Expanding Flow Maps

Sophia Tang
Department of
Computer and Information Science
University of Pennsylvania
sophtang@engineering.upenn.edu

Pranam Chatterjee
Department of
Computer and Information Science, Bioengineering
University of Pennsylvania
pranam@engineering.upenn.edu

Abstract

Flow-based generative models have enabled remarkable progress in fast and controllable generation across continuous and discrete state spaces, yet existing parameterizations are constrained to fixed dimensions or fixed sequence lengths. Here, we introduce **Expanding Generative Flows** (EFlows), which define flows between distributions of *increasing dimensionality* along an expanding interpolant that grows the state by augmenting it with conditional noise. Building on this construction, we propose **Expanding Flow Maps** (EFMs), a new class of flow maps that distill the expanding interpolant into efficient few-step generative models. Each EFM factors the map between any two timesteps into two learnable operations: an *expand operator*, which augments the state space with new coordinates or tokens conditioned on the current state, and a *transport map*, which pushes the expanded state forward along the interpolant. Composing these operators yields a single map that jointly *expands* and *denoises* the state, recovering existing fixed-canvas flows and flow maps as the special case in which the expand operator is the identity. We further extend the framework to the discrete simplex, enabling variable-size graph generation and variable-length sequence generation. Across both continuous and discrete modalities, we establish EFlows and EFMs as a principled framework for settings in which output size is itself a learned, controllable degree of freedom.

1 Introduction

Flow maps have recently emerged as a unifying framework for few-step generative modeling, supporting both continuous (Boffi et al., 2025b) and discrete (Roos et al., 2026; Lee et al., 2026; Potaptchik et al., 2026b) state spaces. Building on the success of score (Song et al., 2021; Ho et al., 2020) and flow-based (Lipman et al., 2023; Tong et al., 2024; Albergo et al., 2025) models, flow maps cast generation as a continuous-time interpolant between noise and data and learn to parameterize discrete jumps along the ODE trajectory between any pair of timesteps (Boffi et al., 2025b; Geng et al., 2025; Song et al., 2023; Kim et al., 2024). They are typically trained either by *distillation from a teacher flow* (Boffi et al., 2025a; Frans et al., 2025; Zhou et al., 2025; Salimans et al., 2024) or by *self-distillation from data* (Boffi et al., 2025b), enabling sampling in as few as one or two function evaluations while retaining the modeling flexibility of their multi-step counterparts.

Despite this progress, existing flow maps are fundamentally restricted to generation on a *fixed canvas*: a continuous state space of fixed dimensionality d, or a discrete state space of fixed sequence length L. This rigid constraint conflicts with the structure of many real-world generative tasks, where the size of the output is itself a quantity to be modeled – for example, variable-resolution images and 3D shapes, audio and video of arbitrary duration, language with unknown sequence length, and multi-modal data that interleaves modalities of differing intrinsic dimensionality. These settings require a generative framework whose state space can *grow* during inference, rather than being committed to at initialization. This raises the central question motivating this work:

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

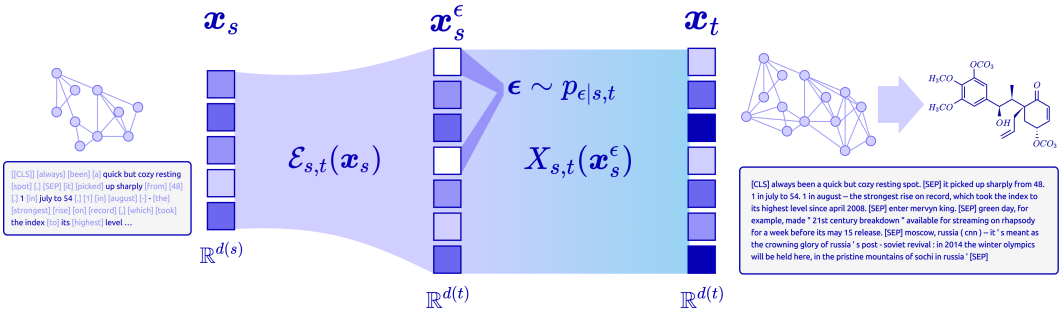

Figure 1: **Expanding Flow Maps.** Step from lower-dimensional space on $\mathbb{R}^{d(s)}$ at time s to a higher dimensional space $\mathbb{R}^{d(t)}$ at time t by **(i)** applying the expand operator $\mathcal{E}_{s,t}$ with augmented noise $\boldsymbol{\epsilon} \sim p_{\epsilon|s,t}$ and **(ii)** applying the transport map $X_{s,t}$ in the augmented space.

How can we enable few-step generation of both continuous and discrete data with adaptive dimensionality at inference?

We address this question by introducing **Expanding Flow Maps** (EFMs), a general framework for few-step generative modeling over both continuous and discrete data with variable dimensionality and sequence length. The core idea is to factor the flow map between any two timesteps into two learnable operations: an *expand operator*, which expands the state space by inserting new coordinates or tokens, and a *transport map*, which moves the expanded state forward along the interpolant. Composing these operators yields a single map that jointly *expands* and *denoises* the state, positioning standard fixed-dimensional flow maps as the special case in which the expand operator is the identity.

Our **main contributions** can be summarized in the following points:

1. **Expanding Generative Flows:** We define flows on an *expanding interpolant* between distributions of increasing dimensionality, and characterize them as a piecewise-deterministic Markov process that combines smooth transport with jumps to higher dimensions.
2. **Expanding Flow Maps:** We derive a map between arbitrary points on this interpolant by composing an *expand operator* with a *transport map*.
3. **Discrete Expanding Flows and Maps:** We extend both to the discrete simplex, where expansion becomes insertion along a sequence or graph.

Through experiments on molecular conformer generation, discrete molecular graph generation, and language modeling, we establish EFlow and EFMs as a *single, unified recipe* for variable-dimensional generation across domains and at any sampling budget. Further discussion of related works is provided in App A.

2 Preliminaries

Generative Flows and Stochastic Interpolants The *stochastic interpolants* framework (Albergo et al., 2025; Albergo and Vanden-Eijnden, 2023) constructs a continuous-time path between a prior p_0 and the data distribution p_1. Given endpoints $\boldsymbol{x}_0 \sim p_0$ and $\boldsymbol{x}_1 \sim p_1$, it defines $\boldsymbol{x}_t = I_t(\boldsymbol{x}_0, \boldsymbol{x}_1)$, typically the linear interpolant $(1-t)\boldsymbol{x}_0 + t\boldsymbol{x}_1$. The velocity field $b_t(\boldsymbol{x}) = \mathbb{E}[\dot{\boldsymbol{x}}_t | \boldsymbol{x}_t = \boldsymbol{x}]$ transports samples from p_0 to p_1 along the ODE $d\boldsymbol{x}_t = b_t(\boldsymbol{x}_t)dt$. We fit it by minimizing the *conditional flow-matching objective* (Lipman et al., 2023; Albergo and Vanden-Eijnden, 2023):

$$\mathcal{L}_{\text{CFM}}(\hat{b}) := \int_0^1 \mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{x}_1} \left[\left\| \hat{b}_t(I_t(\boldsymbol{x}_0, \boldsymbol{x}_1)) - \partial_t I_t(\boldsymbol{x}_0, \boldsymbol{x}_1) \right\|^2 \right] dt \tag{1}$$

Flow Maps for Trajectory Distillation Integrating the ODE above typically requires many small steps. Consistency models (Song et al., 2023; Song and Dhariwal, 2024) and flow maps (Boffi et al., 2025b; Sabour et al., 2025; Geng et al., 2025; Frans et al., 2025; Guo et al., 2025) distill this process into a few steps. A flow map $X_{s,t} : \mathbb{R}^d \to \mathbb{R}^d$ directly outputs the state of the ODE at time t given its

state at an earlier time s:

$$X_{s,t}(\boldsymbol{x}_s) = \boldsymbol{x}_t = \boldsymbol{x}_s + (t-s)v_{s,t}(\boldsymbol{x}_s), \quad \forall s,t \in [0,1] \tag{2}$$

where $v_{s,t}$ is the *mean velocity*. On the diagonal, $v_{s,s} = b_s$, which we train with the CFM loss in (1), called the *diagonal loss* $\mathcal{L}_{\text{diag}}$. For $t \neq s$, we use three equivalent conditions of the flow map $X_{s,t}$:

$$\textit{Lagrangian Condition:} \quad \partial_t X_{s,t}(\boldsymbol{x}_s) = v_{t,t}(X_{s,t}(\boldsymbol{x})) \tag{3a}$$

$$\textit{Eulerian Condition:} \quad \partial_s X_{s,t}(\boldsymbol{x}) + v_{s,s}(\boldsymbol{x}) \cdot \nabla_{\boldsymbol{x}_s} X_{s,t}(\boldsymbol{x}) = 0 \tag{3b}$$

$$\textit{Semigroup Condition:} \quad X_{u,t}(X_{s,u}(\boldsymbol{x})) = X_{s,t}(\boldsymbol{x}) \tag{3c}$$

for all $\boldsymbol{x} \in \mathbb{R}^d$ and $s,u,t \in [0,1]$. The squared residual of each condition gives a *consistency objective* $\mathcal{L}_{\text{cons}}$, and the full objective is $\mathcal{L}_{\text{FM}}(\hat{v}) = \mathcal{L}_{\text{diag}}(\hat{v}) + \mathcal{L}_{\text{cons}}(\hat{v})$.

Flow Maps for Discrete Data Recently, flow maps have been extended to the discrete state space (Roos et al., 2026; Lee et al., 2026; Potaptchik et al., 2026b), where the states are defined as vectors on the simplex space Δ^{V-1} with vocabulary $|\mathcal{V}| = V$. A sequence of categorical distributions is denoted $\boldsymbol{x}_t \in (\Delta^{V-1})^L$. Representing discrete data as continuous distributions over the simplex enables the repurposing of standard flow-map objectives in the discrete setting. The equivalent parameterization of the mean velocity on the discrete state space is the *mean denoiser* $\psi_{s,t} : \mathbb{R}^{L\times V} \to \mathbb{R}^{L\times V}$ which reduces to the standard denoiser on the diagonal $t = s$ and is defined by taking a full $(1-s)$ step in the direction of the mean velocity $v_{s,t}$ to the denoised state:

$$\psi_{s,t}(\boldsymbol{x}) = \boldsymbol{x} + (1-s)v_{s,t}(\boldsymbol{x}), \quad v_{s,t}(\boldsymbol{x}) = \frac{\psi_{s,t}(\boldsymbol{x}) - \boldsymbol{x}}{1-s} \tag{4}$$

Substituting $v_{s,t}$ into (2) gives the discrete flow map, whose expanding form appears in Section 5. The mean denoiser lies on the simplex, so it can be trained with *cross-entropy* analogs of the diagonal and consistency objectives, whose target $\bar{\psi}_{s,t}$ comes from discrete analogs of the Lagrangian, Eulerian, and semigroup conditions (Lee et al., 2026; Potaptchik et al., 2026b).

3 Expanding Generative Flows

The **key limitation** of standard continuous flows is that the state space is fixed at initialization: $\boldsymbol{x}_0 \in \mathbb{R}^d$ with fixed d in the continuous state space and $\boldsymbol{x}_0 \in \mathbb{R}^{L\times V}$ with fixed length L in the discrete sequence space. We overcome this limitation by presenting a rigorous treatment of generative flows between distributions of *different dimensionality*, specifically the setting where the source distribution has *lower dimensionality* than the target distribution. We construct this by introducing three ideas: an *expanding interpolant* that grows the state along a dimension schedule, an *expand operator* that lifts the source into the target space, and a *piecewise deterministic Markov process (PDMP)* characterization that unifies expansion and transport into a single well-defined generative process.

Expanding Interpolant To accommodate generative tasks in which the size of the output is itself variable, we generalize the standard flow interpolant construction to allow the dimension to increase along the trajectory. Concretely, we replace the fixed dimension d with a non-decreasing dimension schedule $d(t) : [0,1] \to \mathbb{N}$ and let $\boldsymbol{x}_t \in \mathbb{R}^{d(t)}$, so that the marginal distribution $p_t \in \mathcal{P}(\mathbb{R}^{d(t)})$ is defined on the state space whose dimensionality depends on t. We write $d(t)$ as shorthand for $d(\boldsymbol{x}, t)$, the dimension at time t of a specific realization of the interpolant with $I_t = \boldsymbol{x}$. It is not fixed before sampling: when the expand operator is learned, the dimension at t depends on the current state $\boldsymbol{x}_s$. The initial state may be empty with $d(0) = 0$, or predefined with $d(0) > 0$, such as a heavy-atom backbone. A constant $d(t)$ recovers the standard fixed-dimensional flow.

Generative Flow with Expanding Dimensionality

$$\mathcal{P}(\mathbb{R}^{d(0)}) \ni p_0 \to \cdots \to p_s \to \cdots \to p_t \to \cdots \to p_1 \in \mathcal{P}(\mathbb{R}^{d(1)}) \tag{5}$$

where $d(s) \leq d(t)$ for all $0 \leq s < t \leq 1$.

For $s < t$ with $d(s) < d(t)$, the states $\boldsymbol{x}_s$ and $\boldsymbol{x}_t$ live in spaces of different dimensions, so no diffeomorphism or fixed-dimension velocity field can transport p_s to p_t. We therefore *augment* the source state with coordinates of noise drawn from a tractable conditional distribution $p_{\epsilon|s,t} \in \mathcal{P}(\mathbb{R}^{d(t)-d(s)})$ before applying a transport map.

Expand Operator To parameterize the expanding generative flow, we define a *two-stage* process that, at each discrete time step, expands the state space with augmented noise coordinates and then integrates the interpolant from the augmented state to the target time step. The first stage is carried out by the *expand operator* $\mathcal{E}_{s,t}$.

Expand Operator

$$\mathcal{E}_{s,t}(\boldsymbol{x}_s, \boldsymbol{\epsilon}) : \mathbb{R}^{d(s)} \times \mathbb{R}^{d(t)-d(s)} \to \mathbb{R}^{d(t)}, \quad \boldsymbol{\epsilon} \sim p_{\epsilon|s,t} \in \mathbb{R}^{d(t)-d(s)} \tag{6}$$

which defines the placement scheme defining how the coordinates of $\boldsymbol{\epsilon}$ are composed with $\boldsymbol{x}_s$.

The augmented noise variable $\boldsymbol{\epsilon}$ adds new coordinates to the growing state space, which can then be mapped to a sample from the target distribution p_t. The expand operator $\mathcal{E}_{s,t}$ can take several instantiations, including:

(i) **Concatenation** appends the new coordinates to the end of the state, as in $\mathrm{Cat}(\boldsymbol{x}_s, \boldsymbol{\epsilon})$, which is suitable for autoregressive generation and point clouds.

(ii) **Positional Insertion** interleaves the entries of $\boldsymbol{\epsilon}$ at a subset of positions, which is suitable for any-length sequence generation and infilling.

(iii) **Child Expansion** initializes each new coordinate relative to a parent entry, $\boldsymbol{x}_s^\epsilon[i] = \boldsymbol{x}_s[\mathrm{pa}(i)] + \sigma\boldsymbol{\epsilon}_i$, which is suitable for coarse-to-fine generation.

The augmented noise and its placement may be deterministic (e.g., always concatenating i.i.d. Gaussian noise to the end) or predicted via a parameterized network $\hat{\mathcal{E}}$ (e.g., predicting the number of tokens to insert at each gap). In all cases, the expand operator lifts the state to $\mathbb{R}^{d(t)}$ so that transport along the interpolant from s to t can proceed via a $d(t)$-dimensional velocity field. For simplicity, we drop the explicit noise argument and write $\boldsymbol{x}_s^\epsilon := \mathcal{E}_{s,t}(\boldsymbol{x}_s) \equiv \mathcal{E}_{s,t}(\boldsymbol{x}_s, \boldsymbol{\epsilon})$.

The expand operator alone fixes the conditional law of the augmented state, as the pushforward of the noise law (Prop C.2, App C.5). To complete the construction, we specify how each new coordinate is denoised, which requires tracking the time elapsed since its insertion.

Local Time Coordinate Since newly inserted coordinates are instantiated at different points along the interpolant, we assign each inserted component i a *local time coordinate* t_i, with the set of local times denoted $\boldsymbol{t}_{\text{local}}$. At the insertion time t_i^{ins} of component i, the local time is initialized at $t_i = 0$. At global time $t > t_i^{\text{ins}}$, it is defined as:

$$t_i = \frac{t - t_i^{\text{ins}}}{1 - t_i^{\text{ins}}} \in [0, 1] \tag{7}$$

This is a bijection that projects the global time onto a $[0, 1]$ local time interval for each dimension i, so that every inserted dimension is denoised on a common local clock.

Transport Along the Expanding Interpolant After applying the expand operator, we integrate the standard velocity field of the continuous-time interpolant over $\tau \in [s, t]$ from the augmented state $\boldsymbol{x}_s^\epsilon$ to the target state $\boldsymbol{x}_t$:

$$\boldsymbol{x}_t[i] = \boldsymbol{x}_s^\epsilon[i] + \int_{s_i}^{t_i} b_{\tau_i}(\boldsymbol{x}_{\tau_i}^\epsilon[i])d\tau_i, \quad b_t(\boldsymbol{x}_t^\epsilon)[i] = \begin{cases} b_{t_i}(\boldsymbol{x}_{t_i}^\epsilon[i]) & t_i^{\text{ins}} \le t \\ \mathbf{0} & t_i^{\text{ins}} > t \end{cases} \tag{8}$$

where each coordinate i is integrated along its own local time $\tau_i \in [s_i, t_i]$. Since $b_t : \mathbb{R}^{d(t)} \to \mathbb{R}^{d(t)}$ is a standard flow velocity field on the active coordinates ($t_i^{\text{ins}} \le t$), we parameterize it as $\hat{b}_t$ by minimizing the standard CFM objective in (1) and set it to zero otherwise, leaving the inactive coordinates unchanged. Given the expand operator and the transport ODE, we can define the *Expanding Generative Flow* (EFlow).

Definition 3.1 (Expanding Generative Flow). *Let $0 \le s < t \le 1$ with marginals $p_s \in \mathcal{P}(\mathbb{R}^{d(s)})$, $p_t \in \mathcal{P}(\mathbb{R}^{d(t)})$, expand operator $\mathcal{E}_{s,t}$ (6), and conditional noise $p_{\epsilon|s,t}(\cdot \mid \boldsymbol{x}_s) \in \mathcal{P}(\mathbb{R}^{d(t)-d(s)})$. Write $\boldsymbol{x}_s^\epsilon := \mathcal{E}_{s,t}(\boldsymbol{x}_s, \boldsymbol{\epsilon})$ and $p_{s,t}^\epsilon := (\mathcal{E}_{s,t})_\#(p_s \otimes p_{\epsilon|s,t})$. An **expanding generative flow** from p_s to p_t is the composition*

$$\Phi_{s,t} := X_{s,t} \circ \mathcal{E}_{s,t} : \mathbb{R}^{d(s)} \to \mathbb{R}^{d(t)}, \tag{9}$$

where $X_{s,t} : \mathbb{R}^{d(t)} \to \mathbb{R}^{d(t)}$ is the time-t flow map of the augmented interpolant ODE $\dot{\boldsymbol{x}}_\tau^\epsilon = b_\tau(\boldsymbol{x}_\tau^\epsilon)$ (8) initialized at $\boldsymbol{x}_s^\epsilon$ at time s. This map satisfies the marginal constraint $(X_{s,t})_\# p_{s,t}^\epsilon = p_t$, and being the deterministic flow from $\boldsymbol{x}_s^\epsilon$, it fixes the full coupling between $\boldsymbol{x}_s$ and $\boldsymbol{x}_t$ rather than only the marginal p_t. The induced conditional law is

$$p_{t|s}(\cdot \mid \boldsymbol{x}_s) = \big(\Phi_{s,t}(\boldsymbol{x}_s, \cdot)\big)_\# p_{\epsilon|s,t}(\cdot \mid \boldsymbol{x}_s). \tag{10}$$

In practice, we parameterize both the expand operator and the velocity field with a neural network. The jump at each insertion breaks the flow construction at that time, so we provide an alternative characterization that is valid over the full interval $t \in [0, 1]$.

Piecewise-Deterministic Markov Processes The two stages can be unified as a *piecewise-deterministic Markov process* (PDMP) (Davis, 1984) on the disjoint union $\mathcal{M} := \bigsqcup_t \mathbb{R}^{d(t)}$. The process $(\boldsymbol{x}_t, \boldsymbol{t}_{\text{local}})$ has the extended generator (Prop C.1, App C.4):

$$\mathcal{L}_t f(\boldsymbol{x}) = \underbrace{b_t(\boldsymbol{x}) \cdot \nabla f(\boldsymbol{x})}_{\text{transport}} + \underbrace{\lambda_t(\boldsymbol{x}) \int \big(f(\mathcal{E}_t(\boldsymbol{x}, \boldsymbol{\epsilon})) - f(\boldsymbol{x})\big) p_{\epsilon|t}(d\boldsymbol{\epsilon})}_{\text{jumps}}, \tag{11}$$

where λ_t is the insertion intensity, and $\mathcal{E}_t$ and $p_{\epsilon|t}$ are the instantaneous insertion map and noise law, given by the limits of $\mathcal{E}_{s,t}$ and $p_{\epsilon|s,t}$ as $s \to t$. App C.4 gives the corresponding change-of-variables formula (Corollary C.1).

