4]$. We train using two NVIDIA H100 GPUs. Owing to compute constraints, each CIFAR-10 configuration is trained with a single seed, following the single-run reporting practice of prior diffusion evaluations on this benchmark [Ho et al., 2020, Chen et al., 2023, Shi et al., 2024, Zheng et al., 2026], so FID differences on the order of run-to-run variability should be read with caution.

Model-specific settings. SJD used a linear VP schedule with $\beta_{\min} = 0.1$, $\beta_{\max} = 20.0$, and $T = 1$, a polynomial-α hazard with $p = 1$. MD4 used the continuous-time masked discrete diffusion setup with a linear noise schedule, antithetic time sampling, ancestral sampling, a cosine grid, and top-$p = 0.98$. DDPM used a linear β schedule from 10^{-4} to 2×10^{-2} and ϵ-prediction. MDLM used a continuous-time masked discrete model with a linear noise schedule, ancestral sampling, a cosine grid, and no prediction caching. Bit Diffusion used x_0-prediction, an MSE loss, a linear signal schedule, and deterministic DDIM-style sampling on a uniform grid. In our experiments Bit Diffusion is not self-conditioned.

E.2 Text8

We follow the setup of Lou et al. [2024], Austin et al. [2021] on 256-character sequences with an 8M-parameter DiT denoiser. The blending matrix W is a 1D Gaussian convolution of bandwidth σ_W, swept over $\sigma_W \in \{0.5, 1.0, 1.5, 2.0, 2.5\}$; $W = I$ ($\sigma_W = 0$) recovers the CANDI baseline.

Noise schedule. All methods use the log-linear schedule for the discrete masking corruption [Sahoo et al., 2024, Pynadath et al., 2026]. CANDI and SJD additionally use the CANDI rank-degradation schedule with $r_{\min} = 0.05$ and $r_{\max} = 0.4$.

Training. Table 3 lists the optimizer and batch settings, shared across all methods.

Table 3: Text8 training hyperparameters.

Hyperparameter	Value
Optimizer	
Algorithm	AdamW
Learning rate	1×10^{-3}
β_1	0.9
β_2	0.999
ϵ	1×10^{-8}
Weight decay	0
Gradient clip	1.0

Hyperparameter	Value
Batch size	
Global batch size	512
Gradient accumulation steps	4
Training	
Max steps	40,000
Precision	`bf16`
EMA decay	0.9999
Antithetic time sampling	✓
LR scheduler	Constant
Hardware	1 RTX A6000

Inference and quality–diversity frontier. MDLM uses ancestral sampling. CANDI and SJD use the exact hybrid sampler of Pynadath et al. [2026]. For each method and each NFE budget we sweep the sampling temperature over 11 evenly spaced values in $[0.05, 1.0]$ to trace the quality–diversity frontier (Figure 5); Figure 3 reports the maximum valid-word count along each frontier. SJD at $\sigma_W \geq 1.0$ Pareto-dominates CANDI and MDLM throughout the high-diversity regime, with the dominance widening at higher NFE.

E.2.1 Intermediate commitment versus terminal decoding

We compare SJD's intermediate commitment with terminal nearest-anchor decoding on Text8. In the terminal-decoding baseline, each final continuous state y is mapped to a token by

$$\hat{a}(y) = \arg\min_{a \in \mathcal{A}} \big\| y - E(a) \big\|_2.$$

The comparison matches the classifier, token embeddings, and training budget. We sample the continuous baseline using both the reverse SDE and the probability-flow ODE (PF-ODE). Whereas SJD commits to anchors during generation, the continuous baselines defer all discrete decisions to the terminal rounding step. Table 4 reports valid-word fractions at matched numbers of function evaluations (NFE).