4 Expanding Flow Maps

Given the definition of an *Expanding Generative Flow*, we now define the **Expanding Flow Map** (EFM), which transports between two states along the expanding interpolant in a *single* learned step, jointly increasing dimensionality and pushing the flow forward in time.

Flow Map in Augmented State Space We define the expanding flow map (EFM) $\Phi_{s,t} : \mathbb{R}^{d(s)} \to \mathbb{R}^{d(t)}$ as the composition of the expand operator $\mathcal{E}_{s,t}$ (6), which raises the dimension from $d(s)$ to $d(t)$, and a transport map $X_{s,t}$ in $d(t)$-dimensional space that pushes the state forward along the *local time coordinate* of each dimension. In this section, we define the transport map as the mean velocity $v_{s,t}$ for continuous data, and in Section 5, we define it as the mean denoiser $\psi_{s,t}$ for discrete data.

Expanding Flow Map

$$\Phi_{s,t} = X_{s,t} \circ \mathcal{E}_{s,t}, \quad \Phi_{s,t}(\boldsymbol{x}_s) := X_{s,t}(\mathcal{E}_{s,t}(\boldsymbol{x}_s)) =: \mathcal{E}_{s,t}(\boldsymbol{x}_s) + (t-s) v_{s,t}(\mathcal{E}_{s,t}(\boldsymbol{x}_s)) \tag{12}$$

$\forall s, t \in [0, 1]$, where $\mathcal{E}_{s,t} : \mathbb{R}^{d(s)} \to \mathbb{R}^{d(t)}$ is the expand operator and $v_{s,t} : \mathbb{R}^{d(t)} \to \mathbb{R}^{d(t)}$ is the mean velocity that parameterizes the transport map.

Leveraging the *expand operator* $\mathcal{E}_{s,t}$ from (6), which lifts the state $\boldsymbol{x}_s \in \mathbb{R}^{d(s)}$ to the dimensionality of the target state along the interpolant $\boldsymbol{x}_t \in \mathbb{R}^{d(t)}$, we can apply the standard residual form of the flow map via the average velocity $v_{s,t}$, which satisfies the *tangent condition* (Kim et al., 2024):

$$\lim_{s \to t} \partial_t \Phi_{s,t}(\boldsymbol{x}) = v_{t,t}(\boldsymbol{x}) = b_t(\boldsymbol{x}) \tag{13}$$

where b_t is the standard flow velocity field defined in (8). This condition can be enforced via the diagonal objective:

$$\mathcal{L}_{\text{diag}}(\hat{v}) := \int_0^1 \mathbb{E}\left\|\hat{v}_{t,t}(\boldsymbol{x}_t) - \dot{\boldsymbol{x}}_t\right\|^2 dt, \quad \dot{\boldsymbol{x}}_t = \begin{cases} \left(\frac{\boldsymbol{x}_1^i - \boldsymbol{x}_0^i}{1 - t_i^{\text{ins}}}\right)_{i:\, t_i^{\text{ins}} \le t} & \text{(self-distillation)} \\ \hat{b}_t(\boldsymbol{x}_t) & \text{(distillation from teacher)} \end{cases} \tag{14}$$

Each active coordinate follows the expanding interpolant $\boldsymbol{x}_t[i] = (1 - t_i)\boldsymbol{x}_0^i + t_i \boldsymbol{x}_1^i$ at its local time t_i, and the regression target is the time-derivative $\dot{\boldsymbol{x}}_t$ in global time.

Expanding Flow Map Identities In addition to the tangent condition, the expanding flow map $\Phi_{s,t}$ must also satisfy analogous *consistency constraints*, which characterize a valid flow map as introduced in (3). Given our parameterization, we define these constraints with respect to the *expand operator* $\hat{\mathcal{E}}_{s,t}$ and the average velocity $\hat{v}_{s,t}$ in the continuous-state setting.

Proposition 4.1 (Expanding Flow Map Identities)**.** *Let $\boldsymbol{x}_s^\epsilon := \mathcal{E}_{s,t}(\boldsymbol{x}_s)$. Then, the expanding flow map $\Phi_{s,t}(\boldsymbol{x}_s)$ satisfies the following consistency conditions:*

$$\textit{Lagrangian:} \quad \partial_t \Phi_{s,t}(\boldsymbol{x}) = v_{t,t}(\Phi_{s,t}(\boldsymbol{x})) \tag{15a}$$

$$\textit{Eulerian:} \quad \partial_s \Phi_{s,t}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} \Phi_{s,t}(\boldsymbol{x}) \tilde{v}_{s,s}(\boldsymbol{x}) = 0 \tag{15b}$$

$$\textit{Semigroup:} \quad \Phi_{u,t}(\Phi_{s,u}(\boldsymbol{x})) = \Phi_{s,t}(\boldsymbol{x}) \tag{15c}$$

where $\tilde{v}_{s,s} : \mathbb{R}^{d(s)} \to \mathbb{R}^{d(s)}$ is the unlifted diagonal velocity of A4, satisfying $v_{s,s}(\mathcal{E}_{s,t}(\boldsymbol{x})) = E\tilde{v}_{s,s}(\boldsymbol{x})$. $\Phi_{s,t}(\boldsymbol{x}) := X_{s,t}(\mathcal{E}_{s,t}(\boldsymbol{x})) = \mathcal{E}_{s,t}(\boldsymbol{x}) + (t - s)v_{s,t}(\mathcal{E}_{s,t}(\boldsymbol{x}))$ is the residual form of the expanding flow map in the continuous state space.

These identities coincide with the fixed-dimensional flow map identities (3) with proof in Boffi et al. (2025b). The only difference is in the Eulerian condition, where the Jacobian $\nabla_{\boldsymbol{x}} \Phi_{s,t} \in \mathbb{R}^{d(t) \times d(s)}$ is rectangular, so multiplying with the unlifted velocity $\tilde{v}_{s,s}(\boldsymbol{x}) \in \mathbb{R}^{d(s)}$ of Assumption A4 gives the diagonal velocity $v_{s,s}$ on the expanded state $\mathcal{E}_{s,t}(\boldsymbol{x}) \in \mathbb{R}^{d(t)}$. We provide proof in App C.6. To enforce these constraints, we define the analogous consistency objectives:

$$\mathcal{L}_{\text{LSD}}(\hat{\mathcal{E}}, \hat{v}) := \int_0^1 \int_0^t \mathbb{E}\left\|\partial_t \hat{\Phi}_{s,t}(\boldsymbol{x}_s) - \text{sg}\left(\hat{b}_t(\hat{\Phi}_{s,t}(\boldsymbol{x}_s))\right)\right\|^2 dsdt + \mathcal{L}_{\text{diag}}(\hat{v}) \tag{16a}$$

$$\mathcal{L}_{\text{ESD}}(\hat{\mathcal{E}}, \hat{v}) := \int_0^1 \int_0^t \mathbb{E}\left\|\partial_s \hat{\Phi}_{s,t}(\boldsymbol{x}_s) + \text{sg}\left(\nabla_{\boldsymbol{x}} \hat{\Phi}_{s,t}(\boldsymbol{x}_s) \hat{b}_s(\boldsymbol{x}_s)\right)\right\|^2 dsdt + \mathcal{L}_{\text{diag}}(\hat{v}) \tag{16b}$$

$$\mathcal{L}_{\text{PSD}}(\hat{\mathcal{E}}, \hat{v}) := \int_0^1 \int_0^t \int_s^t \mathbb{E}\left\|\hat{\Phi}_{s,t}(\boldsymbol{x}_s) - \text{sg}(\hat{\Phi}_{u,t}(\hat{\Phi}_{s,u}(\boldsymbol{x}_s)))\right\|^2 dudsdt + \mathcal{L}_{\text{diag}}(\hat{v}) \tag{16c}$$

where unlike fixed-dimensional flow maps, $\hat{\Phi}_{s,t}$ is parameterized via both $\hat{\mathcal{E}}_{s,t}$ and $\hat{v}_{s,t}$.

Stochastic Flow Map Interpretation The expanding flow map defines a *stochastic flow*, whose randomness comes from the explicit conditional noise. For fixed $\boldsymbol{x}_s$, it pushes $p_{\epsilon|s,t}$ forward to $p_{t|s}(\cdot \mid \boldsymbol{x}_s)$, and recovers the clean posterior at $t = 1$ (Prop C.3, App C.7).

5 Discrete Expanding Flow Maps

The expanding generative flows and expanding flow maps framework is naturally positioned for *variable-length discrete generation*, where a sequence or graph dynamically expands from a single token or node into a full sequence or graph. We instantiate the objects of Sections 3 and 4 on the discrete simplex: the expand operator $\mathcal{E}_{s,t}$ becomes *token insertions* and the transport $X_{s,t}$ becomes *token denoising*.

Discrete Expanding Interpolant for Growing Sequence Lengths We first define the *discrete expanding interpolant*, in which the number of tokens $d(t) : [0, 1] \to \mathbb{N}$ grows over intermediate times. At training time, we fix an *insertion schedule* α_t interpolating between $\alpha_0 = 0$ and $\alpha_1 = 1$. An intermediate sequence $\boldsymbol{x}_t$ is constructed by first sampling a clean sequence from the data distribution, $\boldsymbol{x}_1 \sim p_1$, and then independently sampling an insertion time t_i^{ins} for each position i according to the

schedule. Each position is either not yet inserted, denoted (empty), or in an intermediate latent state $\boldsymbol{x}_{t_i}^i$ along the continuous-time flow at its rescaled local time t_i:

$$\Pr[t_i^{\text{ins}} \le t] = \alpha_t, \quad \boldsymbol{x}_t^i = \begin{cases} \text{(empty)}, & t < t_i^{\text{ins}} \\ (1-t_i)\boldsymbol{x}_0^i + t_i \boldsymbol{x}_1^i, & t \ge t_i^{\text{ins}} \end{cases}, \quad t_i = \frac{t - t_i^{\text{ins}}}{1 - t_i^{\text{ins}}} \tag{17}$$

A sequence at global time t is therefore the subsequence of currently inserted tokens, $\boldsymbol{x}_t \in \mathbb{R}^{d(t)\times V}$.

Parameterizing Token Insertions Given an intermediate subsequence $\boldsymbol{x}_s$ and a target time $t > s$, we decide where and how many V-dimensional latent tokens to insert on the jump via the (two-time) *insertion expectation* $\mathcal{I}_{s,t}(\boldsymbol{x}_s)[i]$, the expected number of tokens inserted into gap i (the slot between active tokens $\boldsymbol{x}_s^{i-1}$ and $\boldsymbol{x}_s^i$) between time s and t:

$$\mathcal{I}_{s,t}(\boldsymbol{x}_s)[i] = \tfrac{\alpha_t - \alpha_s}{1-\alpha_s} \mathcal{I}_s(\boldsymbol{x}_s)[i], \quad \mathcal{I}_t(\boldsymbol{x}_t)[i] := \mathbb{E}\big[g_i(t) \mid \boldsymbol{x}_t\big] \tag{18}$$

where $g_i(t) := \text{ind}_i(t) - \text{ind}_{i-1}(t) - 1$ is the count still missing from gap i at t, $\text{ind}_i(t)$ is the index of active position i in $\boldsymbol{x}_1$, and $\frac{\alpha_t - \alpha_s}{1-\alpha_s}$ is the fraction of the remaining tokens inserted over $[s,t]$ with the insertion hazard $h_t := \frac{\dot\alpha_t}{1-\alpha_t}$ defining the instantaneous insertion rate of an uninserted token at time t. The insertion head $\hat{\mathcal{I}}_{s,t}(\boldsymbol{x}_s)$ is conditioned on both endpoints and trained on diagonal and off-diagonal time pairs with the insertion loss:

$$\mathcal{L}_{\text{insert}}(\hat{\mathcal{I}}) = \mathop{\mathbb{E}}_{t,\boldsymbol{x}_0,\boldsymbol{x}_1}\Big[\tfrac{\dot\alpha_t}{1-\alpha_t} \textstyle\sum_i \phi\big(g_i(t), \hat{\mathcal{I}}_t(\boldsymbol{x}_t)[i]\big)\Big] + \mathop{\mathbb{E}}_{s<t,\boldsymbol{x}_0,\boldsymbol{x}_1}\Big[\textstyle\sum_i \phi\big(\mu_i(s,t), \hat{\mathcal{I}}_{s,t}(\boldsymbol{x}_s)[i]\big)\Big] \tag{19}$$

where $\phi(a,b) := b - a + a\log\frac{a}{b}$ is the Poisson NLL of count a under mean b up to a b-independent constant, so its minimizer is the conditional mean. The diagonal term supervises against $g_i(t)$, and the off-diagonal term supervises against the interval count $\mu_i(s,t) := \#\{j : t_j^{\text{ins}} \in (s,t], j \in \text{gap } i\}$, whose mean is $\mathcal{I}_{s,t}(\boldsymbol{x}_s)[i]$.

Discrete Expand Operator Given the learned insertion expectation, the *discrete expand operator* $\mathcal{E}_{s,t}$ inserts latent tokens $\boldsymbol{x}_0 \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_V)$ with prior scale σ and local time $t_i = 0$ per gap according to the gap-lengths ℓ_i:

$$\mathcal{E}_{s,t}(\boldsymbol{x}_s) := \boldsymbol{x}_s^\epsilon = \left[\bigoplus_{i=1}^{d(s)} \big(\boldsymbol{x}_0^{\ell_i} \oplus \boldsymbol{x}_{s_i}^i\big)\right] \oplus \boldsymbol{x}_0^{\ell_{d(s)+1}} \tag{20}$$

Since the head predicts the interval count $\hat{\mathcal{I}}_{s,t}(\boldsymbol{x}_s)[i]$ directly, we draw ℓ_i from a binomial that matches this predicted mean and is capped at the remaining budget:

$$\ell_i \sim \text{Binomial}\left(L - n_s,\ \tfrac{\hat{\mathcal{I}}_{s,t}(\boldsymbol{x}_s)[i]}{L - n_s}\right) \tag{21}$$

We truncate the joint draw left-to-right so that $\sum_i \ell_i \le L - n_s$. Many-step sampling inserts few tokens per step, so $L - n_s \gg \hat{\mathcal{I}}_{s,t}$ and (21) converges to Poisson$(\hat{\mathcal{I}}_{s,t}(\boldsymbol{x}_s)[i])$, the infinitesimal jump law of the PDMP of Section 3 with insertion intensity $\lambda_t := h_t \sum_i \mathcal{I}_t(\boldsymbol{x}_t)[i]$. Near the boundary pair $(s,t) = (0,1)$, the operator must produce the whole sequence in one jump. If the sequence nearly reaches the max token length, the binomial variance $np(1-p)$ vanishes as $p \to 1$, whereas the Poisson law has variance equal to its mean. This concentration relies on L being close to the true length.

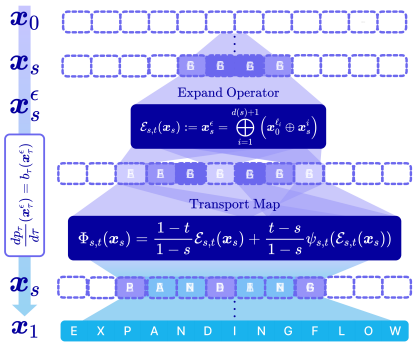

Figure 2: **Discrete Expanding Flow Maps.** Latent tokens are inserted via the expand operator $\mathcal{E}_{s,t}$ and denoised via the transport map defined with the mean denoiser $\psi_{s,t}$.

Discrete Expanding Flow Map Following (4), we define the *expanding mean denoiser* as $\psi_{s,t}(\mathcal{E}_{s,t}(\boldsymbol{x}_s)) := \mathcal{E}_{s,t}(\boldsymbol{x}_s) + (1-s)v_{s,t}(\mathcal{E}_{s,t}(\boldsymbol{x}_s))$. Substituting this into (12) yields the *discrete expanding flow map*, which jointly inserts and denoises tokens.

Table 1: **Molecular conformer generation results on GEOM-Drugs and GEOM-QM9**. Comparison against state-of-the-art multi-step diffusion and deep generative models. +R indicates sampling with 10 additional steps, using a learned coordinate-refinement network. Reported baseline values are taken from (Hu et al., 2026a). Best values are **bolded**, and second-best are underlined.

		GEOM-QM9 ($\delta = 0.5$ Å)				**GEOM-Drugs** ($\delta = 1.25$ Å)			
Models	Steps ↓	COV-R(%) ↑ Mean/Median	AMR-R(Å) ↓ Mean/Median	COV-P(%) ↑ Mean/Median	AMR-P(Å) ↓ Mean/Median	COV-R(%) ↑ Mean/Median	AMR-R(Å) ↓ Mean/Median	COV-P(%) ↑ Mean/Median	AMR-P(Å) ↓ Mean/Median
GraphDG	500	73.33/84.21	0.4245/0.3973	43.90/35.33	0.5809/0.5823	8.27/0.00	1.9722/1.9845	2.08/0.00	2.4340/2.4100
CGCF	500	78.05/82.48	0.4219/0.3900	36.49/33.57	0.6615/0.6427	53.96/57.06	1.2487/1.2247	21.68/13.72	1.8671/1.8066
ConfVAE	500	55.84/88.20	0.4154/0.3739	38.02/34.67	0.6215/0.6091	55.20/59.43	1.2380/1.1417	22.96/14.05	1.8287/1.8159
GeoMol	500	71.26/72.00	0.3731/0.3731	-/-	-/-	67.16/71.71	1.0875/1.0586	-/-	-/-
ConfGF	500	88.49/94.31	0.2673/0.2685	46.43/43.41	0.5224/0.5124	62.15/70.93	1.1629/1.1596	23.42/15.52	1.7219/1.6863
GeoDiff	500	87.80/93.66	0.3179/0.3216	46.25/45.02	0.6173/0.5112	64.12/75.56	1.1444/1.1296	43.16/42.02	1.3806/1.3314
SubgDiff	500	**89.40**/94.39	**0.2543**/0.2601	49.21/47.45	0.5030/0.4724	74.30/77.87	1.0003/0.9905	-/-	-/-
MSGEN	1000	88.36/**95.10**	0.2606/**0.2332**	49.54/48.39	0.4934/**0.4624**	85.06/**97.16**	0.9538/0.9478	49.28/48.20	1.3140/1.2719
EFlow	20	88.51/93.01	0.3849/0.3585	53.07/52.27	0.5185/0.4891	81.78/90.91	0.9971/0.9826	55.88/61.58	1.2672/1.2028
EFlow+R	30	88.87/94.05	0.3212/0.2924	54.86/52.38	**0.4876**/0.4636	85.80/93.85	0.9069/0.8855	64.05/70.38	1.1663/1.0997
EFM	1	69.21/69.23	0.4884/0.4635	32.18/29.09	0.6093/0.5794	63.28/72.47	1.1921/1.1721	31.03/26.09	1.4966/1.4454
EFM	2	72.83/74.25	0.4703/0.4436	34.09/31.33	0.6000/0.5654	77.21/88.67	1.0740/1.0499	47.14/49.24	1.3483/1.2966
EFM	4	83.62/86.91	0.4165/0.3954	48.76/45.26	0.5412/0.5159	81.26/91.13	1.0201/1.0029	53.37/56.72	1.2883/1.2354
EFM+R	14	87.79/92.01	0.3271/0.3049	**55.56/53.09**	0.4889/0.4691	**87.97**/95.45	**0.9029/0.8836**	**64.67/73.25**	**1.1575/1.0875**

Discrete Expanding Flow Map

$$\Phi_{s,t}(\boldsymbol{x}_s) := X_{s,t}(\mathcal{E}_{s,t}(\boldsymbol{x}_s)) = \frac{1-t}{1-s}\mathcal{E}_{s,t}(\boldsymbol{x}_s) + \frac{t-s}{1-s}\psi_{s,t}(\mathcal{E}_{s,t}(\boldsymbol{x}_s)) \tag{22}$$

where $\mathcal{E}_{s,t}(\boldsymbol{x}_s) : \mathbb{R}^{d(s)\times V} \to \mathbb{R}^{d(t)\times V}$ is the expand operator from sequence length $d(s)$ to $d(t)$ and vocabulary size V and $\psi_{s,t} : \mathbb{R}^{d(t)\times V} \to (\Delta^{V-1})^{d(t)}$ is the mean denoiser that parameterizes the transport map.

Discrete Expanding Flow Map Identities To parameterize discrete EFM, we extend the continuous-space consistency identities (15) to the discrete setting with respect to the mean denoiser on the expanded state $\psi_{s,t}(\mathcal{E}_{s,t}(\boldsymbol{x}))$, extending prior work on fixed-length discrete flow maps (Lee et al., 2026; Potaptchik et al., 2026b).