At the five-character threshold, SJD attains 1.81–2.31 times the valid-word fraction of the continuous baselines across the tested budgets. At 128 NFE, its relative advantage increases at the six-character threshold, reaching factors of 2.36 over the reverse SDE and 3.33 over the PF-ODE. Moreover, SJD

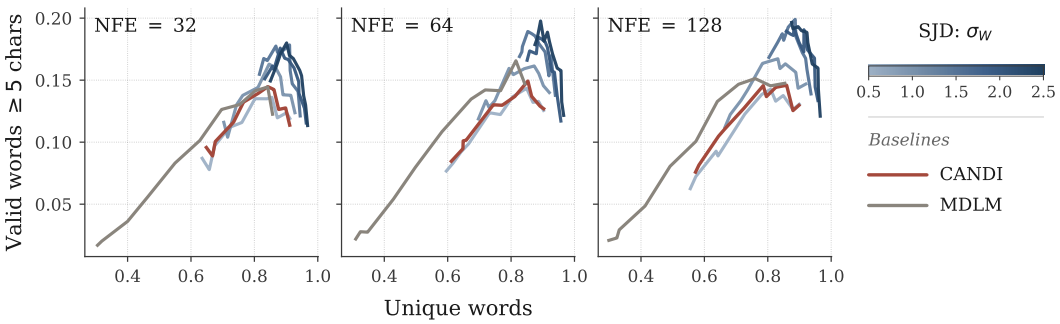

Figure 5: Quality–diversity frontiers on Text8, by NFE budget. Each curve is a temperature sweep in the (unique-word fraction, valid-word fraction at length ≥ 5) plane; upper-right dominates. The SJD family at $\sigma_W \geq 1.0$ Pareto-dominates CANDI and MDLM at high diversity.

Table 4: Intermediate commitment versus terminal decoding on Text8. Higher valid-word fractions are better. The classifier, embeddings, and training budget are matched; the continuous baselines use terminal nearest-anchor rounding.

	Word length ≥ 5			Word length ≥ 6
Sampler	32 NFE	64 NFE	128 NFE	128 NFE
Continuous reverse SDE	0.081	0.102	0.117	0.055
Continuous PF-ODE	0.078	0.088	0.092	0.039
SJD	**0.180**	**0.193**	**0.212**	**0.130**

at 32 NFE achieves 0.180, exceeding both continuous baselines at 128 NFE (0.117 and 0.092). Thus, in this comparison, intermediate commitment yields higher measured validity with one quarter of the function evaluations.

E.2.2 Likelihood-bound comparison with MD4

We compare SJD with MD4 [Shi et al., 2024] under the Text8 protocol of Appendix E.2, using an 8M-parameter DiT, a matched discrete noise schedule and training budget, and three training seeds per method. Table 5 reports likelihood bounds alongside valid-word generation at 128 NFE.

Training objective and evaluated bound. Under the assumptions of Proposition C.7, the SJD negative ELBO in Appendix C.5 decomposes as

$$\mathcal{L} = \mathcal{L}_{\mathrm{CE}} + \mathcal{L}_{\mathrm{RB}} + \mathcal{L}_{\mathrm{score}}.$$

Here, $\mathcal{L}_{\mathrm{CE}}$ is cross-entropy weighted by the per-anchor mass-release rate, $\mathcal{L}_{\mathrm{RB}}$ is a rate-balancing correction that vanishes at the Bayes classifier, and $\mathcal{L}_{\mathrm{score}}$ is the conditional score-matching contribution. We train SJD by minimizing $\mathcal{L}_{\mathrm{CE}} + \mathcal{L}_{\mathrm{RB}}$ and obtain its score from the classifier as in Appendix D.3; MD4 is trained on its own negative ELBO. The induced score is exact at the Bayes classifier, but the score contribution is retained when evaluating the full bound. We denote this SJD training variant by SJD-ELBO. In the degenerate-kernel limit, the score term disappears and the jump contribution $\mathcal{L}_{\mathrm{CE}} + \mathcal{L}_{\mathrm{RB}}$ reduces to the GenMD4 negative ELBO. For an anchor-agnostic schedule, $\mathcal{L}_{\mathrm{RB}} \equiv 0$.

Table 5: Text8 likelihood-bound comparison with MD4, using three training seeds per method. BPC denotes bits per character.