Prop C.4 (App C.8) gives the discrete Lagrangian, Eulerian, and semigroup identities. We train with the semigroup identity, with $\omega_{s,u,t} := \frac{(u-s)(1-t)}{(t-s)(1-u)}$, and we write $\psi^{(t)}_{s,u}$ for the mean denoiser over $[s,u]$ applied at the augmented dimension $d(t)$, which leaves coordinates inserted over $[u,t]$ unchanged. Since $\hat{\psi} : \mathbb{R}^{d(t)\times V} \to (\Delta^{V-1})^{d(t)}$, we follow (Lee et al., 2026) and define the discrete EFM objective in the cross-entropy form:

$$\mathcal{L}_{\text{DEFM}}(\hat{\psi}) := \mathbb{E}_{s,u,t}\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{x}_1}\left[-\sum_{i=1}^{d(t)}(\bar{\psi}_{s,t}\cdot\log\hat{\psi}_{s,t})(\boldsymbol{x}^\epsilon_s)_i\right] + \mathbb{E}_t\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{x}_1}\left[-\sum_{i=1}^{d(t)}(D_t\cdot\log\hat{\psi}_{t,t})(\boldsymbol{x}^i_t)\right] \tag{23}$$

where $D_t : \mathbb{R}^{d(t)\times V} \to (\Delta^{V-1})^{d(t)}$ is either a teacher instantaneous denoiser or the diagonal values of the student mean denoiser. Both terms in the target act on the state $\boldsymbol{x}^\epsilon_s = \mathcal{E}_{s,t}(\boldsymbol{x}_s)$ expanded once to the terminal length $d(t)$, with positions inserted after u left empty:

$$\bar{\psi}_{s,t}(\boldsymbol{x}^\epsilon_s) := \text{sg}\left[\omega_{s,u,t}\hat{\psi}^{(t)}_{s,u}(\boldsymbol{x}^\epsilon_s) + (1-\omega_{s,u,t})\hat{\psi}_{u,t}\big(\hat{\Phi}^{(t)}_{s,u}(\boldsymbol{x}^\epsilon_s)\big)\right] \tag{24}$$

which enforces the discrete semigroup condition in Prop C.4 as shown in App C.8 and, together with $\mathcal{L}_{\text{insert}}$, trains the mean denoiser and expand operator for few-step generation in the discrete setting. The training procedures for discrete EFlow and EFM are provided in Algorithms 1 and 2.

6 Experiments

To establish EFlow and EFM as a generalizable framework for generative modeling with variable canvas sizes across continuous and discrete state spaces, we evaluate our approach on three diverse tasks: coarse-to-fine molecular conformer generation on continuous 3D coordinate space (Section 6.1), discrete molecular graph generation (Section 6.2), and language modeling (Section 6.3).

6.1 Coarse-to-Fine Molecular Conformer Generation

Setup and Training Framework We evaluate our framework on molecular conformer generation over the GEOM-QM9 and GEOM-Drugs datasets (Axelrod and Gomez-Bombarelli, 2022), following the GeoDiff split (Xu et al., 2022), and report Coverage (COV) and Average Minimum RMSD (AMR) in both recall (R) and precision (P) forms at thresholds $\delta = 0.5$ Å for QM9 and $\delta = 1.25$ Å for Drugs (Table 1; see App F.1 for experimental details). EFlow treats a conformer as a coarse-to-fine expanding interpolant. We denoise the heavy-atom backbone from Gaussian noise and freeze it at an insertion-end time, then insert and resolve the hydrogens around it. One network with per-atom local times generates the all-atom conformer, and an optional low-noise refiner (EFlow+R, EFM+R) refines the result. The velocity field is the GeoDiff dual-encoder (Xu et al., 2022) used by MSGEN (Hu et al., 2026a). Since the molecular graph fixes the atom count, the expand operator is the deterministic case of child expansion, with a fixed insertion schedule.

Results We compare against the multi-step diffusion baselines GeoDiff (Xu et al., 2022), SubgDiff (Zhang et al., 2024), and the two-stage MSGEN (Hu et al., 2026a), which run 500-1000 steps, alongside classical and generative baselines (App F.1). EFlow uses 16-50× fewer steps, remains competitive on coverage, and improves on precision. On GEOM-Drugs at 20 steps, EFlow exceeds MSGEN on both mean precision metrics. On GEOM-QM9, EFlow+R attains the best mean AMR-P of any method (0.4876 Å) at 30 steps. The largest gains are on GEOM-Drugs (up to 181 atoms), where our methods take the best and second-best mean on all four metrics. After distillation, EFM generates conformers in as few as 4, 2, and 1 steps, and EFM+R attains the best mean on all four GEOM-Drugs metrics at 14 steps. We also evaluate ensemble properties (Axelrod and Gomez-Bombarelli, 2022) for 30 GEOM-QM9 molecules with Psi4 (Smith et al., 2020) (Table 5). These results establish expanding flows as a single framework that scales from multi-step to few-step conformer generation.

6.2 Discrete Molecular Graph Generation

Setup and Training Framework We evaluate our framework on small molecules from the QM9 dataset (Wu et al., 2018), reporting validity, uniqueness, and Fréchet ChemNet Distance (FCD) (Preuer et al., 2018) on 10,000 generated molecules (Table 2; see App F.2 for experimental details). EFlow treats a molecule as a discrete expanding interpolant over a variable number of nodes. The state is a pair $(\boldsymbol{x}, \boldsymbol{E})$ of node categories and a symmetric edge-category matrix. Each node has an independent insertion time, a learned head predicts the per-gap node count, and a single network jointly denoises node and edge categories. The denoiser is the DeFoG graph transformer (Qin et al., 2025) (see App E).

Results First, we evaluate EFlow without any distillation against the state-of-the-art discrete graph flow model DeFoG (Qin et al., 2025). At a 100-step budget, EFlow already surpasses DeFoG with lower FCD, with greater improvement as the step budget shrinks. From 100 to 10 steps, EFlow's FCD rises from 0.116 to 0.390 while validity and uniqueness remain above 95%. Over the same range, DeFoG's FCD rises from 0.134 to 1.630, and its validity falls to 53.6% at 4 steps, while EFlow still produces 91.7% valid and 92.8% unique molecules at FCD 1.780. Notably, this is the non-distilled EFlow teacher, suggesting meaningful room for further gains in the few-step regime.

Table 2: **Discrete molecular graph generation results.** Comparison against state-of-the-art multi-step baseline DeFoG (Qin et al., 2025) (top) and flow map baseline CFM (Roos et al., 2026) (bottom) on 10,000 molecules with FCD computed on the test split. The baselines sample node counts from the training distribution. Best values in each row are **bolded**.

Multi-Step	DeFoG			EFlow (ours)		
Steps	**Valid ↑**	**Unique ↑**	**FCD ↓**	**Valid ↑**	**Unique ↑**	**FCD ↓**
100	**99.2**	**96.2**	0.134	98.9	95.9	**0.116**
50	**98.8**	**96.2**	0.260	**98.8**	96.1	**0.131**
10	88.6	92.8	1.630	**97.9**	**96.1**	**0.390**
4	53.6	63.1	3.008	**91.7**	**92.8**	**1.780**
Few-Step	CFM			EFM (ours)		
Steps	**Valid ↑**	**Unique ↑**	**FCD ↓**	**Valid ↑**	**Unique ↑**	**FCD ↓**
2	**91.1**	97.6	0.49	90.0	**97.9**	**0.40**
1	90.0	91.8	2.14	**90.2**	**97.7**	**0.44**

We then evaluate the distilled EFM student against categorical flow maps (CFM) (Roos et al., 2026) in the one- to two-step regime (Table 2). In two steps, EFM is comparable to CFM in validity and uniqueness and achieves a lower FCD (0.40 vs. 0.49). EFM's advantage widens at a single step, where CFM's FCD degrades to 2.14, and uniqueness drops to 91.8%, while EFM's FCD remains

Table 3: **EFlow performance on LM1B.** Gen. PPL (↓) and entropy across sampling budgets. FLM results are computed by running publicly released checkpoints from Lee et al. (2026). Best Gen. PPL values are **bolded**.

	EFlow (ours)		**FLM**	
Steps	Gen. PPL ↓	Ent.	Gen. PPL ↓	Ent.
1024	**103.63**	4.16	111.36	4.30
512	**110.97**	4.16	114.89	4.31
256	**117.69**	4.16	120.19	4.32
128	**126.20**	4.15	129.06	4.34
64	**133.62**	4.14	143.73	4.36

Table 4: **EFM performance on LM1B.** Generative PPL (↓) and entropy (↑) across step counts 1, 2, 4. All reported baseline results are taken from Lee et al. (2026). Best values are **bolded**, and second-best are underlined. †Not ranked because one-step sampling collapses toward a mode.

Steps	1		2		4	
Method	Gen. PPL ↓	Ent.	Gen. PPL ↓	Ent.	Gen. PPL ↓	Ent.
Duo + DCD	1224.52	4.33	520.08	4.20	210.88	4.23
Duo + Di4C	292.94	3.79	247.69	3.87	150.67	4.00
MDLM + SDTT	1429.48	4.31	602.14	4.28	241.01	4.28
MDLM + Di4C	1217.10	4.38	621.59	4.37	247.32	4.00
CFM	269.72	3.10	267.39	3.15	267.97	3.28
FMLM	**119.34**	4.16	**110.19**	4.21	**98.76**	4.21
EFM (Ours)	98.35†	3.56	125.95	3.95	99.85	3.93

strong at 0.44 with 97.7% uniqueness. At one step, EFM's FCD is nearly 5× lower than CFM's (0.44 vs. 2.14). On QM9, expanding flows therefore perform well from multi-step to one-step generation of variable-sized graphs.

6.3 Variable-Length Language Modeling

Setup and Training Framework Finally, we evaluate unconditional text generation on the One Billion Word benchmark (LM1B) (Chelba et al., 2014), reporting generative perplexity under GPT-2-Large (Radford et al., 2019) and token entropy (Tables 3 and 4; details in App D and App F.3). Each token has an independent insertion time, and a learned head predicts per-gap insertions, so the sequence grows from an empty sequence to a clean sequence. Sampling integrates the instantaneous or mean denoiser and decodes with the argmax at the final step. Because the vocabulary is large ($V = 30{,}522$), we restrict insertion times to $[0, t_{\text{ins-end}}]$ to leave enough denoising time. We discuss the language-specific algorithmic choices in App D, and provide experiment details in App F.3.

Results As shown in Table 3, EFlow improves on the fixed-length flow language model FLM (Lee et al., 2026) at every step budget we evaluate, despite solving the strictly more expressive variable-length problem. Generative perplexity falls monotonically with the sampling budget, from 133.62 at 64 steps to 103.63 at 1024 steps. Notably, EFlow beats FLM in Gen. PPL at all step counts, indicating that variable-length conditioning does not sacrifice sample quality. Entropy is flat across the sweep at 4.14-4.16, slightly below FLM's 4.30-4.36, but remains coherent as shown in Figure 4. Together, these results indicate that the expand-transport decomposition adds variable-length flexibility without sacrificing sample quality.

We then compare the distilled EFM student in the few-step regime (Table 4) with Duo (Sahoo et al., 2025) and MDLM (Sahoo et al., 2024) distilled with DCD, SDTT (Deschenaux and Gulcehre, 2025), or Di4C (Hayakawa et al., 2025), as well as categorical flow maps (CFM) (Roos et al., 2026) and flow map language models (FMLM) (Lee et al., 2026). EFM generates coherent text at 2 and 4 steps (Fig 5). It outperforms CFM and all distilled diffusion baselines by a wide margin and remains competitive with FMLM. Single-step sampling shows mode collapse, which we attribute to the difficulty of predicting the full expansion and applying the flow map over the entire interval in one jump (App D.5). Sample entropy is also lower than for the fixed-length baselines. One likely factor is that EFM conditions on local time coordinates instead of a shared uniform schedule, and we expect that further schedule tuning will improve performance. To our knowledge, this is the first demonstration of flow maps on variable-length discrete sequences, expanding the design space of existing flow map frameworks that operate on a fixed-length token grid with uniform time conditioning.

7 Conclusion

We introduced **Expanding Generative Flows** (EFlow) and **Expanding Flow Maps** (EFMs), a unified framework that recasts generative modeling as learning a transport along an expanding interpolant between marginals of increasing dimensionality. By composing an *expand operator*, which projects a state into a higher-dimensional space via conditional noise augmentation, and a *transport map*, which pushes the expanded state forward toward the target distribution, EFMs perform few-step generation across dimensionalities in continuous and discrete state spaces.

Declarations

Acknowledgements We thank Mark III Systems for providing database and hardware support that has contributed to the research reported within this manuscript.

Author Contributions S.T. devised and developed model architectures and theoretical formulations and performed experiments. S.T. drafted the manuscript and designed the figures. P.C. supervised and directed the study, and reviewed and finalized the manuscript.

Data and Materials Availability The codebase is freely accessible to the academic community at https://github.com/sophtang/ExpandingFlowMaps and at https://huggingface.co/ChatterjeeLab/ExpandingFlowMaps.

Funding Statement This research was supported by NIH grant R35GM155282 to the lab of P.C..

Competing Interests P.C. is a co-founder of Gameto, Inc., UbiquiTx, Inc., AtomBioworks, Inc., and Kosha Sciences, Inc. and advises companies involved in therapeutics development. P.C.'s interests are reviewed and managed by the University of Pennsylvania in accordance with their conflict-of-interest policies. S.T. has no conflicts of interest to declare.

References

Michael Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. 2025. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 26(209):1–80.

Michael S Albergo and Eric Vanden-Eijnden. 2023. Building normalizing flows with stochastic interpolants. *International Conference on Learning Representations*.

Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. 2025. Block diffusion: Interpolating between autoregressive and diffusion language models. *International Conference on Learning Representations*, 2025:50726–50753.

Simon Axelrod and Rafael Gomez-Bombarelli. 2022. Geom, energy-annotated molecular conformations for property prediction and molecular generation. *Scientific data*, 9(1):185.

Lukas Billera, Hedwig Nora Nordlinder, Jack Collier Ryder, Anton Oresten, Aron Stålmarck, Theodor Mosetti Björk, and Ben Murrell. 2025. Branching flows: Discrete, continuous, and manifold flow matching with splits and deletions. *arXiv preprint arXiv:2511.09465*.

Nicholas M Boffi, Michael S Albergo, and Eric Vanden-Eijnden. 2025a. Flow map matching with stochastic interpolants: A mathematical framework for consistency models. *Transactions on Machine Learning Research*.

Nicholas M Boffi, Michael S Albergo, and Eric Vanden-Eijnden. 2025b. How to build a consistency model: Learning flow maps via self-distillation. *Advances in Neural Information Processing Systems*, 38:33346–33382.

Andrew Campbell, William Harvey, Christian Weilbach, Valentin De Bortoli, Thomas Rainforth, and Arnaud Doucet. 2023. Trans-dimensional generative modeling via jump diffusion models. *Advances in Neural Information Processing Systems*, 36:42217–42257.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. *Interspeech*, pages 2635–2639.

Yuxin Chen, Chumeng Liang, Hangke Sui, Ruihan Guo, Chaoran Cheng, Jiaxuan You, and Ge Liu. 2026. Langflow: Continuous diffusion rivals discrete in language modeling. *Advances in Neural Information Processing Systems*.

Chaoran Cheng, Jiahan Li, Jiajun Fan, and Ge Liu. 2025. α-flow: A unified framework for continuous-state discrete flow matching models. *arXiv preprint arXiv:2504.10283*.

Mark HA Davis. 1984. Piecewise-deterministic markov processes: A general class of non-diffusion stochastic models. *Journal of the Royal Statistical Society: Series B (Methodological)*, 46(3):353–376.

Oscar Davis, Samuel Kessler, Mircea Petrache, İsmail İ Ceylan, Michael Bronstein, and Avishek J Bose. 2024. Fisher flow matching for generative modeling over discrete data. *Advances in Neural Information Processing Systems*, 37:139054–139084.

Warren L DeLano et al. 2002. Pymol: An open-source molecular graphics tool. *CCP4 Newsl. protein crystallogr*, 40(1):82–92.

Justin Deschenaux and Caglar Gulcehre. 2025. Beyond autoregression: Fast llms via self-distillation through time. In *International Conference on Learning Representations*, volume 2025, pages 5007–5045.

Justin Deschenaux and Caglar Gulcehre. 2026. Language modeling with hyperspherical flows. *Transactions on Machine Learning Research*.

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

Fangyu Ding, Ding Ding, Sijin Chen, Kaibo Wang, Peng Xu, Zijin Feng, Haoli Bai, Kai Han, Youliang Yan, Binhang Yuan, et al. 2026. Beyond masks: Efficient, flexible diffusion language models via deletion-insertion processes. *International Conference on Learning Representations*, 2026:103093–103149.

Ian Dunn and David R Koes. 2026. Flowmol3: flow matching for 3d de novo small-molecule generation. *Digital Discovery*, 5(5):2052–2066.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. 2025. One step diffusion via shortcut models. *International Conference on Learning Representations*, 2025:34668–34684.

Octavian Ganea, Lagnajit Pattanaik, Connor Coley, Regina Barzilay, Klavs Jensen, William Green, and Tommi Jaakkola. 2021. Geomol: Torsional geometric generation of molecular 3d conformer ensembles. *Advances in Neural Information Processing Systems*, 34:13757–13769.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. 2025. Mean flows for one-step generative modeling. *Advances in Neural Information Processing Systems*, 38:75460–75482.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. 2024. Better & faster large language models via multi-token prediction. *International Conference on Machine Learning*, 235:15706–15734.

Yi Guo, Wei Wang, Zhihang Yuan, Rong Cao, Kuan Chen, Zhengyang Chen, Yuanyuan Huo, Yang Zhang, Yuping Wang, Shouda Liu, et al. 2025. Splitmeanflow: Interval splitting consistency in few-step generative modeling. *arXiv preprint arXiv:2507.16884*.

Marton Havasi, Brian Karrer, Itai Gat, and Ricky TQ Chen. 2025. Edit flows: Variable length discrete flow matching with sequence-level edit operations. *Advances in Neural Information Processing Systems*, 38:139037–139070.

Satoshi Hayakawa, Yuhta Takida, Masaaki Imaizumi, Hiromi Wakaki, and Yuki Mitsufuji. 2025. Distillation of discrete diffusion through dimensional correlations. *International Conference on Machine Learning*, 267:22259–22297.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851.

Peter Holderrieth, Douglas Chen, Luca Eyring, Ishin Shah, Giri Anantharaman, Yutong He, Zeynep Akata, Tommi Jaakkola, Nicholas Matthew Boffi, and Max Simchowitz. 2026a. Diamond maps: Efficient reward alignment via stochastic flow maps. *International Conference on Machine Learning*, 306:43401–43435.

Peter Holderrieth, Uriel Singer, Tommi Jaakkola, Ricky TQ Chen, Yaron Lipman, and Brian Karrer. 2026b. Glass flows: Efficient inference for reward alignment of flow and diffusion models. *International Conference on Learning Representations*, 2026:106232–106266.

Jiapeng Hu, Weizhi Gao, Zhichao Hou, and Xiaorui Liu. 2026a. Hierarchical multi-scale molecular conformer generation. *International Conference on Learning Representations*, 2026:142010–142039.

Keya Hu, Linlu Qiu, Yiyang Lu, Hanhong Zhao, Tianhong Li, Yoon Kim, Jacob Andreas, and Kaiming He. 2026b. Elf: Embedded language flows. *Advances in Neural Information Processing Systems*.

Wolfgang Kabsch. 1976. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 32(5):922–923.

Olav Kallenberg. 1997. *Foundations of modern probability*. Springer.

Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. 2024. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. *International Conference on Learning Representations*, 2024:44493–44525.

Jaeyeon Kim, Cheuk-Kit Lee, Carles Domingo-Enrich, Yilun Du, Sham Kakade, Timothy Ngotiaoco, Sitan Chen, and Michael Albergo. 2026. Any-order flexible length masked diffusion. *International Conference on Learning Representations*, 2026:40151–40183.

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

Jinsong Li, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, Jiaqi Wang, and Dahua Lin. 2026. Beyond fixed: Training-free variable-length denoising for diffusion large language models. *International Conference on Learning Representations*, 2026:91715–91731.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. 2023. Flow matching for generative modeling. *International Conference on Learning Representations*.

John Nguyen, Marton Havasi, Tariq Berrada, Luke Zettlemoyer, and Ricky TQ Chen. 2025. Oneflow: Concurrent mixed-modal and interleaved generation with edit flows. *arXiv preprint arXiv:2510.03506*.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. 2025. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. *International Conference on Learning Representations*, 2025:64972–65009.

Peter Potaptchik, Adhi Saravanan, Abbas Mammadov, Alvaro Prat, Michael S Albergo, and Yee Whye Teh. 2026a. Meta flow maps enable scalable reward alignment. *International Conference on Machine Learning*, 306:99255–99295.

Peter Potaptchik, Jason Yim, Adhi Saravanan, Peter Holderrieth, Eric Vanden-Eijnden, and Michael S Albergo. 2026b. Discrete flow maps. *arXiv preprint arXiv:2604.09784*.

Kristina Preuer, Philipp Renz, Thomas Unterthiner, Sepp Hochreiter, and Gunter Klambauer. 2018. Fréchet chemnet distance: a metric for generative models for molecules in drug discovery. *Journal of chemical information and modeling*, 58(9):1736–1741.

Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. 2025. Defog: Discrete flow matching for graph generation. *International Conference on Machine Learning*, 267:50269–50326.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Daan Roos, Oscar Davis, Floor Eijkelboom, Michael Bronstein, Max Welling, İsmail İlkan Ceylan, Luca Ambrogioni, and Jan-Willem van de Meent. 2026. Categorical flow maps. *International Conference on Machine Learning*, 306:105618–105642.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. 2025. Align your flow: Scaling continuous-time flow map distillation. *Advances in Neural Information Processing Systems*, 38:146459–146512.

Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184.

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin Chiu, and Volodymyr Kuleshov. 2025. The diffusion duality. *International Conference on Machine Learning*, 267:52584–52619.

Tim Salimans, Thomas Mensink, Jonathan Heek, and Emiel Hoogeboom. 2024. Multistep distillation of diffusion models via moment matching. *Advances in Neural Information Processing Systems*, 37:36046–36070.

Chence Shi, Shitong Luo, Minkai Xu, and Jian Tang. 2021. Learning gradient fields for molecular conformation generation. In *International Conference on Machine Learning*, volume 139, pages 9558–9568.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. 2024. Simplified and generalized masked diffusion for discrete data. *Advances in Neural Information Processing Systems*, 37:103131–103167.

Gregor NC Simm and Jose Miguel Hernandez-Lobato. 2020. A generative model for molecular distance geometry. *International Conference on Machine Learning*, 119:8949–8958.

Daniel GA Smith, Lori A Burns, Andrew C Simmonett, Robert M Parrish, Matthew C Schieber, Raimondas Galvelis, Peter Kraus, Holger Kruse, Roberto Di Remigio, Asem Alenaizan, et al. 2020. Psi4 1.4: Open-source software for high-throughput quantum chemistry. *The Journal of chemical physics*, 152(18):184108.

Yang Song and Prafulla Dhariwal. 2024. Improved techniques for training consistency models. *International Conference on Learning Representations*, 2024:15078–15097.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. 2023. Consistency models. *International Conference on Machine Learning*, 202:32211–32252.

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

Hannes Stark, Bowen Jing, Chenyu Wang, Gabriele Corso, Bonnie Berger, Regina Barzilay, and Tommi Jaakkola. 2024. Dirichlet flow matching with applications to dna sequence design. *International Conference on Machine Learning*, 235:46495–46513.

Sophia Tang, Yinuo Zhang, Alexander Tong, and Pranam Chatterjee. 2025. Gumbel-softmax flow matching with straight-through guidance for controllable biological sequence generation. *arXiv preprint arXiv:2503.17361*.

Sophia Tang, Yuchen Zhu, Molei Tao, and Pranam Chatterjee. 2026. A2d2: Fine-tuning any-length discrete diffusion for adaptive decoding. *arXiv preprint arXiv:2606.13565*.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. 2024. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. 2018. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530.

Zirui Wu, Lin Zheng, Zhihui Xie, Jiacheng Ye, Jiahui Gao, Shansan Gong, Yansong Feng, Zhenguo Li, Wei Bi, Guorui Zhou, et al. 2026. Dreamon: Diffusion language models for code infilling beyond fixed-size canvas. *International Conference on Learning Representations*, 2026:76626–76643.

Minkai Xu, Shitong Luo, Yoshua Bengio, Jian Peng, and Jian Tang. 2021a. Learning neural generative dynamics for molecular conformation generation. *International Conference on Learning Representations*.

Minkai Xu, Wujie Wang, Shitong Luo, Chence Shi, Yoshua Bengio, Rafael Gomez-Bombarelli, and Jian Tang. 2021b. An end-to-end framework for molecular conformation generation via bilevel programming. In *International Conference on Machine Learning*, volume 139, pages 11537–11547.

Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. 2022. Geodiff: A geometric diffusion model for molecular conformation generation. *International Conference on Learning Representations*.

Jingyi Yang, Yuxian Jiang, and Jing Shao. 2026a. ρ-eos: Training-free bidirectional variable-length control for masked diffusion llms. *arXiv preprint arXiv:2601.22527*.

Zhihan Yang, Wei Guo, Shuibai Zhang, Subham Sekhar Sahoo, Yongxin Chen, Arash Vahdat, Morteza Mardani, and John Thickstun. 2026b. Continuous diffusion scales competitively with discrete diffusion for language. *Advances in Neural Information Processing Systems*.

Jaehoon Yoo, Wonjung Kim, and Seunghoon Hong. 2025. Redi: Rectified discrete flow. *Advances in Neural Information Processing Systems*, 38:81651–81683.

Jiying Zhang, Zijing Liu, Yu Wang, Bin Feng, and Yu Li. 2024. Subgdiff: A subgraph diffusion model to improve molecular representation learning. *Advances in Neural Information Processing Systems*, 37:29620–29656.

Kaiwen Zheng, Yongxin Chen, Hanzi Mao, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. 2025. Masked diffusion models are secretly time-agnostic masked models and exploit inaccurate categorical sampling. *International Conference on Learning Representations*, 2025:63186–63227.

Linqi Zhou, Stefano Ermon, and Jiaming Song. 2025. Inductive moment matching. *International Conference on Machine Learning*, 267:78651–78686.

Appendix

A Related Works **17**

B Limitations and Future Work **18**

C Theoretical Proofs **18**

C.1 Notation . . . 18

C.2 Assumptions . . . 19

C.3 Assumptions Hold in Practice . . . 19

C.4 Proof of Proposition C.1 . . . 20

C.5 Proof of Proposition C.2 . . . 21

C.6 Proof of Proposition 4.1 . . . 21

C.7 Proof of Proposition C.3 . . . 23

C.8 Proof of Proposition C.4 . . . 24

D Discrete Sequence Implementation Details **26**

D.1 Local Time Insertions and Denoising . . . 26

D.2 Time Reparameterization . . . 26

D.3 Time Conditioning . . . 27

D.4 Adaptive Loss Weighting . . . 27

D.5 One-Step Sampling . . . 27

E Discrete Graph Implementation Details **28**

E.1 Notation . . . 28

E.2 Per-node Insertion Times and Edge Activation . . . 28

E.3 Node and Edge Interpolants . . . 28

E.4 Compacting the Partially Active Graph . . . 29

E.5 Node and Edge Insertions . . . 29

E.6 Denoising Nodes and Edges . . . 29

F Experiment Details **30**

F.1 Coarse-to-Fine Molecular Conformer Generation . . . 30

F.2 Discrete Molecular Graph Generation . . . 31

F.3 Language Modeling . . . 32

G Algorithms **34**

H Example Generations **36**

A Related Works

Few-Step Generative Models Generation under a flow- or diffusion-based model requires numerical integration of an ODE or SDE, and inference cost scales with the number of function evaluations. A long line of work amortizes this cost by learning a solution operator that maps between two points on the generative trajectory in a single network call. Consistency models (Song et al., 2023; Song and Dhariwal, 2024) train networks to be self-consistent along the probability-flow ODE, and flow maps generalize this by transporting samples between any two times s and t, recovering consistency models at $s = 0$ and the instantaneous velocity as $s \to t$. Recent work develops the theory and training recipes for flow maps (Boffi et al., 2025b; Geng et al., 2025; Frans et al., 2025; Guo et al., 2025; Sabour et al., 2025), with parallel efforts on few-step discrete language generation (Hu et al., 2026b; Deschenaux and Gulcehre, 2025; Gloeckle et al., 2024; Hayakawa et al., 2025; Yoo et al., 2025; Roos et al., 2026; Lee et al., 2026; Potaptchik et al., 2026b).

Continuous Flows for Discrete Generation Discrete diffusion (Shi et al., 2024; Sahoo et al., 2024; Ou et al., 2025; Zheng et al., 2025) needs many function evaluations for accurate generation, motivating extensions of flow maps to discrete domains where standard regression losses are unstable. Continuous-space parameterizations for discrete generation operate either in simplex space (Stark et al., 2024; Davis et al., 2024; Tang et al., 2025) or in continuous embedding spaces (Cheng et al., 2025; Chen et al., 2026; Dieleman et al., 2022; Hu et al., 2026b; Deschenaux and Gulcehre, 2026; Yang et al., 2026b). Building on these, FMLMs (Lee et al., 2026) and DFMs (Potaptchik et al., 2026b) distill a continuous flow over simplex-embedded tokens into a few-step generator trained with simplex-aware cross-entropy, while CFM (Roos et al., 2026) self-distills a flow-matching model constrained to the simplex. Together, these show the flow-map recipe transfers to categorical data once the loss matches the simplex geometry.

Stochastic Flow Maps Standard flow maps are deterministic, but posterior sampling, reward alignment, and inverse problems require many independent endpoints from a single intermediate state. Flow Map Matching (Boffi et al., 2025a) builds flow maps for stochastic interpolants that generalize consistency models. Diamond Maps (Holderrieth et al., 2026a) amortize many SDE steps into one call while preserving the noise injection needed for value estimation, distilled from GLASS Flows (Holderrieth et al., 2026b). Meta Flow Maps (Potaptchik et al., 2026a) condition on an intermediate pair $(\boldsymbol{x}, t)$ to learn a family of conditional flow maps generating the posterior $p_{u|t}(\cdot \mid \boldsymbol{x})$, recovering the clean posterior $p_{1|t}(\cdot \mid \boldsymbol{x})$ at $u = 1$.

Variable-Length Generation Most non-autoregressive models, including masked diffusion and standard discrete flows, operate on a fixed-length canvas that forces a length commitment upfront and wastes compute on padding. A growing body of work removes this restriction. FlexMDMs (Kim et al., 2026) extend stochastic interpolants (Albergo et al., 2025; Albergo and Vanden-Eijnden, 2023) to build sequences by inserting and then unmasking tokens, and can retrofit pretrained models like LLaDA-8B. Edit Flows (Havasi et al., 2025) model generation as a continuous-time Markov chain over insertion, deletion, and substitution edits, and OneFlow (Nguyen et al., 2025) extends this to interleaved multimodal text-and-image outputs. Within masked diffusion, several methods adjust length via the model's own predictions, including block diffusion (Arriola et al., 2025), DAEDAL (Li et al., 2026), ρ-EOS (Yang et al., 2026a), DreamOn (Wu et al., 2026), and Deletion-Insertion Diffusion (Ding et al., 2026). A2D2 (Tang et al., 2026) formulates any-length generation as a controlled continuous-time Markov chain and jointly optimizes insertion and unmasking policies for reward-guided fine-tuning. These works establish variable-length generation but remain outside the flow-map paradigm, motivating how to combine length flexibility with few-step flow-map efficiency.

Other work changes dimension during generation directly. The closest to our setting is trans-dimensional jump diffusion (Campbell et al., 2023), which generates state values and dimension jointly rather than sampling the size first. Its forward process adds Gaussian noise and deletes one component at a time at a user-defined rate. Its generative process is the time reversal of this forward process, a stochastic jump diffusion that learns the score, the jump rate, and the distribution of each inserted component. Like our construction, this generative process only adds components, and like our discrete model, it requires a maximum dimension. The two approaches differ in how they are built. Campbell et al. (2023) obtain insertions from a time-reversal argument, and each jump adds one component to a stochastic trajectory. We instead define an expanding interpolant, transport the state

deterministically between insertions, and learn an expand operator that can insert many coordinates in one step. Branching Flows (Billera et al., 2025) evolves a population of elements over a forest of binary trees, with learned split and deletion rates. FlowMol3 (Dunn and Koes, 2026) instead keeps a fixed canvas by adding fake atoms in a fixed-dimensional flow matching model, and removes them after generation. Unlike these methods, EFM learns two-time flow maps across dimensions for few-step sampling.

B Limitations and Future Work

Scope Our construction only increases dimensionality. The expand operator is injective, so $d(t) \geq d(s)$, and it cannot delete a coordinate. A deleted coordinate would have no regression target under our losses and no image under the direct $s \to t$ map, which breaks two-time consistency. The discrete model also requires an upper bound L on length.

Length calibration Generated lengths on unpacked LM1B match held-out lengths only approximately (34.1 ± 16.4 versus 30.5 ± 14.8 tokens with a max token length of 128, Fig 3). Further hyperparameter tuning could enhance the accurate learning of the joint size-content distribution.

Evaluation and cost We evaluate on GEOM-QM9 (up to 29 atoms), GEOM-Drugs (up to 181 atoms), QM9 graphs (up to 9 heavy atoms), and LM1B (up to 128 tokens), with a single run per configuration. Per-coordinate local times and the larger space of interpolants per data sample suggest that larger dimensionalities will need further tuning, which we could not fully explore because of the computational expense. The expand operator adds 9.3% to training time per step for EFlow relative to FLM and about 18% for EFM relative to FMLM (App F.3).

Future work The general construction of several components of our framework also suggests future extensions. For instance, the conditional noise distribution is not restricted to be defined as Gaussian noise and can be an additional learnable parameter, where the mean and covariance from which the inserted dimensions are sampled are parameterized. Furthermore, different interpolants that are *conditioned* on the insertion time, or enforcing some structure to the ordering of insertion times, remain promising directions to be explored. Finally, while we restrict ourselves to interpolants of *increasing dimensionality*, we note that decreasing dimensionality along the interpolant can be considered as an operation in the PDMP construction.

C Theoretical Proofs

C.1 Notation

State spaces and time variables We write $s, u, t \in [0,1]$ for time coordinates and $d(t) : [0,1] \to \mathbb{N}$ for a non-decreasing dimension schedule. In the continuous setting, $\mathbb{R}^{d(t)}$ is the state space at time t and, in the discrete setting, $\mathbb{R}^{d(t)\times V}$ is the space of length-$d(t)$ sequences whose tokens are V-dimensional latent vectors that resolve onto the simplex $\Delta^{V-1} \subset \mathbb{R}^V$. The PDMP of Section 3 lives on the disjoint union $\mathcal{M} := \bigsqcup_t \mathbb{R}^{d(t)}$. For each coordinate i, t_i^{ins} is its insertion time and $t_i = (t - t_i^{\text{ins}})/(1 - t_i^{\text{ins}})$ its local time. The vector of local times is denoted $\boldsymbol{t}_{\text{local}}$.

States and distributions $\boldsymbol{x}_t$ denotes the state at time t, a latent vector in $\mathbb{R}^{d(t)}$ (continuous) or a sequence of latent vectors in $\mathbb{R}^{d(t)\times V}$ (discrete). We write $\boldsymbol{x}_0 \sim p_0$ for a prior sample, $\boldsymbol{x}_1 \sim p_1$ for a clean data sample, and $\boldsymbol{x}_1^{(t)}$ for the data sample restricted to the coordinates present at time t. The marginal law is $p_t \in \mathcal{P}(\mathbb{R}^{d(t)})$ and the conditional law of the jump is $p_{t|s}(\cdot \mid \boldsymbol{x}_s)$. Augmented noise $\boldsymbol{\epsilon} \sim p_{\epsilon|s,t}$ lifts the dimension from $d(s)$ to $d(t)$, giving the augmented state $\boldsymbol{x}_s^{\epsilon} := \mathcal{E}_{s,t}(\boldsymbol{x}_s) = \mathcal{E}_{s,t}(\boldsymbol{x}_s, \boldsymbol{\epsilon})$.

Operators $\mathcal{E}_{s,t}$ is the expand operator, $X_{s,t}$ the transport map in the augmented space, and $\Phi_{s,t} := X_{s,t} \circ \mathcal{E}_{s,t}$ the full expanding flow map. Transport is parameterized by the average velocity $v_{s,t}$ in the continuous setting and by the mean denoiser $\psi_{s,t}$ (valued on the simplex) in the discrete setting. On the diagonal $t = s$, b_t is the instantaneous velocity and D_t the instantaneous denoiser. In the

discrete setting, α_t is the insertion schedule with $\Pr[t_i^{\text{ins}} \le t] = \alpha_t$, $\mathcal{I}_{s,t}(\boldsymbol{x}_s)[i]$ the per-gap insertion expectation, and λ_t the insertion intensity of the PDMP. Hats denote network-parameterized objects ($\hat{v}_{s,t}$, $\hat{\psi}_{s,t}$, $\hat{\mathcal{E}}_{s,t}$, $\hat{\mathcal{I}}_{s,t}$) and $\mathrm{sg}(\cdot)$ the stop-gradient operator used in the consistency objectives.

C.2 Assumptions

Assumptions on the Expand Operator The expand operator $\mathcal{E}_{s,t}$ obeys the following structural properties:

Assumption A1 (Fixed-noise differentiability)**.** *Once the augmented noise $\boldsymbol{\epsilon}$ is sampled, $\mathcal{E}_{s,t}(\boldsymbol{x}, \boldsymbol{\epsilon})$ depends differentiably on $\boldsymbol{x}$, and its $\boldsymbol{x}$-derivative is the embedding matrix $E \in \mathbb{R}^{d(t)\times d(s)}$ that places existing coordinates at their target positions and fills the rest with the entries of $\boldsymbol{\epsilon}$. In particular, $\partial_t \mathcal{E}_{s,t}(\boldsymbol{x}, \boldsymbol{\epsilon}) = 0$ and $\partial_s \mathcal{E}_{s,t}(\boldsymbol{x}, \boldsymbol{\epsilon}) = 0$ at fixed $\boldsymbol{\epsilon}$ and fixed placement, and $\nabla_{\boldsymbol{x}} \mathcal{E}_{s,t}(\boldsymbol{x}, \boldsymbol{\epsilon}) = E$.*

Assumption A2 (Composability)**.** *For $s \le u \le t$, the placements compose: $\mathcal{E}_{s,t} = \mathcal{E}_{u,t} \circ \mathcal{E}_{s,u}$ when the noise samples are drawn consistently from the disintegration $p_{\epsilon|s,t} = p_{\epsilon|u,t} \otimes p_{\epsilon|s,u}$.*

Assumption A3 (Inactive coordinates are unchanged)**.** *For $s \le u$, the augmented-space flow $X_{s,u}$ on $\mathbb{R}^{d(t)}$ does not change the $d(t) - d(u)$ coordinates that are scheduled to be inserted only at times $\ge u$. Equivalently, $X^{(t)}_{s,u} \circ \mathcal{E}_{u,t} = \mathcal{E}_{u,t} \circ X_{s,u}$ on the image of $\mathcal{E}_{s,u}$.*

Assumption A4 (Active-inactive velocity definition)**.** *On the image of $\mathcal{E}_{s,t}$, the diagonal velocity has no component along inactive coordinates, such that there exists $\tilde{v}_{s,s} : \mathbb{R}^{d(s)} \to \mathbb{R}^{d(s)}$ such that:*

$$v_{s,s}(\mathcal{E}_{s,t}(\boldsymbol{x})) = E\tilde{v}_{s,s}(\boldsymbol{x}), \quad E := \nabla_{\boldsymbol{x}} \mathcal{E}_{s,t}(\boldsymbol{x}, \boldsymbol{\epsilon}) \in \mathbb{R}^{d(t)\times d(s)} \tag{25}$$

Intuitively, A3 states that the flow does not transform coordinates before they are inserted. A4 extends A3 by stating that if inactive coordinates do not move under the augmented flow, then their instantaneous velocity vanishes, so $v_{s,s}(\mathcal{E}_{s,t}(\boldsymbol{x}))$ is supported on the active block (i.e., in the column space of E). This means that it can be reduced back to a $d(s)$-dimensional velocity $\tilde{v}_{s,s} : \mathbb{R}^{d(s)} \to \mathbb{R}^{d(s)}$.

C.3 Assumptions Hold in Practice

The assumptions on the expand operator split into two groups. A1 (fixed-noise differentiability) and A2 (composability) constrain only the placement map of $\mathcal{E}_{s,t}$ and hold for any learned velocity, while A3 (inactive coordinates are unchanged) and A4 (active-inactive split) constrain the learned dynamics and are enforced by the local-time velocity of (8). We verify both groups hold by construction in each experiment.

A1 and A2 are automatically enforced by the expand operation. In every experiment, $\mathcal{E}_{s,t}$ only decides which slot each coordinate goes in. It copies the existing coordinates of $\boldsymbol{x}_s$ into their slots and fills the new ones with the noise $\boldsymbol{\epsilon}$ (or a noisy copy of the parent coordinate for child expansion). Once the noise and the slots are fixed, the output is affine in $\boldsymbol{x}_s$ with a fixed derivative E and no dependence on time, giving A1. For concatenation and gap-wise insertion, E is a $0/1$ slot-selection matrix.

Composability (A2) then says that inserting in two steps ($s \to u \to t$) lands each coordinate in the same slot as inserting in one step ($s \to t$), which holds because concatenating slots is associative. The only condition is that the augmented noise be drawn consistently across the two paths. For a fixed schedule, each token's insertion time satisfies $\Pr[t_i^{\text{ins}} \le t] = \alpha_t$, which satisfies the disintegration $p_{\epsilon|s,t} = p_{\epsilon|u,t} \otimes p_{\epsilon|s,u}$. When the per-gap allocation is learned (language), the schedule factor in (18) and the two-time insertion loss constrain the head to respect this disintegration, so composability is enforced by the fixed schedule.

A3 and A4 are enforced by the velocity mask. The velocity field sets $b_t(\boldsymbol{x}_t^\epsilon)[i] = \mathbf{0}$ for every uninserted coordinate ($t_i^{\text{ins}} > t$), so uninserted coordinates never move (A3) and $v_{s,s}(\mathcal{E}_{s,t}(\boldsymbol{x}))$ is supported on the active block spanned by E (A4). This mask is applied at the parameterization level and does not rely on training. In conformer generation, the heavy-atom backbone is denoised and frozen before hydrogens are inserted, which is A3 applied to resolved atoms. In discrete graphs, uninserted node and edge coordinates stay empty until inserted. In language, a position enters the mean-denoiser loss only after being inserted, so the denoiser emits no gradient for empty positions.