Method	Full bound (BPC ↓)	Discrete term (BPC ↓)	Valid-word fraction (length ≥ 5, 128 NFE ↑)
MD4	1.968 ± 0.005	1.968 ± 0.005	0.207 ± 0.004
SJD-ELBO	2.122 ± 0.010	1.291 ± 0.003	0.211 ± 0.005

Table 6: End-to-end hazard learning on Text8 ($W = I$, $\eta = 1$): valid-word fraction at length ≥ 5, maximized over the temperature/NFE sweep of Appendix E.2; best checkpoint within 150k steps (deployed learned-hazard runs peak at 25k). "Deployed" samples with the trained hazard; "$w \equiv 1$ sampler" overrides it to uniform at sampling only. Spread is $\max_a w(a)/\min_a w(a)$ at convergence; parentheses give the rank correlation between $\log w(a)$ and character frequency.

Configuration	w schedule	spread	deployed	$w \equiv 1$ sampler
Fixed hazard, plain CE (14)	uniform	$1\times$	0.27	0.27
Fixed hazard, ELBO-trained θ	uniform	$1\times$	0.13	0.13
Bounded (cross-entropy–driven)	frequency ($+0.99$)	$4\times$	0.066	0.10
Uniform-CE ($\mathcal{L}_{\mathrm{RB}}$-driven)	anti-freq. (-0.7)	$403\times$	0.014	0.14

Interpretation. The SJD discrete term is 0.677 BPC below MD4's negative ELBO, but its denoiser also observes a noisy continuous representation of the clean token. This supplies information unavailable at masked positions in MD4, so the discrete term alone does not establish a likelihood improvement. The continuous contribution adds 0.831 BPC, giving a full bound 0.154 BPC above MD4's. These full bounds can be reported together, but their ordering does not establish the ordering of the exact model likelihoods because their variational gaps may differ. CADD similarly cautions against directly comparing bound tightness across models and reports its discrete component for reference [Zheng et al., 2026, App. C.2]. CCDD removes information about masked tokens from the continuous input for its main-text validation-perplexity evaluation [Zhou et al., 2026, App. D.2]. Finally, SJD-ELBO and MD4 attain similar valid-word fractions at 128 NFE, so the likelihood-bound comparison is accompanied by comparable performance on this generation metric.

E.2.3 Forward-hazard ablations

Learned schedules degrade generation. We test on Text8 at $W = I$, $\eta = 1$, training for 150k steps and reporting the valid-word fraction at length ≥ 5, maximized over the temperature/NFE sweep of Appendix E.2. A learned hazard can hurt through two routes: at training, the weight $-\dot{m}_t(a)$ skews which (a, t) pairs P_θ sees; at inference, $\widehat{\lambda}_t(a)$ reweights the allocation (19). An offline read-out that overrides $w \equiv 1$ in the sampler only separates them. In Table 6, the *bounded* variant ($w \in [0.5, 2]$, cross-entropy driven) learns the frequency-ordered schedule predicted by the envelope identity; the *uniform-CE* variant ($w \equiv 1$ inside the weight, so $\partial_w \mathcal{L}_{\mathrm{CE}} = 0$ and the residual signal is $\partial_w \mathcal{L}_{\mathrm{RB}}$) runs anti-frequency to its clamp $[0.05, 20]$. Both classifiers stay healthy, forcing $w \equiv 1$ at sampling recovers 0.10–0.14, at the fixed-hazard ELBO reference of 0.13, yet the deployed hazard degrades generation monotonically in spread: $1\times \to 0.13$, $4\times \to 0.066$, $403\times \to 0.014$, against 0.27 for the plain-CE baseline within the same budget (0.28 at 300k). Note that the ELBO weighting alone already costs half the metric: at $w \equiv 1$, $\mathcal{L}_{\mathrm{RB}} \equiv 0$ and the objective reduces to $-\dot{m}_t$-weighted cross-entropy, dropping $0.27 \to 0.13$.