C.4 Proof of Proposition C.1

Proposition C.1 (Expanding Flows are Piecewise-Deterministic Markov Processes)**.** *Let $\mathcal{M} := \bigsqcup_{t\in[0,1]} \mathbb{R}^{d(t)}$ be the state space and $(\boldsymbol{x}_t)_{t\in[0,1]}$ the expanding generative flow of Def. 3.1, with velocity b_t in the augmented space. Writing $\mathcal{E}_t := \lim_{s\to t} \mathcal{E}_{s,t}$ and $p_{\epsilon|t} := \lim_{s\to t} p_{\epsilon|s,t}$ for the instantaneous insertion map and noise law at time t, the augmented process $\boldsymbol{S}_t := (\boldsymbol{x}_t, \boldsymbol{t}_{local})$ is a PDMP with extended generator acting on $\boldsymbol{x}$-functions given by (11), where λ_t is the insertion intensity and the jump kernel is the pushforward $(\mathcal{E}_t(\boldsymbol{x}, \cdot))_{\#} p_{\epsilon|t}$.*

Proof. A piecewise-deterministic Markov process (PDMP) on $\mathcal{M}$ is characterized by (i) a deterministic flow $\dot{\boldsymbol{x}}_\tau$, (ii) a jump intensity $\lambda_t \geq 0$, and (iii) a transition kernel Q_t of the jump process. The generator of a PDMP is given by:

$$\mathcal{L}_t f = \nabla f \cdot b_t + \lambda_t \int (f(\boldsymbol{y}) - f(\boldsymbol{x}))\, Q_t(\boldsymbol{x}, d\boldsymbol{y}) \tag{26}$$

Because the jump intensity and transport depend on how far each coordinate has been denoised, we track the full state $\boldsymbol{S}_t := (\boldsymbol{x}_t, \boldsymbol{t}_{\text{local}})$, where $\boldsymbol{t}_{\text{local}} = \{t_i\}$ collects the local time coordinates $t_i = (t - t_i^{\text{ins}})/(1 - t_i^{\text{ins}})$ of the inserted coordinates (equivalently the insertion times, by the bijection at fixed t). We write each component of the expanding generative flow as a component of the PDMP generator.

Step 1: Deterministic Flow. Given an ordered set of insertion times $\{t_k^{\text{ins}}\}$, the time interval between two consecutive insertion times $(t_k^{\text{ins}}, t_{k+1}^{\text{ins}})$ has fixed dimension $d(t)$ and following Def 3.1, the state evolves via an augmented-space ODE $\dot{\boldsymbol{x}}_\tau = b_\tau(\boldsymbol{x}_\tau)$ on the fixed-dimensional space $\boldsymbol{x}_\tau \in \mathbb{R}^{d(t_k^{\text{ins}+})}$, where the superscript $+$ denotes the right limit, i.e. the dimension immediately after the kth insertion. For any test function $f \in C_b^1$ with a bounded, continuous first derivative, the chain rule yields:

$$\tfrac{d}{dt} f(\boldsymbol{x}_t) = \dot{\boldsymbol{x}}_t \cdot \nabla f(\boldsymbol{x}_t) = b_t(\boldsymbol{x}_t) \cdot \nabla f(\boldsymbol{x}_t) \tag{27}$$

which is the transport term in (26). By A3, $b_\tau(\boldsymbol{x})[i] = 0$ for every uninserted coordinate i and all $\boldsymbol{x}$, so the velocity field maps into the active subspace and its flow keeps the uninserted coordinates fixed for the whole interval. Hence $b_t \cdot \nabla f$ is supported on the inserted coordinates and the flow stays bound to $\mathbb{R}^{d(t_k^{\text{ins}+})}$.

Step 2: Jump Between Dimensions. At each insertion time t_k^{ins}, the state makes a discontinuous jump from $\boldsymbol{x} \in \mathbb{R}^{d(t_{k-}^{\text{ins}})}$ to the augmented state $\boldsymbol{x}^\epsilon = \mathcal{E}_{t_k^{\text{ins}}}(\boldsymbol{x}, \boldsymbol{\epsilon}) \in \mathbb{R}^{d(t_k^{\text{ins}+})}$ where $\boldsymbol{\epsilon} \sim p_{\epsilon|t_k^{\text{ins}}}$. Since the expanding operator $\mathcal{E}_{t_k^{\text{ins}}}(\boldsymbol{x}, \boldsymbol{\epsilon})$ depends only on the current state $\boldsymbol{x}$ and freshly sampled noise $\boldsymbol{\epsilon}$, the jump is Markov.

By A1, for fixed noise $\mathcal{E}_t(\cdot, \boldsymbol{\epsilon})$ is affine in both $\boldsymbol{x}$ and $\boldsymbol{\epsilon}$, so as a map of the pair $(\boldsymbol{x}, \boldsymbol{\epsilon})$ it is continuous and Borel measurable. This guarantees that the preimage $\{\boldsymbol{\epsilon} : \mathcal{E}_t(\boldsymbol{x}, \boldsymbol{\epsilon}) \in A\}$ is measurable and the transition kernel $Q_t(\boldsymbol{x}, A)$, defined as the probability of the expanded state starting from $\boldsymbol{x}$ landing in a measurable set A, is well-defined. Given $\boldsymbol{x}$, the stochasticity is dependent on $\boldsymbol{\epsilon}$, so the probability of landing in A is the probability that $\boldsymbol{\epsilon}$ falls in the preimage set, equal to the pushforward measure:

$$Q_t(\boldsymbol{x}, A) = p_{\epsilon|t}(\{\boldsymbol{\epsilon} : \mathcal{E}_t(\boldsymbol{x}, \boldsymbol{\epsilon}) \in A\}) = \left((\mathcal{E}_t(\boldsymbol{x}, \cdot))_{\#}\, p_{\epsilon|t}\right)(A) \tag{28}$$

so $Q_t(\boldsymbol{x}, \cdot)$ is the law of $\mathcal{E}_t(\boldsymbol{x}, \boldsymbol{\epsilon})$, a Markov kernel. The generator of a Markov jump process is the expected instantaneous change in the test function with jump intensity λ_t:

$$\lambda_t \int (f(\boldsymbol{y}) - f(\boldsymbol{x}))\, Q_t(\boldsymbol{x}, d\boldsymbol{y}) = \lambda_t \int (f(\mathcal{E}_t(\boldsymbol{x}, \boldsymbol{\epsilon})) - f(\boldsymbol{x})) p_{\epsilon|t}(d\boldsymbol{\epsilon}) \tag{29}$$

where we substitute the post-jump state with the expanded state $\mathcal{E}_t(\boldsymbol{x}, \boldsymbol{\epsilon})$ and integrate over $\boldsymbol{\epsilon} \sim p_{\epsilon|t}$.

Step 3: Markov Property. The post-jump state depends only on $(\boldsymbol{x}, \boldsymbol{\epsilon})$, with $\boldsymbol{\epsilon}$ drawn independently of the past. Between jumps the local times advance deterministically by $\dot{t}_i = (1 - t_i^{\text{ins}})^{-1}$ and a newly inserted coordinate enters at $t_i = 0$, so $\boldsymbol{t}_{\text{local}}$ is determined by the state.

Since between jumps $\boldsymbol{S}_t$ evolves deterministically and at a jump depends only on $(\boldsymbol{x}, \boldsymbol{\epsilon})$, it depends on the past only through $\boldsymbol{S}_{t^-}$, so $(\boldsymbol{S}_t)_{t\in[0,1]}$ is Markov. The state process $(\boldsymbol{x}_t)_{t\in[0,1]}$ is not Markov on its own, as its intensity and transport require $\boldsymbol{t}_{\text{local}}$. By Prop C.2, the augmented state has conditional law $p^{\epsilon}_{s,t}(\cdot \mid \boldsymbol{x}_s) = (\mathcal{E}_{s,t}(\boldsymbol{x}_s, \cdot))_{\#} p_{\epsilon|s,t}$, and taking $s \uparrow t$ and comparing with the kernel above gives $Q_t(\boldsymbol{x}_s, \cdot) = p_{t|s}(\cdot \mid \boldsymbol{x}_s)$, where the disintegration $p_{\epsilon|s,t} = p_{\epsilon|u,t} \otimes p_{\epsilon|s,u}$ is A2.

Step 4: Jump Intensity in Discrete Case. A coordinate inserted with schedule $\Pr[t_i^{\text{ins}} \le t] = \alpha_t$ has survival $1 - \alpha_t$ and hazard $-\frac{d}{dt}\log(1-\alpha_t) = \dot{\alpha}_t/(1-\alpha_t)$ (the insertion rate for each expected uninserted token) so summing over gaps i gives $\lambda_t(\boldsymbol{x}) = \frac{\dot{\alpha}_t}{1-\alpha_t}\sum_i \mathcal{I}_t(\boldsymbol{x})[i]$. Since each jump raises the dimension by at least one and $d(1) < \infty$, there are finitely many jumps on $[0,1]$ and the process does not explode. Combining the generator components from the deterministic flow, insertion jump, and jump intensity yields the generator for the PDMP of the expanding flow in (11). □

Corollary C.1 (Change-of-Variables Formula)**.** *Given that $X_{s,t}$ is the time-t map of an ODE in the augmented space $\frac{d}{d\tau}\boldsymbol{x}^{\epsilon}_{\tau} = b_\tau(\boldsymbol{x}^{\epsilon}_{\tau})$, the marginal density satisfies the instantaneous change-of-variables formula:*

$$\forall s \le t, \quad \log p_t(X_{s,t}(\boldsymbol{x}^{\epsilon}_s)) = \log p_s(\boldsymbol{x}^{\epsilon}_s) - \int_s^t \nabla \cdot b_\tau(X_{s,\tau}(\boldsymbol{x}^{\epsilon}_s))d\tau, \quad \nabla \cdot b_\tau = \sum_{i:t_i^{ins}\le\tau} \partial_{\boldsymbol{x}^i} b^i_\tau \tag{30}$$

The divergence sums only over active coordinates, so non-inserted coordinates contribute nothing until their insertion time.

C.5 Proof of Proposition C.2

Proposition C.2 (Expand Operator is Pushforward of Noise Law)**.** *Fix an expand operator $\mathcal{E}_{s,t} : \mathbb{R}^{d(s)} \times \mathbb{R}^{d(t)-d(s)} \to \mathbb{R}^{d(t)}$ that places source and noise coordinates according to one of the instantiations. Then, the conditional law of the augmented state $\boldsymbol{x}^{\epsilon}_s$ is*

$$p^{\epsilon}_{s,t}(\cdot \mid \boldsymbol{x}_s) = (\mathcal{E}_{s,t}(\boldsymbol{x}_s, \cdot))_{\#} p_{\epsilon|s,t}(\cdot \mid \boldsymbol{x}_s) \tag{31}$$

As $s \uparrow t$, this law is the conditional law $p_{t|s}(\cdot \mid \boldsymbol{x}_s)$ of the jump.

Proof. Fix $\boldsymbol{x}_s \in \mathbb{R}^{d(s)}$. The map $\mathcal{E}_{s,t}(\boldsymbol{x}_s, \cdot)$ is measurable in the noise alone, since concatenation, positional insertion, and child expansion are all affine in the noise. By definition, $p^{\epsilon}_{s,t}(\cdot \mid \boldsymbol{x}_s)$ is the law of $\mathcal{E}_{s,t}(\boldsymbol{x}_s, \boldsymbol{\epsilon})$ with $\boldsymbol{\epsilon} \sim p_{\epsilon|s,t}(\cdot \mid \boldsymbol{x}_s)$. For every Borel set $A \subseteq \mathbb{R}^{d(t)}$,

$$p^{\epsilon}_{s,t}(A \mid \boldsymbol{x}_s) = \Pr\left[\boldsymbol{\epsilon} \in \mathcal{E}_{s,t}(\boldsymbol{x}_s, \cdot)^{-1}(A) \;\middle|\; \boldsymbol{x}_s\right] = \left(\mathcal{E}_{s,t}(\boldsymbol{x}_s, \cdot)\right)_{\#} p_{\epsilon|s,t}(\cdot \mid \boldsymbol{x}_s)(A), \tag{32}$$

which is (31). As $s \uparrow t$, the transport over $[s,t]$ vanishes, so the augmented state and $\boldsymbol{x}_t$ have the same conditional law. □

C.6 Proof of Proposition 4.1

Setup Let $\boldsymbol{x}^{\epsilon}_s = \mathcal{E}_{s,t}(\boldsymbol{x}) \in \mathbb{R}^{d(t)}$ and $X_{s,t} : \mathbb{R}^{d(t)} \to \mathbb{R}^{d(t)}$ be the time-t flow map of the augmented-space ODE $\dot{\boldsymbol{z}}_\tau = b_\tau(\boldsymbol{z}_\tau)$ on $\mathbb{R}^{d(t)}$ with $\boldsymbol{z}_s = \boldsymbol{x}^{\epsilon}_s$. By the proofs in Boffi et al. (2025b), $X_{s,t}$ satisfies the standard fixed-dimensional flow map identities. Here, we lift these through $\mathcal{E}_{s,t}$ to $\Phi_{s,t} = X_{s,t} \circ \mathcal{E}_{s,t}$.

Proposition 4.1 (Expanding Flow Map Identities)**.** *Let $\boldsymbol{x}_s^\epsilon := \mathcal{E}_{s,t}(\boldsymbol{x}_s)$. Then, the expanding flow map $\Phi_{s,t}(\boldsymbol{x}_s)$ satisfies the following consistency conditions:*

$$\begin{aligned} \textit{Lagrangian:} \quad & \partial_t \Phi_{s,t}(\boldsymbol{x}) = v_{t,t}(\Phi_{s,t}(\boldsymbol{x})) && \text{(15a)} \\ \textit{Eulerian:} \quad & \partial_s \Phi_{s,t}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} \Phi_{s,t}(\boldsymbol{x}) \tilde{v}_{s,s}(\boldsymbol{x}) = 0 && \text{(15b)} \\ \textit{Semigroup:} \quad & \Phi_{u,t}(\Phi_{s,u}(\boldsymbol{x})) = \Phi_{s,t}(\boldsymbol{x}) && \text{(15c)} \end{aligned}$$

where $\tilde{v}_{s,s} : \mathbb{R}^{d(s)} \to \mathbb{R}^{d(s)}$ is the unlifted diagonal velocity of A4, satisfying $v_{s,s}(\mathcal{E}_{s,t}(\boldsymbol{x})) = E\tilde{v}_{s,s}(\boldsymbol{x})$. $\Phi_{s,t}(\boldsymbol{x}) := X_{s,t}(\mathcal{E}_{s,t}(\boldsymbol{x})) = \mathcal{E}_{s,t}(\boldsymbol{x}) + (t-s)v_{s,t}(\mathcal{E}_{s,t}(\boldsymbol{x}))$ is the residual form of the expanding flow map in the continuous state space.

Proof. Throughout this proof, we fix the augmented noise $\boldsymbol{\epsilon}$ and simplify notation by writing $\mathcal{E}_{s,t}(\boldsymbol{x}) \equiv \mathcal{E}_{s,t}(\boldsymbol{x}, \boldsymbol{\epsilon})$. By A1, the placement is held fixed, so the Lagrangian and Eulerian identities are local statements valid on any interval of times over which the dimension schedule $d(\cdot)$ is constant, i.e., between consecutive insertion times. The semigroup identity requires no differentiability and holds for all $0 \le s \le u \le t \le 1$. We prove each identity one by one.

Lagrangian Identity By the chain rule applied to $\Phi_{s,t}(\boldsymbol{x}) = X_{s,t}(\mathcal{E}_{s,t}(\boldsymbol{x}))$, we have:

$$\partial_t \Phi_{s,t}(\boldsymbol{x}) = \partial_t X_{s,t}(\boldsymbol{y})\big|_{\boldsymbol{y}=\mathcal{E}_{s,t}(\boldsymbol{x})} + \nabla_{\boldsymbol{y}} X_{s,t}(\boldsymbol{y})\big|_{\boldsymbol{y}=\mathcal{E}_{s,t}(\boldsymbol{x})} \cdot \underbrace{\partial_t \mathcal{E}_{s,t}(\boldsymbol{x}, \boldsymbol{\epsilon})}_{=0 \text{ by A1}} = \partial_t X_{s,t}(\boldsymbol{y})\big|_{\boldsymbol{y}=\mathcal{E}_{s,t}(\boldsymbol{x})} \tag{33}$$

The standard Lagrangian condition (3) in the fixed-dimensional augmented space $\mathbb{R}^{d(t)}$, $\partial_t X_{s,t}(\boldsymbol{y}) = v_{t,t}(X_{s,t}(\boldsymbol{y}))$, evaluated at $\boldsymbol{y} = \mathcal{E}_{s,t}(\boldsymbol{x})$, gives:

$$\partial_t \Phi_{s,t}(\boldsymbol{x}) = v_{t,t}\big(X_{s,t}(\mathcal{E}_{s,t}(\boldsymbol{x}))\big) = v_{t,t}(\Phi_{s,t}(\boldsymbol{x})), \tag{34}$$

which is the Lagrangian identity in (15).

Eulerian Identity By A1, $\partial_s \mathcal{E}_{s,t}(\boldsymbol{x}, \boldsymbol{\epsilon}) = 0$ at fixed $\boldsymbol{\epsilon}$, so the chain rule in s gives:

$$\begin{aligned} \partial_s \Phi_{s,t}(\boldsymbol{x}) &= \partial_s X_{s,t}(\boldsymbol{y})\big|_{\boldsymbol{y}=\mathcal{E}_{s,t}(\boldsymbol{x})} + \nabla_{\boldsymbol{y}} X_{s,t}(\boldsymbol{y})\big|_{\boldsymbol{y}=\mathcal{E}_{s,t}(\boldsymbol{x})} \cdot \underbrace{\partial_s \mathcal{E}_{s,t}(\boldsymbol{x}, \boldsymbol{\epsilon})}_{=0} \\ &= \partial_s X_{s,t}(\boldsymbol{y})\big|_{\boldsymbol{y}=\mathcal{E}_{s,t}(\boldsymbol{x})} \end{aligned} \tag{35}$$

Similarly, since $\nabla_{\boldsymbol{x}} \mathcal{E}_{s,t}(\boldsymbol{x}, \boldsymbol{\epsilon}) = E \in \mathbb{R}^{d(t) \times d(s)}$ by A1, the chain rule in $\boldsymbol{x}$ gives:

$$\nabla_{\boldsymbol{x}} \Phi_{s,t}(\boldsymbol{x}) = \nabla_{\boldsymbol{y}} X_{s,t}(\boldsymbol{y})\big|_{\boldsymbol{y}=\mathcal{E}_{s,t}(\boldsymbol{x})} E \ \in \ \mathbb{R}^{d(t) \times d(s)} \tag{36}$$

The standard Eulerian condition (3) on the augmented space $\mathbb{R}^{d(t)}$, evaluated at $\boldsymbol{y} = \mathcal{E}_{s,t}(\boldsymbol{x})$, is given by:

$$\partial_s X_{s,t}(\boldsymbol{y}) + \nabla_{\boldsymbol{y}} X_{s,t}(\boldsymbol{y}) v_{s,s}(\boldsymbol{y}) = 0 \tag{37}$$

By A4, on the image of $\mathcal{E}_{s,t}$ the diagonal velocity satisfies $v_{s,s}(\mathcal{E}_{s,t}(\boldsymbol{x})) = E\tilde{v}_{s,s}(\boldsymbol{x})$ with $\tilde{v}_{s,s}(\boldsymbol{x}) \in \mathbb{R}^{d(s)}$. Substituting into the second term of (37) and using (36), we get:

$$\nabla_{\boldsymbol{y}} X_{s,t}(\boldsymbol{y})\big|_{\boldsymbol{y}=\mathcal{E}_{s,t}(\boldsymbol{x})} v_{s,s}(\mathcal{E}_{s,t}(\boldsymbol{x})) = \nabla_{\boldsymbol{y}} X_{s,t}(\boldsymbol{y})\big|_{\boldsymbol{y}=\mathcal{E}_{s,t}(\boldsymbol{x})} E\tilde{v}_{s,s}(\boldsymbol{x}) \overset{(36)}{=} \nabla_{\boldsymbol{x}} \Phi_{s,t}(\boldsymbol{x}) \tilde{v}_{s,s}(\boldsymbol{x}) \tag{38}$$

Combining with (35) yields:

$$\partial_s \Phi_{s,t}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} \Phi_{s,t}(\boldsymbol{x}) \tilde{v}_{s,s}(\boldsymbol{x}) = 0, \tag{39}$$

which is the Eulerian identity in (15). Both terms are vectors in $\mathbb{R}^{d(t)}$, since $\nabla_{\boldsymbol{x}} \Phi_{s,t}(\boldsymbol{x}) \in \mathbb{R}^{d(t) \times d(s)}$ multiplies the vector $\tilde{v}_{s,s}(\boldsymbol{x}) \in \mathbb{R}^{d(s)}$ to give a vector in $\mathbb{R}^{d(t)}$, matching $\partial_s \Phi_{s,t}(\boldsymbol{x})$.

Semigroup Identity Fix $0 \le s \le u \le t \le 1$. The maps involved act between spaces of different dimensions:

$$\Phi_{s,t} : \mathbb{R}^{d(s)} \to \mathbb{R}^{d(t)}, \quad \Phi_{s,u} : \mathbb{R}^{d(s)} \to \mathbb{R}^{d(u)}, \quad \Phi_{u,t} : \mathbb{R}^{d(u)} \to \mathbb{R}^{d(t)}, \tag{40}$$

so the claim is that expanding and transporting from $d(s)$ to $d(t)$ in one step agrees with doing so in two steps through $d(u)$.

Let $X^{(t)}_{s,u} : \mathbb{R}^{d(t)} \to \mathbb{R}^{d(t)}$ denote the transport map on the augmented space $\mathbb{R}^{d(t)}$ over $[s, u]$. By (8) its velocity vanishes on coordinates scheduled for insertion at times $\ge u$, so $X^{(t)}_{s,u}$ acts as $X_{s,u}$ on the coordinates active by time u and as the identity on the remaining $d(t) - d(u)$ coordinates. Since $X^{(t)}_{s,u}$, $X_{u,t}$, and $X_{s,t}$ are all flow maps of the same ODE on the fixed space $\mathbb{R}^{d(t)}$, the standard semigroup condition (3) gives

$$\Phi_{s,t}(\boldsymbol{x}) = X_{s,t}(\mathcal{E}_{s,t}(\boldsymbol{x})) = X_{u,t}\big(X^{(t)}_{s,u}(\mathcal{E}_{s,t}(\boldsymbol{x}))\big). \tag{41}$$

By A2, the placements compose, $\mathcal{E}_{s,t} = \mathcal{E}_{u,t} \circ \mathcal{E}_{s,u}$, and by A3 the augmented-space flow leaves the coordinates inserted at times $\ge u$ unchanged, so on the image of $\mathcal{E}_{s,u}$, we have:

$$X^{(t)}_{s,u} \circ \mathcal{E}_{u,t} = \mathcal{E}_{u,t} \circ X_{s,u} \tag{42}$$

Substituting A2 and then (42) into (41), we get:

$$\begin{aligned} \Phi_{s,t}(\boldsymbol{x}) &= X_{u,t}\Big(X^{(t)}_{s,u}\big(\mathcal{E}_{u,t}(\mathcal{E}_{s,u}(\boldsymbol{x}))\big)\Big) = X_{u,t}\Big(\mathcal{E}_{u,t}\big(X_{s,u}(\mathcal{E}_{s,u}(\boldsymbol{x}))\big)\Big) \\ &= X_{u,t}\big(\mathcal{E}_{u,t}(\Phi_{s,u}(\boldsymbol{x}))\big) = \Phi_{u,t}(\Phi_{s,u}(\boldsymbol{x})) \end{aligned} \tag{43}$$

which is the semigroup identity in (15). □

C.7 Proof of Proposition C.3

Proposition C.3 (Expanding Flow Maps are Stochastic Flow Maps)**.** *Let $\Phi_{s,t}(\boldsymbol{x}_s, \boldsymbol{\epsilon})$ be an expanding flow map (12) with conditional noise $\boldsymbol{\epsilon} \sim p_{\epsilon|s,t}$ satisfying the consistency identities (15). Then for all $t \in [s, 1]$, the pushforward satisfies $p_{t|s}(\cdot|\boldsymbol{x}_s) = (\Phi_{s,t}(\boldsymbol{x}_s, \cdot))_{\#} p_{\epsilon|s,t}$, and recovers the clean posterior $p_{1|s}(\cdot|\boldsymbol{x}_s)$ at $t = 1$.*

Proof. By Definition 3.1, the flow map on the augmented space $X_{s,t} : \mathbb{R}^{d(s)} \times \mathbb{R}^{d(t)-d(s)} \to \mathbb{R}^{d(t)}$ satisfies the pushforward identity:

$$(X_{s,t})_{\#} p^{\epsilon}_{s,t} = (X_{s,t})_{\#}(p_s \otimes p_{\epsilon|s,t}) = p_t \tag{44}$$

Using the disintegration theorem (Kallenberg, 1997), we can disintegrate both sides by the marginal p_s. The LHS disintegrates by the property that, for fixed $\boldsymbol{x}_s$, the map $X_{s,t}(\boldsymbol{x}_s, \cdot) : \mathbb{R}^{d(t)-d(s)} \to \mathbb{R}^{d(t)}$ is a function of $\boldsymbol{\epsilon}$ alone, so the conditional measure given $\boldsymbol{x}_s$ is the pushforward of $p_{\epsilon|s,t}$ along $X_{s,t}(\boldsymbol{x}_s, \cdot)$:

$$[(X_{s,t})_{\#} p^{\epsilon}_{s,t}](\cdot|\boldsymbol{x}_s) = (X_{s,t}(\boldsymbol{x}_s, \cdot))_{\#} p_{\epsilon|s,t} \tag{45}$$

The RHS of the disintegration written through the joint $p_{s,t}(\boldsymbol{x}_s, \boldsymbol{x}_t)$, is given by the conditional distribution $p_{t|s}(\cdot|\boldsymbol{x}_s)$:

$$p_t(\cdot) = \int p_{t|s}(\cdot \mid x_s) p_s(dx_s) \tag{46}$$

By the uniqueness of disintegration, we have:

$$p_{t|s}(\cdot|\boldsymbol{x}_s) = \big(X_{s,t}(\boldsymbol{x}_s, \cdot)\big)_{\#} p_{\epsilon|s,t} \tag{47}$$

The clean posterior is recovered when $t = 1$:

$$p_{1|s}(\cdot|\boldsymbol{x}_s) = \big(X_{s,1}(\boldsymbol{x}_s, \cdot)\big)_{\#} p_{\epsilon|s,1} \tag{48}$$

which concludes the proof. □

C.8 Proof of Proposition C.4

In the discrete expanding flow map setting, we define the parameterization:

$$\Phi_{s,t}(\boldsymbol{x}_s) = X_{s,t}(\mathcal{E}_{s,t}(\boldsymbol{x}_s)), \quad X_{s,t}(\boldsymbol{y}) = \tfrac{1-t}{1-s}\boldsymbol{y} + \tfrac{t-s}{1-s}\psi_{s,t}(\boldsymbol{y}) \tag{49}$$

which satisfies the tangent condition $\psi_{s,s}(\boldsymbol{y}) = \boldsymbol{y} + (1-s)b_s(\boldsymbol{y})$. We use A1-A4 from Proposition 4.1 and the additional assumption for the mean denoiser:

Assumption A5 (Diagonal-expand consistency)**.** *$\psi^{(t)}_{s,s}(\mathcal{E}_{s,t}(\boldsymbol{x})) = \mathcal{E}_{s,t}(\psi_{s,s}(\boldsymbol{x}))$ for the diagonal denoiser, where $\psi^{(t)}_{s,s} : \mathbb{R}^{d(t)\times V} \to \mathbb{R}^{d(t)\times V}$ is the diagonal denoiser at the augmented dimension $d(t)$, which leaves uninserted coordinates unchanged, and $\psi_{s,s} : \mathbb{R}^{d(s)\times V} \to (\Delta^{V-1})^{d(s)}$ is the diagonal denoiser at the native dimension $d(s)$. Equivalently, the model's clean prediction at the diagonal on a newly lifted state equals the lift of its clean prediction on the unlifted state.*

Proposition C.4 (Discrete Expanding Flow Map Identities)**.** *Let $\boldsymbol{x}^\epsilon_s := \mathcal{E}_{s,t}(\boldsymbol{x}_s)$. Then, for $s < t$ (and $s < u < t$ in the semigroup case), the expanding flow map $\Phi_{s,t}(\boldsymbol{x}_s)$ and mean denoiser $\psi_{s,t}(\boldsymbol{x}^\epsilon_s)$ satisfy the following consistency conditions:*

$$\textit{Lagrangian:} \quad \psi_{s,t}(\boldsymbol{x}^\epsilon_s) = \psi_{t,t}(\Phi_{s,t}(\boldsymbol{x}_s)) - \gamma_{s,t}\partial_t\psi_{s,t}(\boldsymbol{x}^\epsilon_s) \tag{50a}$$

$$\textit{Eulerian:} \quad \partial_s\psi_{s,t}(\boldsymbol{x}^\epsilon_s) + J_x\psi_{s,t}(\boldsymbol{x}^\epsilon_s)b_s(\boldsymbol{x}^\epsilon_s) = \kappa_{s,t}(\psi_{s,t}(\boldsymbol{x}^\epsilon_s) - \mathcal{E}_{s,t}(\psi_{s,s}(\boldsymbol{x}_s))) \tag{50b}$$

$$\textit{Semigroup:} \quad \psi_{s,t}(\boldsymbol{x}^\epsilon_s) = \omega_{s,u,t}\psi^{(t)}_{s,u}(\boldsymbol{x}^\epsilon_s) + (1-\omega_{s,u,t})\psi_{u,t}\big(\mathcal{E}_{u,t}(\Phi_{s,u}(\boldsymbol{x}_s))\big) \tag{50c}$$

where $J_x\psi_{s,t}(\boldsymbol{x}^\epsilon_s)$ is the Jacobian of $\psi_{s,t}$ with respect to its $d(t)$-dimensional input, $\omega_{s,u,t} := \frac{(u-s)(1-t)}{(t-s)(1-u)}$, $\gamma_{s,t} := \frac{(1-t)(t-s)}{1-s}$, and $\kappa_{s,t} := \frac{1-t}{(1-s)(t-s)}$. We define $\psi^{(t)}_{s,u} : \mathbb{R}^{d(t)\times V} \to \mathbb{R}^{d(t)\times V}$ as the mean denoiser over $[s,u]$ evaluated at the augmented dimension $d(t)$, the same parameterized map as $\psi_{s,u}$ applied to a longer input. Because the velocity mask of (8) enforces A3, $\psi^{(t)}_{s,u}$ leaves the coordinates inserted over $(u,t]$ unchanged, so $\psi^{(t)}_{s,u}(\boldsymbol{x}^\epsilon_s) = \mathcal{E}_{u,t}(\psi_{s,u}(\mathcal{E}_{s,u}(\boldsymbol{x}_s)))$.

Proof. Throughout this proof, we denote $\boldsymbol{y} := \boldsymbol{x}^\epsilon_s = \mathcal{E}_{s,t}(\boldsymbol{x}_s)$.

Lagrangian Identity Differentiating $X_{s,t}(\boldsymbol{y}) = \frac{1-t}{1-s}\boldsymbol{y} + \frac{t-s}{1-s}\psi_{s,t}(\boldsymbol{y})$ in t at fixed $s, \boldsymbol{y}$, we get:

$$\partial_t X_{s,t}(\boldsymbol{y}) = \frac{\psi_{s,t}(\boldsymbol{y}) - \boldsymbol{y}}{1-s} + \frac{t-s}{1-s}\partial_t\psi_{s,t}(\boldsymbol{y}) \tag{51}$$

The continuous Lagrangian on the augmented space gives $\partial_t X_{s,t}(\boldsymbol{y}) = b_t(X_{s,t}(\boldsymbol{y}))$, and the boundary condition $\psi_{t,t}(\boldsymbol{z}) = \boldsymbol{z} + (1-t)b_t(\boldsymbol{z})$ implies:

$$b_t(\boldsymbol{z}) = \frac{\psi_{t,t}(\boldsymbol{z}) - \boldsymbol{z}}{1-t} \tag{52}$$

Setting $\boldsymbol{z} = X_{s,t}(\boldsymbol{y})$ in b_t and setting the expression equal to (51) multiplied by $(1-t)$, we get:

$$\frac{1-t}{1-s}\big(\psi_{s,t}(\boldsymbol{y}) - \boldsymbol{y}\big) + \frac{(1-t)(t-s)}{1-s}\partial_t\psi_{s,t}(\boldsymbol{y}) = \psi_{t,t}(X_{s,t}(\boldsymbol{y})) - X_{s,t}(\boldsymbol{y}) \tag{53}$$

Substituting $X_{s,t}(\boldsymbol{y}) = \frac{1-t}{1-s}\boldsymbol{y} + \frac{t-s}{1-s}\psi_{s,t}(\boldsymbol{y})$ on the right-hand side, the y-terms cancel and we get:

$$\Big(\tfrac{1-t}{1-s} + \tfrac{t-s}{1-s}\Big)\psi_{s,t}(\boldsymbol{y}) + \tfrac{(1-t)(t-s)}{1-s}\partial_t\psi_{s,t}(\boldsymbol{y}) = \psi_{t,t}(X_{s,t}(\boldsymbol{y})) \tag{54}$$

The terms inside the bracket sum to 1, so we have:

$$\psi_{s,t}(\boldsymbol{y}) = \psi_{t,t}(X_{s,t}(\boldsymbol{y})) - \frac{(1-t)(t-s)}{1-s}\partial_t\psi_{s,t}(\boldsymbol{y}) \tag{55}$$

Substituting $\boldsymbol{y} = \mathcal{E}_{s,t}(\boldsymbol{x}_s) = \boldsymbol{x}^\epsilon_s$ and $X_{s,t}(\boldsymbol{y}) = \Phi_{s,t}(\boldsymbol{x}_s)$, we get the final identity:

$$\psi_{s,t}(\boldsymbol{x}^\epsilon_s) = \psi_{t,t}(\Phi_{s,t}(\boldsymbol{x}_s)) - \gamma_{s,t}\partial_t\psi_{s,t}(\boldsymbol{x}^\epsilon_s), \quad \gamma_{s,t} = \tfrac{(1-t)(t-s)}{1-s} \tag{56}$$

which is the Lagrangian identity of discrete expanding flow maps in (50).

Eulerian Identity Using the partial derivatives:

$$\partial_s\left(\tfrac{1-t}{1-s}\right) = \tfrac{1-t}{(1-s)^2}, \quad \partial_s\left(\tfrac{t-s}{1-s}\right) = -\tfrac{1-t}{(1-s)^2} \tag{57}$$

and differentiating $X_{s,t}(\boldsymbol{y}) = \frac{1-t}{1-s}\boldsymbol{y} + \frac{t-s}{1-s}\psi_{s,t}(\boldsymbol{y})$ in s at fixed $\boldsymbol{y}, t$, we have:

$$\partial_s X_{s,t}(\boldsymbol{y}) = -\tfrac{1-t}{(1-s)^2}\big(\psi_{s,t}(\boldsymbol{y}) - \boldsymbol{y}\big) + \tfrac{t-s}{1-s}\partial_s\psi_{s,t}(\boldsymbol{y}) \tag{58}$$

The continuous Eulerian identity on $\mathbb{R}^{d(t)}$ states:

$$\partial_s X_{s,t}(\boldsymbol{y}) + J_{\boldsymbol{y}} X_{s,t}(\boldsymbol{y}) b_s(\boldsymbol{y}) = 0, \quad J_{\boldsymbol{y}} X_{s,t}(\boldsymbol{y}) = \tfrac{1-t}{1-s}\boldsymbol{I} + \tfrac{t-s}{1-s} J_{\boldsymbol{y}}\psi_{s,t}(\boldsymbol{y}) \tag{59}$$

Combining (58) and (59), we get:

$$0 = -\tfrac{1-t}{(1-s)^2}\big(\psi_{s,t}(\boldsymbol{y}) - \boldsymbol{y}\big) + \tfrac{t-s}{1-s}\partial_s\psi_{s,t}(\boldsymbol{y}) + \tfrac{1-t}{1-s} b_s(\boldsymbol{y}) + \tfrac{t-s}{1-s} J_{\boldsymbol{y}}\psi_{s,t}(\boldsymbol{y}) b_s(\boldsymbol{y}) \tag{60}$$

Using $b_s(\boldsymbol{y}) = (\psi_{s,s}(\boldsymbol{y}) - \boldsymbol{y})/(1-s)$ in the third term, we get:

$$0 = -\tfrac{1-t}{(1-s)^2}\big(\psi_{s,t}(\boldsymbol{y}) - \psi_{s,s}(\boldsymbol{y})\big) + \tfrac{t-s}{1-s}\Big[\partial_s\psi_{s,t}(\boldsymbol{y}) + J_{\boldsymbol{y}}\psi_{s,t}(\boldsymbol{y}) b_s(\boldsymbol{y})\Big] \tag{61}$$

Rearranging and defining $\kappa_{s,t} := \frac{1-t}{(1-s)(t-s)}$, we get:

$$\begin{aligned}\partial_s\psi_{s,t}(\boldsymbol{y}) + J_{\boldsymbol{y}}\psi_{s,t}(\boldsymbol{y}) b_s(\boldsymbol{y}) &= \frac{1-t}{(1-s)(t-s)}\big(\psi_{s,t}(\boldsymbol{y}) - \psi_{s,s}(\boldsymbol{y})\big) \\ &= \kappa_{s,t}\big(\psi_{s,t}(\boldsymbol{y}) - \psi_{s,s}(\boldsymbol{y})\big)\end{aligned} \tag{62}$$

Substituting $\boldsymbol{y} = \boldsymbol{x}_s^\epsilon$ and using A5 to rewrite $\psi_{s,s}^{(t)}(\boldsymbol{x}_s^\epsilon) = \mathcal{E}_{s,t}(\psi_{s,s}(\boldsymbol{x}_s))$, we have:

$$\partial_s\psi_{s,t}(\boldsymbol{x}_s^\epsilon) + J_x\psi_{s,t}(\boldsymbol{x}_s^\epsilon) b_s(\boldsymbol{x}_s^\epsilon) = \kappa_{s,t}\left(\psi_{s,t}(\boldsymbol{x}_s^\epsilon) - \mathcal{E}_{s,t}(\psi_{s,s}(\boldsymbol{x}_s))\right) \tag{63}$$

which recovers the Eulerian identity of discrete expanding flow maps in (50).

Semigroup Identity For $s \le u \le t$, we introduce $\psi_{s,u}^{(t)} : \mathbb{R}^{d(t)\times V} \to \mathbb{R}^{d(t)\times V}$ as the two-time mean denoiser over $[s,u]$ applied at the augmented dimension $d(t)$ that leaves the dimensions inserted over $(u,t]$ unchanged, and $\psi_{s,u} : \mathbb{R}^{d(u)\times V} \to (\Delta^{V-1})^{d(u)}$ for the same denoiser at its native dimension $d(u)$. Since the network is applied to whatever sequence it receives, these are the same parameterized map evaluated on inputs of different length. Given this, we denote the transport map defined by $\psi_{s,u}^{(t)}$ as $X_{s,u}^{(t)} : \mathbb{R}^{d(t)\times V} \to \mathbb{R}^{d(t)\times V}$ which satisfies the semigroup condition in augmented space $X_{s,t} = X_{u,t} \circ X_{s,u}^{(t)}$.

For any $s \le u \le t$, we can expand $\psi_{s,t}$ from the definition $X_{s,t}(\boldsymbol{y}) = \frac{1-t}{1-s}\boldsymbol{y} + \frac{t-s}{1-s}\psi_{s,t}(\boldsymbol{y})$:

$$\psi_{s,t}(\boldsymbol{y}) = \frac{1-s}{t-s} X_{s,t}(\boldsymbol{y}) - \frac{1-t}{t-s}\boldsymbol{y} \tag{64}$$

Applying the augmented-space semigroup $X_{s,t} = X_{u,t} \circ X_{s,u}^{(t)}$ and the definition of $X_{u,t}$, we get:

$$X_{s,t}(\boldsymbol{y}) = X_{u,t}(X_{s,u}^{(t)}(\boldsymbol{y})) = \frac{1-t}{1-u} X_{s,u}^{(t)}(\boldsymbol{y}) + \frac{t-u}{1-u}\psi_{u,t}(X_{s,u}^{(t)}(\boldsymbol{y})) \tag{65}$$

Substituting (65) into (64), we get:

$$\psi_{s,t}(\boldsymbol{y}) = \frac{(1-s)(1-t)}{(t-s)(1-u)} X_{s,u}^{(t)}(\boldsymbol{y}) + \frac{(1-s)(t-u)}{(t-s)(1-u)}\psi_{u,t}(X_{s,u}^{(t)}(\boldsymbol{y})) - \frac{1-t}{t-s}\boldsymbol{y} \tag{66}$$

Now expanding $X_{s,u}^{(t)}(\boldsymbol{y}) = \frac{1-u}{1-s}\boldsymbol{y} + \frac{u-s}{1-s}\psi_{s,u}^{(t)}(\boldsymbol{y})$ in the first term, we have:

$$\frac{(1-s)(1-t)}{(t-s)(1-u)} X_{s,u}^{(t)}(\boldsymbol{y}) = \frac{1-t}{t-s}\boldsymbol{y} + \frac{(1-t)(u-s)}{(t-s)(1-u)}\psi_{s,u}^{(t)}(\boldsymbol{y}) \tag{67}$$

Subtracting out $\frac{1-t}{t-s}\boldsymbol{y}$ cancels the $\boldsymbol{y}$-term, and we get:

$$\psi_{s,t}(\boldsymbol{y}) = \frac{(u-s)(1-t)}{(t-s)(1-u)}\psi_{s,u}^{(t)}(\boldsymbol{y}) + \frac{(t-u)(1-s)}{(t-s)(1-u)}\psi_{u,t}(X_{s,u}^{(t)}(\boldsymbol{y})) \tag{68}$$

Defining $\omega_{s,u,t} := \frac{(u-s)(1-t)}{(t-s)(1-u)}$ and verifying:

$$1 - \omega_{s,u,t} = \frac{(t-s)(1-u) - (u-s)(1-t)}{(t-s)(1-u)} = \frac{(t-u)(1-s)}{(t-s)(1-u)} \tag{69}$$

given $(t-s)(1-u) - (u-s)(1-t) = t - tu - s + su - u + ut + s - st = (t-u)(1-s)$, which means that (68) is a convex combination. Substituting $\boldsymbol{y} = \boldsymbol{x}_s^\epsilon = \mathcal{E}_{s,t}(\boldsymbol{x}_s)$, the second term simplifies by A2 and A3:

$$X_{s,u}^{(t)}(\boldsymbol{x}_s^\epsilon) = X_{s,u}^{(t)}(\mathcal{E}_{u,t}(\mathcal{E}_{s,u}(\boldsymbol{x}_s))) = \mathcal{E}_{u,t}(X_{s,u}(\mathcal{E}_{s,u}(\boldsymbol{x}_s))) = \mathcal{E}_{u,t}(\Phi_{s,u}(\boldsymbol{x}_s)) \tag{70}$$

so that

$$\psi_{s,t}(\boldsymbol{x}_s^\epsilon) = \omega_{s,u,t}\psi_{s,u}^{(t)}(\boldsymbol{x}_s^\epsilon) + (1-\omega_{s,u,t})\psi_{u,t}\big(\mathcal{E}_{u,t}(\Phi_{s,u}(\boldsymbol{x}_s))\big) \tag{71}$$

which recovers the semigroup identity for the discrete expanding flow map in (50). Equivalently, keeping the pre-substitution term of (68) at $\boldsymbol{y} = \boldsymbol{x}_s^\epsilon$ and writing $\Phi_{s,u}^{(t)}(\boldsymbol{x}_s^\epsilon) := \frac{1-u}{1-s}\boldsymbol{x}_s^\epsilon + \frac{u-s}{1-s}\psi_{s,u}^{(t)}(\boldsymbol{x}_s^\epsilon) = X_{s,u}(\boldsymbol{x}_s^\epsilon)$ gives the *expand-once* form:

$$\psi_{s,t}(\boldsymbol{x}_s^\epsilon) = \omega_{s,u,t}\psi_{s,u}^{(t)}(\boldsymbol{x}_s^\epsilon) + (1-\omega_{s,u,t})\psi_{u,t}\big(\Phi_{s,u}^{(t)}(\boldsymbol{x}_s^\epsilon)\big), \tag{72}$$

in which both terms act on the single $d(t)$ state $\boldsymbol{x}_s^\epsilon$. This is the form implemented in Algorithm 2 and used as the semigroup training target (24). It equals the native form above by A2-A3. □

D Discrete Sequence Implementation Details

D.1 Local Time Insertions and Denoising

To integrate insertion and denoising into a single flow map framework, we must define insertion times for each position along the sequence. These insertions should be determined by the surrounding context and the sequence's denoising state.