Fixed non-uniform hazards are nearly neutral. The learned runs confound three factors: the objective's w-dynamics, the damage the w-dependent CE weight does to θ, and, in the $w \equiv 1$ read-outs, a train/inference hazard mismatch. To probe the axis itself we remove all three: we hardcode $w(a) \propto \mathrm{freq}(a)^\gamma$, normalized so $\sum_a p_0(a) w(a) = 1$, freeze them, and train by plain cross-entropy (14), so the hazard enters only the corruption and the inference reweighting and the classifier is trained and sampled under the same hazard.

The exponent γ sets spreads of $2\times$ and $4\times$ in each direction: frequency-keyed ($\gamma > 0$; common characters commit *last* in reverse time) and anti-frequency ($\gamma < 0$; common characters commit *first*). Every arm runs three seeds under the offline protocol of Appendix E.2; as a plumbing check, $\gamma = 0$ through this path reproduces the historical fixed-hazard baseline. Table 7 is unambiguous: every schedule lands within 0.02 of uniform in the mean, and the same $4\times$ frequency direction that collapsed to 0.066 under the ELBO recipe reaches 0.279 here. The failure of end-to-end hazard learning is therefore a property of the objective, not of the axis: the bound is flat in w at the optimum, the CE weight—the only strong w-signal—corrupts the classifier it trains, and either runaway direction degrades generation through the reverse-sampler allocation.

Table 7: Frozen frequency-keyed hazards on Text8 (plain cross-entropy (14), $W = I$, $\eta = 1$): valid-word fraction at length ≥ 5 under the offline protocol of Appendix E.2, mean$\pm$std over three seeds. Anti-frequency schedules commit common characters first in reverse time; frequency-keyed schedules commit them last. No non-uniform schedule clears uniform under the pre-registered mean$-$std criterion at either budget.

Hazard	spread	valid-word (len ≥ 5)
150*k steps (three seeds)*		
anti-frequency	$4\times$	0.262 ± 0.004
anti-frequency	$2\times$	0.272 ± 0.015
uniform ($w \equiv 1$)	$1\times$	0.263 ± 0.000
frequency	$2\times$	0.279 ± 0.013
frequency	$4\times$	0.279 ± 0.008
300*k steps (three seeds)*		
anti-frequency	$2\times$	0.270 ± 0.035
uniform ($w \equiv 1$)	$1\times$	0.273 ± 0.005

E.3 Sudoku

We use the Sudoku dataset and split from Shah et al. [2024]. Each instance consists of a 9×9 puzzle (with blanks) and its unique completed solution. The dataset is filtered to puzzles solvable without backtracking using a fixed set of human-style strategies, ensuring that test-time evaluation probes logical deduction rather than combinatorial search. The model receives an input such as

Table 8: Sudoku dataset statistics

Statistic	Value
Grid size	9×9
Vocabulary	digits $\{0, \ldots, 9\}$ ($\lvert\mathcal{V}\rvert = 10$)
Train puzzles	1,800,000
Test puzzles	100,000
Filtering	solvable by fixed strategy set
URL	source

(080050060 . . . 603100007) and predicts the full solution (789251364 . . . 653184297). This matches the MGDM-style Sudoku formulation [Ye et al., 2025] in which the puzzle is represented as an 81-token discrete sequence with (0) used as the masking symbol.

Constraint-graph jump kernel. The Sudoku board is the $L = 81$ token positions of a 9×9 grid. Index a position $i \in \{0, \ldots, 80\}$ by its row, column, and 3×3 box,

$$r_i = \lfloor i/9 \rfloor , \qquad c_i = i \bmod 9, \qquad b_i = 3 \lfloor r_i/3 \rfloor + \lfloor c_i/3 \rfloor . \tag{70}$$

Two positions are Sudoku-constraint neighbours iff they share a unit (row, column, or box):

$$\mathcal{N}(i) = \{i\} \cup \underbrace{\{j : r_j = r_i\}}_{\text{row}} \cup \underbrace{\{j : c_j = c_i\}}_{\text{column}} \cup \underbrace{\{j : b_j = b_i,\ r_j \neq r_i,\ c_j \neq c_i\}}_{\text{box}} . \tag{71}$$

The fixed blending matrix $W \in \mathbb{R}^{81 \times 81}$ is the constraint-graph Gaussian with bandwidth $\sigma_W = 1.5$,

$$W_{ij} = \begin{cases} \exp\left(-\dfrac{(r_i - r_j)^2 + (c_i - c_j)^2}{2\sigma_W^2} \right), & j \in \mathcal{N}(i), \\ 0, & \text{otherwise.} \end{cases} \tag{72}$$

The blended mean (3) of each cell is thus its own embedding plus a Gaussian-weighted sum of its 20 constraint peers,

$$\mu_i(\vec{X}_0) = \big(W E(\vec{X}_0)\big)_i = E\big(a^{(i)}\big) + \sum_{j \in \mathcal{N}(i) \setminus \{i\}} \exp\left(-\frac{(r_i - r_j)^2 + (c_i - c_j)^2}{2\sigma_W^2} \right) E\big(a^{(j)}\big), \tag{73}$$

and the unsticking kernel (4) uses this mean at $\eta = 1$, $r_t(y \mid i, \vec{X}_0) = \mathcal{N}\big(y; \alpha(t)\mu_i(\vec{X}_0), \sigma^2(t) I_d\big)$.

NeurIPS Paper Checklist

(i) **Claims**

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

Answer: [Yes]

Justification: The abstract and Section 1 enumerate four contributions—the SJD process class (Section 2), reverse dynamics from a single classifier via Denoising Hazard Matching (Section 3), the unifying view of discrete, continuous, and hybrid diffusion (Section 4), and empirical evaluation on CIFAR-10, ImageNet 64×64, Text8, and Sudoku (Section 5)—and each contribution is supported by a corresponding section in the paper.

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.

(ii) **Limitations**

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

Answer: [Yes]

Justification: Section 6 ("Conclusion and Limitations") notes that state-dependent forward hazards $\vec{\lambda}_t(a)$ in the spirit of GenMD4 [Shi et al., 2024] are an unexplored axis of the framework. The empirical study is restricted to the VP-matched family of Example 2.2 with anchor-agnostic hazards and is conducted at small to moderate scale; we do not evaluate at language-model scale. The simulation-free training conditions of Section 3.5 restrict the forward families that admit our exact training pipeline; non-Gaussian un-sticking kernels would require a quadrature step in training.

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.

(iii) **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: [Yes]

Justification: The two main theoretical results are Theorem 2.6 (time reversal for jump-diffusions) and Theorem 3.2 (Denoising Hazard Matching). Theorem 2.6 is stated in the main text under regularity assumptions formalized as Assumption C.1 and proved in Appendix C; Theorem 3.2 is stated in the main text and proved in Appendix C.2. The Rao–Blackwellized plug-in is justified by Lemma C.4 and Propositions C.5 and C.6 with proofs in Appendix C.3. Auxiliary closed-form identities (e.g., the convolution closure of (20)) are derived in Appendix D.1. All theorems and lemmas are numbered and cross-referenced.

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.

(iv) **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: Appendix E specifies the architectures, optimizers, schedules, batch sizes, training-step counts, EMA decay, sampler grids, NFE budgets, and dataset splits for each of the four experimental settings (Appendix E.1 for CIFAR-10 and ImageNet, Appendix E.2 with Table 3 for Text8, and Appendix E.3 with Table 8 for Sudoku). Algorithms 1 and 2 spell out the simulation-free training loop and the predictor sampler. Appendices D.1 to D.4 give the closed-form quadratures and implementation details required to reproduce the plug-in hazard, classifier-induced score, and reverse sampler.

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

(i) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
(ii) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(iii) 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).
(iv) 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.

(v) **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: [Yes]

Justification: Code is released at the (anonymized) repository linked in the contributions block of Section 1. All four datasets are public: CIFAR-10, ImageNet 64×64, Text8, and the Sudoku split of Shah et al. [2024] (URL given in Table 8). Training and sampling scripts cover both the SJD method and the baselines reported in Tables 1 and 2.

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.