Each position i has insertion time t_i^{ins} drawn from the insertion schedule $t_i^{\text{ins}} \sim \dot{\alpha}_s ds$ and the *local time* starts and integrates over $t_i \in [0,1]$ starting at global time $t = t_i^{\text{ins}}$ where the position is pure noise initialized as $\boldsymbol{x}_{t_i^{\text{ins}}}^i = \boldsymbol{z}$. At global time $t > t_i^{\text{ins}}$, the local time is defined as:

$$t_i = \frac{t - t_i^{\text{ins}}}{1 - t_i^{\text{ins}}} \in [0,1] \tag{73}$$

Concretely, we have

$$\boldsymbol{x}_{t_i}^i = (1-t_i)\boldsymbol{x}_0^i + t_i\boldsymbol{x}_1^i, \quad \boldsymbol{x}_0^i = \boldsymbol{z} \sim p_0 \tag{74}$$

Therefore, at time t, the input to the denoising model is a padded sequence of length $d(t)$, where each active position has a different noise level $\boldsymbol{t}_{\text{local}} = (t_1, \ldots, t_{d(t)}, 0, \ldots, 0)$, providing rich context for the denoiser.

In addition to the linear per-position interpolant, one can define a per-position coefficient $\beta : [0,1] \to [0,1]$ (with $\beta(0) = 0$, $\beta(1) = 1$) that remaps the interpolant, independently of the $t^{\text{ins}} \to t$ time reparameterization, to:

$$\boldsymbol{x}_{t_i}^i = (1-\beta(t_i))\boldsymbol{z} + \beta(t_i)\boldsymbol{x}_1^i, \quad \sigma_i = 1 - \beta(t_i), \tag{75}$$

recovering the linear case at $\beta(t_i) = t_i$. An *insertion cutoff time* $t_{\text{ins-end}} \in (0,1]$ confines all insertions to the interval $[0, t_{\text{ins-end}}]$ by reparameterizing the schedule as $\alpha_{t_{\text{ins-end}}}(t) = \alpha(\min(t/t_{\text{ins-end}}, 1))$, so no position is inserted after $t = t_{\text{ins-end}}$ and the interval $[t_{\text{ins-end}}, 1]$ becomes a pure-denoising phase. Setting $t_{\text{ins-end}} = 1$ allows insertions over the full interval.

D.2 Time Reparameterization

We adopt the time-reparameterization scheme of Lee et al. (2026), which shows that for large alphabets the decoding error $P_e(t)$ remains near its initial value across most of $[0,1]$ and collapses only in a thin band near $t = 1$. Uniform training times and evenly spaced inference grids therefore

spend nearly all their budget where the model has little to learn, worsening as $|V|$ grows. The fix is to compose the time axis with a smooth, strictly increasing $\tau : [0,1] \to [0,1]$ with $\tau(0) = 0$, $\tau(1) = 1$, drawing $\tau \sim \mathcal{U}[0,1]$ and setting $t = t(\tau)$ during training, and sampling on the warped grid $t_n = t(n/N)$. Requiring equal increments of τ to correspond to equal reductions in decoding error gives

$$\tau(t) = \frac{P_e(0) - P_e(t)}{P_e(0)} = 1 - \frac{|V|}{|V| - 1} P_e(t) \tag{76}$$

concentrating both the training distribution and the inference grid on the window where token identities are committed. We evaluate (76) once by Gauss-Hermite quadrature on a uniform grid of 10^4 points and fit cubic splines in both directions for constant-time evaluation of $\tau(t)$, $t(\tau)$, and $d\tau/dt$. The warp is applied unconditionally in training and sampling.

D.3 Time Conditioning

Since the expanding flow map $\Phi_{s,t}$ transports a sequence between two times rather than denoising at a single one, the network must condition on the source and target times (s, t). We do this through two separate sinusoidal time embedders, whose outputs are summed and injected into every transformer block via adaptive layer normalization (adaLN). The final projection of the target embedder is zero-initialized so that at the start of training the target time has no effect and the model behaves as a standard single-time denoiser conditioned on s, which we found stabilizes the early distillation phase before the two-time flow-map behavior is learned.

Since positions are inserted at different global times t_i^{ins} and have different local noise levels $\boldsymbol{t}_{\text{local}} = (t_1, \ldots, t_{d(t)}, 0, \ldots, 0)$ at any global time t, we apply *source-time* adaLN conditioning per position. Each active position's local time t_i is passed through the same sinusoidal source-time embedder used for the global source time s, producing a per-position conditioning vector c_i that modulates every transformer block and the output layer via adaLN, while the target-time embedding stays global and is broadcast across positions. Conditioning each position on its own local time makes every position behave similarly to the teacher's single-time denoiser evaluated at noise level t_i, which keeps the flow-map student consistent with the fixed-length denoiser it distills from. Since the state is mixed with the (possibly non-linear) coefficient β, the true local time fed to this embedder is recovered by inverting the mixing schedule, $t_i = \beta^{-1}(1 - \sigma_i)$, so the conditioning is exact even under non-linear mixing rather than reading t_i off the noise level directly.

D.4 Adaptive Loss Weighting

To stabilize training, we scale the distillation losses by a weight that lowers the contribution of large mismatches between the student and teacher models (Potaptchik et al., 2026b). Given the student mean denoiser $\psi_{s,t}(\boldsymbol{x}) \in \Delta^{V-1}$ and the target $\bar{\psi}_{s,t}(\boldsymbol{x}) \in \Delta^{V-1}$, the weight is defined as:

$$w_{s,t}(\boldsymbol{x}) := \texttt{sg}\left[\left(\|\Delta_{s,t}(\boldsymbol{x})\|^2 + c\right)^{-r}\right], \quad \Delta_{s,t}(\boldsymbol{x}) := \psi_{s,t}(\boldsymbol{x}) - \bar{\psi}_{s,t}(\boldsymbol{x}) \tag{77}$$

where $\Delta_{s,t}$ is the mismatch between the categorical distributions and c and r are tunable hyperparameters. This is used to scale the cross-entropy loss defined in (23), where we set $c = 1e-6$ and $r = 0.5$.

D.5 One-Step Sampling

Under the few-step sampling algorithm for discrete EFM, one-step sampling results in degenerate behavior since, in a single step, it determines the number of tokens to insert by sampling from the Binomial distribution in (21), and then takes a jump with the flow map $\Phi_{0,1}(\cdot)$ over time $t \in [0,1]$ for all inserted tokens given the fully noisy sequence. However, since the probability of the insertion time of a token being 1 is near zero, the model is never trained on the case where all tokens are newly inserted noise samples at time $t = 0$ and therefore defaults to the mode of the distribution when sampling from $\Phi_{0,1}(\boldsymbol{x}_0)$.

To overcome this, we define the denoising interval as $[t_{\text{denoise-min}}, 1]$ and the one-step map as $\Phi_{0,1}(\boldsymbol{x}_0) = \psi_{t_{\text{denoise-min}},1}(\mathcal{E}_{0,1}(\boldsymbol{x}_0))$. This allows the model to denoise from a noisy sequence at a time step $t_{\text{denoise-min}}$, which resembles the time at which the model has seen sequences of this form

during training. This ensures that the denoiser is never conditioned on the near-pure-noise full-length input at time $t = 0$, which it rarely encounters in training, while the insertion schedule still uses the true time.

E Discrete Graph Implementation Details

Here we describe the graph implementation of EFlow and EFM. The construction mirrors the sequence variant. Each *node* has an independent insertion time, and the clean target is interpolated with a Gaussian-latent prior, a learned head fires per-gap Poisson insertions, and the denoiser is rolled out with an Euler scheme on the resulting flow ODE. The one difference is that the state is a pair $(\boldsymbol{x}, \boldsymbol{E})$ with a square edge matrix $\boldsymbol{E}$. Every position update is applied consistently on the node axis and both edge axes, and the edge matrix stays symmetric with a zero diagonal at all times.

E.1 Notation

A clean graph with $d(1)$ nodes lives on the discrete space of *nodes* $\mathcal{X}_x$ and edges $\mathcal{X}_e$,

$$\mathcal{X}_x = \{0, \dots, V_x - 1\}^{d(1)}, \quad \mathcal{X}_e \in \{0, \dots, V_e - 1\}^{d(1)\times d(1)}, \tag{78}$$

where V_x counts node categories and V_e counts edge categories (with 0 reserved for the absence of an edge), the graph analogs of sequence vocabularies. Note that $\mathcal{X}_x$ carries no "no-node" class. The node count is a property of the active set, controlled entirely by the insertion process below and never by an absorbing node category. We represent a clean graph as one-hot vectors $\boldsymbol{x}_1^i \in \Delta^{V_x-1}$ at each node i and $\boldsymbol{e}_1 \in \Delta^{V_e-1}$ at each ordered pair (i, j). The full graph is the tensor of nodes $\boldsymbol{x}_1 \in \mathbb{R}^{d(1)\times V_x}$ and edges $\boldsymbol{E}_1 \in \mathbb{R}^{d(1)\times d(1)\times V_e}$, with the edge matrix symmetric and zero on the diagonal, $\boldsymbol{E}_1 = \boldsymbol{E}_1^\top$ and $\mathrm{diag}(\boldsymbol{E}_1) = 0$.

E.2 Per-node Insertion Times and Edge Activation

Each node i is assigned an insertion time $t_i^{\text{ins}} \in [0, 1]$ drawn from the schedule's inverse CDF

$$t_i^{\text{ins}} = t(\tau_i), \quad \tau_i \sim \mathrm{Uniform}(0, 1) \tag{79}$$

For padded slots, we set $t_i^{\text{ins}} = 2$ so they never activate. At global time t, a node is *active* if its insertion time has elapsed,

$$a_i(t) = \mathbf{1}\{t_i^{\text{ins}} \le t\} \in \begin{cases} 1 & \text{(active)} \\ 0 & \text{(inactive)} \end{cases} \tag{80}$$

and its per-node local time and noise level are

$$t_i = \tfrac{t - t_i^{\text{ins}}}{1 - t_i^{\text{ins}}} \cdot a_i(t), \quad \sigma_i = 1 - t_i. \tag{81}$$

An edge inherits its noise level from the nodes it connects. An ordered pair (i, j) with $i \neq j$ is active when both endpoints are active and inherits its local time as the *minimum* of the endpoint local times,

$$a_{ij}(t) = a_i(t)a_j(t)(1 - \delta_{ij}), \quad t_{ij} = \min(t_i, t_j) \tag{82}$$

In the sequence EFM, the only activation variable is per-position a_i. In the graph EFM, edges are *coupled* to nodes through (82), so insertions of nodes implicitly grow the active edge set, and an edge can never be denoised ahead of either endpoint.

E.3 Node and Edge Interpolants

Both nodes and edges are corrupted by the same flow-matching interpolant as in the sequence case, defined as a Gaussian-latent path between a one-hot target and a Gaussian prior $\mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$. For nodes we draw $\boldsymbol{x}_0 \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I}_{V_x})$ and use the linear interpolant

$$\boldsymbol{x}_t^i = t_i \boldsymbol{x}_1^i + (1 - t_i)\boldsymbol{x}_0^i, \quad i \in \{1, \dots, d(1)\} \tag{83}$$

where inactive nodes have $t_i = 0$ and stay fixed at pure noise. For edges we draw $\boldsymbol{E}_0 \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I}_{V_e})$, symmetrize it as $\tilde{\boldsymbol{E}}_0 = (\boldsymbol{E}_0 + \boldsymbol{E}_0^\top)/2$, and use the interpolant

$$\boldsymbol{E}_t^{ij} = t_{ij}\boldsymbol{E}_1^{ij} + (1 - t_{ij})\boldsymbol{E}_0^{ij}, \quad \tilde{\boldsymbol{E}}_t := \frac{1}{2}(\boldsymbol{E}_t + \boldsymbol{E}_t^\top), \quad \mathrm{diag}(\tilde{\boldsymbol{E}}_t) = 0 \tag{84}$$

where the edge local time $t_{ij} = \min(t_i, t_j)$ is inherited from its endpoints as in (82), which symmetrizes the edge matrix and sets the diagonal (indicating self-edges) to zero at every step.

E.4 Compacting the Partially Active Graph

To form the partial graph with active positions compacted to be contiguous, we `gather` on the position vector and on both axes of the edge matrix. A position j in the full graph falls in the gap $\gamma(j)$ given by

$$\gamma(j) = \sum_{k<j} a_k(t) \tag{85}$$

and the number of uninserted nodes at each gap i at time t is

$$g_i(t) = \sum_{j:a_j(t)=0} \mathbf{1}\{\gamma(j) = i\}, \quad i \in \{0, \dots, d(t)\} \tag{86}$$

which defines the prediction target for the insertion head during training.

E.5 Node and Edge Insertions

Given the compacted graph $(\boldsymbol{x}_s, \boldsymbol{E}_s)$ with $d(s)$ active nodes and the per-gap counts $g_i(t)$ of (86), a shared insertion head reads the node hidden states and predicts the two-time insertion expectation $\hat{\mathcal{I}}_{s,t}(\boldsymbol{x}_s)[i]$ for each gap $i \in \{0, \dots, d(s)\}$, trained exactly as in the sequence case (19) against $g_i(t)$. The *graph expand operator* $\mathcal{E}_{s,t}$ jumps from $d(s)$ to $d(t)$ nodes by drawing a per-gap count from the binomial law (21),

$$\ell_i \sim \text{Binomial}\Big(d_{\max} - d(s),\ \tfrac{\hat{\mathcal{I}}_{s,t}(\boldsymbol{x}_s)[i]}{d_{\max} - d(s)}\Big), \tag{87}$$

capped left-to-right at the node budget so $\sum_i \ell_i \le d_{\max} - d(s)$, and converging to $\ell_i \sim \text{Poisson}(\hat{\mathcal{I}}_{s,t}(\boldsymbol{x}_s)[i])$ in the many-step limit.

Unlike sequences, an inserted node carries *both* a node coordinate and a full row and column of edges. Writing A for the $d(s)$ existing nodes and N for the $\sum_i \ell_i$ freshly inserted nodes, the expand operator scatters the old state into the enlarged $d(t)$-node layout and fills every new coordinate with a Gaussian latent at local time $t_i = 0$:

$$\boldsymbol{x}_s^\epsilon[i] = \begin{cases} \boldsymbol{x}_s[i] & i \in A \\ \boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_{V_x}) & i \in N \end{cases}, \quad \boldsymbol{E}_s^\epsilon[i,j] = \begin{cases} \boldsymbol{E}_s[i,j] & i,j \in A \\ \tilde{\boldsymbol{e}} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_{V_e}) & i \in N \text{ or } j \in N \end{cases} \tag{88}$$

so the existing $A \times A$ edge block is copied while every pair touching a new node (the $A \times N$, $N \times A$, and $N \times N$ blocks) is seeded with noise. The result is symmetrized and its diagonal zeroed, $\boldsymbol{E}_s^\epsilon \leftarrow \frac{1}{2}(\boldsymbol{E}_s^\epsilon + \boldsymbol{E}_s^{\epsilon\top})$ with $\text{diag}(\boldsymbol{E}_s^\epsilon) = 0$, so the augmented state stays a valid symmetric graph. Because a graph has *no inherent node ordering*, the per-gap counts are needed only to define the training target against the node order of (86), and at sampling any placement of the $\sum_i \ell_i$ new nodes yields the same graph up to relabeling. This is the graph instance of positional insertion collapsing to concatenation under permutation invariance.

E.6 Denoising Nodes and Edges

The transport map $X_{s,t}$ denoises the augmented graph $(\boldsymbol{x}_s^\epsilon, \boldsymbol{E}_s^\epsilon)$ toward the clean one-hot graph. A single graph transformer $\hat{\psi}_{s,t}$ takes the augmented state, the target time t, and the per-node source local times, emitting a mean denoiser on the simplex at every node and edge:

$$\hat{\psi}_{s,t}^X(\boldsymbol{x}_s^\epsilon)[i] \in \Delta^{V_x - 1}, \quad \hat{\psi}_{s,t}^E(\boldsymbol{E}_s^\epsilon)[i,j] \in \Delta^{V_e - 1}. \tag{89}$$

Node i is conditioned on its own local time s_i and edge (i,j) on the inherited $s_{ij} = \min(s_i, s_j)$ of (82), so a more recently inserted coordinate is denoised from a higher noise level. The discrete expanding flow map is then applied identically on the node axis and on both edge axes,

$$\Phi_{s,t}^X(\boldsymbol{x}_s^\epsilon) = \tfrac{1-t}{1-s}\boldsymbol{x}_s^\epsilon + \tfrac{t-s}{1-s}\hat{\psi}_{s,t}^X(\boldsymbol{x}_s^\epsilon), \quad \Phi_{s,t}^E(\boldsymbol{E}_s^\epsilon) = \tfrac{1-t}{1-s}\boldsymbol{E}_s^\epsilon + \tfrac{t-s}{1-s}\hat{\psi}_{s,t}^E(\boldsymbol{E}_s^\epsilon), \tag{90}$$

with the same global (s,t) coefficients as the discrete sequence implementation. After each jump, the edge tensor is re-symmetrized and its diagonal zeroed, $\boldsymbol{E}_t \leftarrow \frac{1}{2}(\boldsymbol{E}_t + \boldsymbol{E}_t^\top)$ with $\text{diag}(\boldsymbol{E}_t) = 0$,

preserving symmetry and the absence of self-edges. Inactive nodes and edges are masked and stay at their prior noise until inserted.

Training uses the cross-entropy discrete EFM objective (23) summed over the active node set $a_i = 1$ and active edge set $a_{ij} = 1$ of (80)-(82), with the two channels balanced by an edge weight λ_E.

$$\mathcal{L}_{\text{diag}}(\hat{\psi}) = \mathbb{E}_t \mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{x}_1}\Big[- \sum_{i:a_i=1} \big(D_t^X \cdot \log \hat{\psi}_{t,t}^X\big)(\boldsymbol{x}_t^\epsilon)[i] - \lambda_E \sum_{(i,j):a_{ij}=1} \big(D_t^E \cdot \log \hat{\psi}_{t,t}^E\big)(\boldsymbol{E}_t^\epsilon)[i,j]\Big] \tag{91}$$

$$\mathcal{L}_{\text{cons}}(\hat{\psi}) = \mathbb{E}_{s,u,t} \mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{x}_1}\Big[- \sum_{i:a_i=1} \big(\bar{\psi}_{s,t}^X \cdot \log \hat{\psi}_{s,t}^X\big)(\boldsymbol{x}_s^\epsilon)[i] - \lambda_E \sum_{(i,j):a_{ij}=1} \big(\bar{\psi}_{s,t}^E \cdot \log \hat{\psi}_{s,t}^E\big)(\boldsymbol{E}_s^\epsilon)[i,j]\Big] \tag{92}$$

where $D_t^{\{X,E\}}$ are the instantaneous denoising target for $s = t$ and $\bar{\psi}_{s,t}^{\{X,E\}}$ is the semigroup consistency target defined in (24) for the off-diagonal ($s < t$) pairs. Because the denoiser is permutation-equivariant and every update acts identically on the node and both edge axes, the full map is invariant to node relabeling, matching the unordered structure of the target graph.