(vi) **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: Appendix E reports all training and test details. Appendix E.1 lists optimizer (AdamW with $\beta_2 = 0.999$, weight decay 0, learning rate 10^{-4}, 1,000-step warmup), batch size, EMA decay, NFE, FID protocol, and the full ADM-family backbone hyperparameters for CIFAR-10 and ImageNet. Appendix E.2 with Table 3 lists the Text8 optimizer and training settings; Appendix E.3 with Table 8 lists the Sudoku dataset statistics and the GPT-2-style denoiser configuration. Method-specific schedules and samplers (linear VP, log-linear masking, CANDI rank-degradation with $r_{\min} = 0.05$, $r_{\max} = 0.4$, etc.) are reported per method.

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.

(vii) **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: [No]

Justification: All runs use seed 0 (see Appendix E.1) and we report single-run FID, valid-word counts, and Sudoku board accuracy without error bars or confidence intervals. We did not have the compute budget to run multi-seed replicas across all four benchmarks.

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.

(viii) **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: Appendix E.1 reports two NVIDIA H100 GPUs for the CIFAR-10 and ImageNet experiments at 200,000 training steps (CIFAR-10, baselines) and 100,000 steps (ImageNet, SJD only); Appendix E.2 (Table 3) reports one NVIDIA RTX A6000 for the Text8 experiments at 40,000 steps; Appendix E.3 (Table 8) reports one NVIDIA A100 for the Sudoku experiments. Batch size, gradient-accumulation steps, and precision (`bf16` for Text8) are all stated.

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).

(ix) **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: We have reviewed the NeurIPS Code of Ethics and confirm that the research conforms to it. The paper uses only public benchmark datasets (CIFAR-10, ImageNet 64×64, Text8, Sudoku from Shah et al. [2024]) under their respective licenses, involves no human subjects, and preserves anonymity in the submitted version (e.g., the code repository link is anonymized).

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).

(x) **Broader impacts**

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

Answer: [N/A]

Justification: The paper is foundational methodological work: it unifies existing model classes (masked, continuous, and hybrid diffusion) inside a single jump-diffusion framework and proposes a training rule (DHM). It does not introduce new generative capabilities beyond the well-studied baselines it compares against, does not release pretrained models targeted at downstream deployment, and does not evaluate at scales where direct misuse pathways (large-scale text or image generation) become more salient than for the prior work it generalizes.

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).

(xi) **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 release pretrained language models, image generators, or scraped datasets that pose a high risk for misuse. The released code reproduces research models on standard public benchmarks and does not include checkpoints whose capabilities exceed those of the existing baselines we compare against.

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.

(xii) **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 datasets and baseline architectures used in the paper are cited at the point of first use: CIFAR-10 and ImageNet 64×64 (cited via the protocols of Shi et al. [2024] and Zheng et al. [2026]), Text8 (cited via Lou et al. [2024], Austin et al. [2021]), and the Sudoku split of Shah et al. [2024]. Baseline methods are cited at every comparison: MD4 [Shi et al., 2024], MDLM [Sahoo et al., 2024], Bit Diffusion [Chen et al., 2023], DDPM [Ho et al., 2020], CANDI [Pynadath et al., 2026], and the position-selector of Kim et al. [2025]. The ADM backbone [Dhariwal and Nichol, 2021] is cited where used. The released code respects the licenses of all third-party components and uses these assets only for their intended research purpose.

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.

(xiii) **New assets**

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

Answer: [Yes]

Justification: The new asset introduced by the paper is the SJD codebase, released at the anonymized repository linked in the contributions block of Section 1. The repository contains documentation covering training and sampling scripts for the four reported settings, the analytic plug-in quadrature and classifier-induced score implementations described in Appendices D.2 and D.3, and the configuration files needed to reproduce Tables 1 and 2 and Figures 3 and 5.

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.

(xiv) **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: The paper does not involve crowdsourcing or research with human subjects.

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.

(xv) **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: The paper does not involve human subjects, so IRB approval is not applicable.

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.

(xvi) **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: LLMs are not used as an important, original, or non-standard component of the core methods.

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.