F Experiment Details

F.1 Coarse-to-Fine Molecular Conformer Generation

Datasets We conduct our experiments on the GEOM dataset (Axelrod and Gomez-Bombarelli, 2022), which comprises GEOM-QM9 and GEOM-Drugs. Following the split of GeoDiff (Xu et al., 2022), each dataset provides 40,000 molecules for training and 5,000 for validation, each with 5 conformers. For testing we use 200 molecules per dataset, yielding 22,409 conformers for QM9 and 14,324 for Drugs. Atoms are encoded by atomic number and bonds by their 4 RDKit types (single, double, triple, aromatic). For each test molecule, we generate twice its number of reference conformers, matching the GeoDiff/MSGEN evaluation protocol (Hu et al., 2026a).

Framework Setting Our main experiments adopt a coarse-to-fine two-stage decomposition of each molecule into a heavy-atom backbone and its hydrogen atoms. This reflects the fact that heavy atoms form the structural scaffold while hydrogens govern local stereochemistry and fine geometry, so the backbone is generated first to serve as a global spatial anchor that guides hydrogen placement. EFlow naturally implements this hierarchical structure within a single expanding flow: the heavy backbone is denoised from Gaussian noise and frozen by $t_{\text{ins-end}}$, and then the hydrogens are inserted on a schedule and resolved around the fixed backbone. An optional low-noise refiner forms a second stage that polishes the generated all-atom structure.

Model Architecture For fair comparison, the velocity field is the GeoDiff dual-encoder (Xu et al., 2022) used by MSGEN (Hu et al., 2026a), which is a global SchNet over the extended-order and radius graph (6 interactions), a local GIN over the bond graph (4 convolutions), higher-order extended edges (`edge_order`$= 3$), and a radius cutoff of 10 Å, all with hidden dimension 128. Each encoder emits a per-edge invariant scalar that `eq_transform` maps to an equivariant per-node velocity, and the two are summed ($v = v_{\text{global}} + v_{\text{local}}$). The network is conditioned on the global time pair (s, t) and on the per-atom local (insertion) time through Fourier embeddings ($\dim = 64$) added to the edge features. The refiner uses the identical architecture.

Training Details The interpolant is the per-position Gaussian-latent linear interpolant $\boldsymbol{x}_s^i = (1 - t_i)\boldsymbol{x}_0^i + t_i \boldsymbol{x}_1^i$, where each active atom has local time $t_i \in [0, 1]$. Heavy atoms draw $\boldsymbol{x}_0^i \sim \mathcal{N}(0, \sigma_z^2 \boldsymbol{I})$ with $\sigma_z = 1.0$. Each hydrogen is anchored on its bonded heavy parent as $\boldsymbol{x}_0^i = \boldsymbol{x}_0^{\text{parent}(i)} + \sigma_h \boldsymbol{\epsilon}$ with $\sigma_h = 0.1$, and positions are made center-of-mass free at every step. Heavy atoms are present from $t = 0$ with denoise window $[0, t_{\text{ins-end}}]$, and each hydrogen is inserted at a per-atom time t_i^{ins} drawn from a cosine schedule on $[0, t_{\text{ins-end}}]$ with window $[t_i^{\text{ins}}, 1]$. With $t_{\text{ins-end}} = 0.6$ the heavy backbone forms and freezes by $t = 0.6$, leaving a hydrogen-resolving tail on $(0.6, 1]$. The model predicts the mean velocity $v_{s,t}$ and is trained with the per-graph mean-squared error over inserted atoms, with

5,000 diagonal-only flow-matching warmup steps and the off-diagonal range annealed to full by 20,000 steps.

EFlow is trained with the AdamW optimizer, learning rate 5×10^{-4}, a cosine schedule to 10^{-6}, gradient clipping at 1.0, an EMA of the weights with decay 0.999 (used for sampling), batch size 256 for QM9 and 128 for Drugs, for 300 epochs on a single NVIDIA A100 GPU. The refiner is a separate single-scale low-noise all-atom denoiser trained with source $\boldsymbol{x}_0 = \boldsymbol{x}_1 + \sigma_r \boldsymbol{\epsilon}$ (all atoms present), where σ_r is approximated by the error of the initial insertion-denoising process ($\sigma_r = 0.4$ for QM9, 1.0 for Drugs).

Sampling EFlow (no refinement) generates conformers by Euler-integrating the per-position flow ODE on a uniform grid of 20 steps. Insertion follows the deterministic per-atom schedule (heavy atoms present from $t = 0$, each hydrogen inserted at its t_i^{ins}) and costs no additional network evaluation. Heavy atoms are frozen once $t \geq t_{\text{ins-end}}$. Because the molecular graph fixes the atom count, the expand operator is the deterministic case of the general framework, and the per-atom times t_i^{ins} are drawn once from the insertion schedule before integration, so no learned insertion head is required and no count has to be inferred at sampling time. EFlow+R (with refinement) appends a 10-step refiner pass that takes the output of the EFlow or EFM model as its initialization and integrates all atoms over $[0, 1]$.

EFM samples with the same insertion schedule but replaces the Euler steps with two-time flow-map jumps $\Phi_{s,t}$ on a grid of 1, 2, or 4 steps, so the number of network evaluations drops by an order of magnitude while the interpolant is unchanged. EFM+R (with refinement) applies the 4-step flow map followed by 10-step refinement.

Baselines We build EFlow on top of GeoDiff (Xu et al., 2022), used as the backbone network, and denote the hierarchical two-stage variant GeoDiff+MSGEN (Hu et al., 2026a). In the few-step generation setting, we evaluate GeoDiff and SubgDiff (Zhang et al., 2024) at 500 denoising steps and GeoDiff+MSGEN (Hu et al., 2026a) at two 500-step stages. For broader comparison, we include deep generative models GraphDG (Simm and Hernandez-Lobato, 2020), CGCF (Xu et al., 2021a), ConfVAE (Xu et al., 2021b), GeoMol (Ganea et al., 2021), and ConfGF (Shi et al., 2021). Reported baseline values are taken from Xu et al. (2022) and Zhang et al. (2024).

Evaluation We adopt the widely used Coverage (COV) and Average Minimum RMSD (AMR) metrics proposed by Ganea et al. (2021), in both recall (R) and precision (P) forms. Both are computed from the root-mean-square deviation (RMSD) between conformers aligned by the Kabsch algorithm (Kabsch, 1976), following the GeoDiff/MSGEN protocol (Xu et al., 2022; Hu et al., 2026a). Given a threshold δ (0.5 Å for QM9 and 1.25 Å for Drugs), COV-R is the fraction of reference conformers matched within δ by some generated conformer, and AMR-R is the average minimum RMSD to the nearest generated conformer. The precision variants swap the roles of the generated and reference sets. Each metric is reported as the mean and median over test molecules.

Property Prediction Metrics This benchmark aims to evaluate whether a generated conformer set reproduces a molecule's *ensemble-averaged* quantum-chemical behavior (Axelrod and Gomez-Bombarelli, 2022) rather than the geometry of any single conformer. Following Shi et al. (2021), we generate 50 conformers each for 30 test molecules from GEOM-QM9 and obtain the electronic energy and its HOMO and LUMO levels using the quantum chemistry calculation package Psi4 (Smith et al., 2020). Using these values, we compute five properties for each molecule: the mean and minimum conformer energies $\bar{E}$ and $E_{\min}$, the mean $\overline{\Delta\varepsilon}$, minimum $\Delta\varepsilon_{\min}$, and maximum $\Delta\varepsilon_{\max}$ HOMO-LUMO gap $\Delta\varepsilon = |\varepsilon_{\text{LUMO}} - \varepsilon_{\text{HOMO}}|$. Computing the same descriptors from each molecule's reference ensemble, we score a model by the absolute gap between its predicted and reference values, and take the median over the 30 molecules. Results are reported in Table 5 in eV, where lower is better.

F.2 Discrete Molecular Graph Generation

Dataset We train EFlow and EFM on the QM9 dataset (Wu et al., 2018) containing small molecules with up to 9 heavy atoms. Instantiating the state space of App E, the node categories are the $V_x = 4$ heavy-atom types (C, N, O, F) and the edge categories are the $V_e = 5$ bond types (no-bond, single,

Table 5: Ensemble property prediction on GEOM-QM9, reported as the *median* of absolute prediction errors (eV; lower is better). Baseline medians (RDKit, ConfGF) are from Shi et al. (2021).

Model	Steps ↓	E ↓	$E_{\min}$ ↓	$\Delta\varepsilon_{\text{avg}}$ ↓	$\Delta\varepsilon_{\min}$ ↓	$\Delta\varepsilon_{\max}$ ↓
RDKit	N/A	0.8914	0.6629	0.2947	**0.5196**	0.1617
ConfGF	500	0.5328	0.1145	0.3207	0.7365	0.1337
EFlow+R	30	**0.4506**	**0.0731**	**0.2204**	0.8717	**0.1156**
EFM+R	14	2.0521	0.1232	0.3208	2.6383	0.1672

double, triple, aromatic). The node buffer is $d_{\max} = 9$. Following Roos et al. (2026), we split the data into 100K molecules for training, 20K for validation, and 13K for testing.

Model Architecture For fair comparison, we use the same graph Transformer architecture as DeFoG (Qin et al., 2025), with 9 layers with node/edge/global hidden dimensions $(d_X, d_E, d_y) = (256, 64, 64)$, 8 attention heads, and FFN hidden dimensions $(256, 128, 128)$. The input MLPs project to dimensions $(256, 128, 128)$. The RRWP structural features are computed with $k = 12$ random-walk steps.

Training Details Both the EFlow teacher and the EFM flow-map student are trained with batch size 256, AdamW (weight decay 10^{-12}), learning rate 3×10^{-4}, constant schedule with $1,000$ warmup steps, and a maximum of $2{\times}10^6$ training steps on a single NVIDIA A100 GPU.

The Gaussian-latent interpolant of App E is instantiated with prior scale $\sigma = 1.0$ on both the node and edge axes. Training uses an antithetic time sampler with sampling-time floor $\varepsilon_t = 10^{-3}$ and edge-loss weight $\lambda_E = 5.0$. The student is initialized from the teacher checkpoint and, on each step, we sample $s = t$ with probability 0.75, training on the diagonal loss (91) against the frozen EFlow teacher's endpoint prediction, and $s < t$ with probability 0.25, training on the consistency loss (92) against the target $\bar{\psi}_{s,t}$.

Since the size of QM9 is small, we train the learned per-gap insertion head with the diagonal loss and drop the off-diagonal term of (19), so the two-time count is reconstructed at sampling time rather than learned. On a jump $s \to t$ we scale the diagonal head by the schedule fraction $\rho_{s,t} = \frac{\alpha_t - \alpha_s}{1-\alpha_s}$. We use a polynomial insertion schedule with hazard $h_t = (rt^{r-1})/(1-t^r)$ and exponent $r = 0.5$ for $t \in [10^{-4}, 1.0]$ without an explicit insertion cutoff ($t_{\text{ins-end}} = 1$) since the polynomial schedule with $r = 0.5$ is concave with median insertion time $t_i^{\text{ins}} = 0.25$.

Sampling EFlow generates by Euler-integrating the per-position flow ODE on a uniform grid of N_{steps} steps, applying the learned insertion head before each denoiser step to grow the graph. Per-gap insertion counts are drawn from the Poisson law $\ell_i \sim \text{Poisson}(\hat{\mathcal{I}}_{s,t}(\boldsymbol{x}_s, \boldsymbol{E}_s)[i])$. EFM uses the same grid with the flow map in place of the denoiser, and at $N_{\text{steps}} = 1$ reduces to a single call $\psi_{0,1}$ preceded by one insertion-head call at $t = 0$. Reported numbers are 10,000 molecules per step count.

Baselines We compare against DeFoG (Qin et al., 2025) by running their publicly released checkpoints using the same model architecture and sampling steps on $10,000$ molecules per step count and categorical flow maps (CFM) (Roos et al., 2026) via their reported results on the same model architecture and sampling steps.

Evaluation To assess sample quality, we compute the proportion of 10,000 generated graphs per method and step count that correspond to RDKit-sanitizable molecules with valid SMILES (Validity), the proportion of these that are distinct under largest-fragment SMILES canonicalization (Uniqueness), and the Fréchet ChemNet Distance (FCD) between generated and reference molecules in ChemNet's final-layer embedding space against the test dataset.

F.3 Language Modeling

Dataset We evaluate on the One Billion Word Benchmark (LM1B) (Chelba et al., 2014), a standard corpus for large-scale language modeling. We use the `bert-base-uncased` tokenizer with vocab size $V = 30,522$.

Model Architecture The architecture follows the DDiT backbone used in Sahoo et al. (2025), consisting of 12 blocks, hidden size 768, 12 attention heads, conditioning dimension 128, and dropout 0.1. The insertion head described in App D reads the same 768-dimensional pre-output hidden states and is conditioned by the same 128-dimensional time vector, adding 0.79M parameters on top of the backbone.

Training Details The EFlow denoising backbone is warm-started from the publicly released LM1B FLM checkpoint of Lee et al. (2026), but our framework also allows training from scratch. We emphasize that while the warm start accelerates convergence, the variable-length, per-position denoising and insertion objectives diverge from those used to train FLM.

The insertion head is randomly initialized and, for the first 2,000 steps, its gradients are detached from the backbone so that the untrained head does not corrupt the warm-started weights. Training then proceeds for 2×10^5 steps on 4 NVIDIA B200 GPUs with global batch size 512 (per-GPU batch size of 128, no gradient accumulation), AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$, no weight decay, learning rate 3×10^{-4} with a constant schedule after 2,500 warmup steps, gradient clipping 1.0, and EMA decay 0.9999.

The per-position insertion times follow a cosine schedule, whose cumulative insertion fraction $\alpha(t) = 1 - \cos\left(\frac{\pi}{2} t/t_{\text{ins-end}}\right)$ is confined to the window $[0, t_{\text{ins-end}}]$, so all positions are inserted by the cutoff (median insertion time $\frac{2}{3}t_{\text{ins-end}}$) and $[t_{\text{ins-end}}, 1]$ is left for pure denoising. The insertion head is trained with insertion cutoff time $t_{\text{ins-end}} = 0.5$, input noise scale 1.25, and per-position adaLN time conditioning.

We measured training cost on a single NVIDIA B200 GPU with matched architecture and data. FLM takes 0.469 s/step and EFlow takes 0.513 s/step (+9.3%). FMLM takes 0.944 s/step and EFM takes 1.117 s/step (about +18%). Distillation roughly doubles the cost of the base model, which exceeds the cost of the expand operator.

For EFM, we distill from the same frozen FLM teacher (Lee et al., 2026), but our framework also allows training from self-distillation or from an EFlow teacher. We use a diagonal fraction of 0.75, boundary probability $1/32$, and uniform-difference off-diagonal (s, t) sampling. The insertion head learns the binomial interval insertion law, with a 2000-step detached warmup. The two loss terms are merged by gradient surgery following Potaptchik et al. (2026b).

Baselines We compare against both multi-step and few-step baselines. For multi-step baselines, we compare EFlow against the fixed-length flow language model (FLM) (Lee et al., 2026) at budgets from 64 to 1024 steps, evaluated by running their publicly released checkpoints. For the few-step baselines, we compare EFM at 1, 2, and 4 steps against distilled diffusion and flow-map baselines, including Duo distilled with DCD (Sahoo et al., 2025), MDLM with SDTT (Deschenaux and Gulcehre, 2025), both Duo and MDLM distilled with Di4C, categorical flow maps (CFM) (Roos et al., 2026), and flow map language models (FMLM) (Lee et al., 2026), with the numbers reported in Lee et al. (2026).

Evaluation For each step count, we generate 1024 sequences under the EMA weights using Euler integration of the insertion-denoising process, take the argmax of the final vectors, decode with the `bert-base-uncased` tokenizer, and re-encode with the GPT-2 tokenizer. Then, we compute generative perplexity under GPT-2 Large (Radford et al., 2019). We additionally compute the sample entropy as the empirical token-frequency entropy per sample.

Learned sequence-length distribution We train EFlow on unpacked LM1B, where each example is one sentence, with a max token length of 128 for 1M steps and of 64 for 300k steps. We compare the lengths of 1,024 generated samples with all 306,688 sentences of the test split (Figure 3). The model generates sentences that terminate on their own at sample-dependent lengths, but the generated distribution matches the data only approximately. We measure the difference with the 1-Wasserstein distance (W_1), which is the average number of tokens by which lengths must change to turn one distribution into the other and is zero for identical distributions.

With a max token length of 128, generated lengths are 34.1 ± 16.4 tokens against 30.5 ± 14.8 for held-out sentences ($W_1 = 3.7$), with excess mass above 40 tokens. With a max token length of 64, the means and standard deviations are close to the 128-token results (35.9 ± 16.9 against 30.1 ± 13.3), but there is a spike at the max token length.

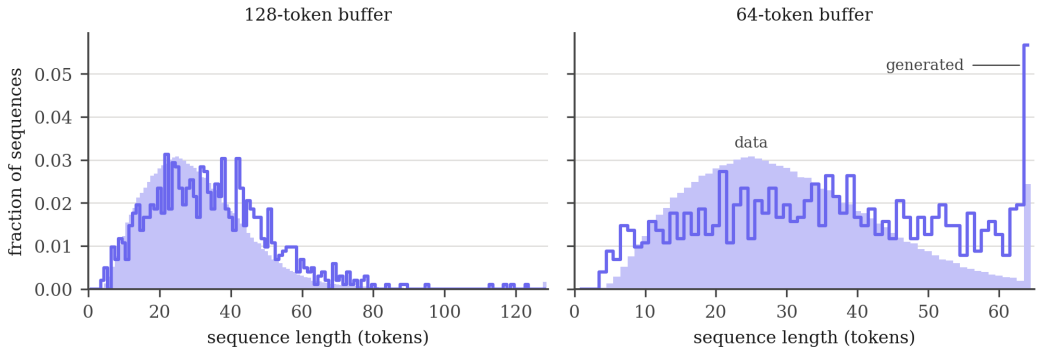

Figure 3: **Generated versus held-out sequence lengths on unpacked LM1B.** Histograms of sequence length in BERT tokens for 1,024 EFlow samples (256 sampling steps) and all 306,688 sentences of the LM1B test split (filled), at the max token length of each model. The left panel uses a max token length of 128 and 1M training steps, with generated 34.1 ± 16.4 against held-out 30.5 ± 14.8 (mean $\pm$ std), $W_1 = 3.7$. The right panel uses a max token length of 64 and 300k training steps, with generated 35.9 ± 16.9 against held-out 30.1 ± 13.3, $W_1 = 6.3$. Held-out sentences longer than 64 tokens (2.4%) are truncated in the right panel, so its last bin collects every sequence at the cap. The generated distribution follows the shape of the data but is shifted toward longer sequences.

G Algorithms

Algorithm 1 Training **Discrete Expanding Generative Flows** (EFlows)

Input: Dataset $\mathcal{D}$, insertion schedule α, time schedule $t(\cdot)$, initialized networks $\hat{D}, \hat{\mathcal{I}}$, insertion loss weight λ_{insert}, maximum length L
while not converged **do**
  $\boldsymbol{x}_1 \sim \mathcal{D}, \boldsymbol{x}_0 \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I}_{L \times V})$
  $\tau \sim \mathcal{U}(0,1), t \leftarrow t(\tau)$ ▷ *sample global time via time schedule*
  for i in $1, \ldots, L$ **do**
    $t_i^{\text{ins}} \sim \dot{\alpha}_s ds$ ▷ *sample per-token insertion time*
    $t_i \leftarrow \max\left(0, \frac{t - t_i^{\text{ins}}}{1 - t_i^{\text{ins}}}\right)$ ▷ *local time induced by global t*
    $\boldsymbol{x}_{t_i}^i \leftarrow (1 - t_i)\boldsymbol{x}_0^i + t_i \boldsymbol{x}_1^i$ ▷ *sample expanding interpolant*
  end for
  $\boldsymbol{x}_t \leftarrow (\boldsymbol{x}_{t_i}^i : t_i^{\text{ins}} \leq t)$ ▷ *active subsequence, $|\boldsymbol{x}_t| = d(t)$*
  $\hat{\boldsymbol{x}}_1 \leftarrow \hat{D}_t(\boldsymbol{x}_t)$
  $\mathcal{L}_{\text{CE}} \leftarrow -\sum_{i: t_i^{\text{ins}} \leq t} \boldsymbol{x}_1^i \cdot \log \hat{\boldsymbol{x}}_1^i$
  $\mathcal{L}_{\text{insert}} \leftarrow \frac{\dot{\alpha}_t}{1 - \alpha_t} \sum_{i=0}^{d(t)} \phi(\text{ind}_i(t) - \text{ind}_{i-1}(t) - 1, \hat{\mathcal{I}}_t(\boldsymbol{x}_t))$
  $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{CE}} + \lambda_{\text{insert}} \mathcal{L}_{\text{insert}}$
  Update $\hat{\mathcal{I}}, \hat{D}$ using $\nabla \mathcal{L}_{\text{total}}$
end while
return $\hat{\mathcal{I}}, \hat{D}$

Algorithm 2 Training **Discrete Expanding Flow Maps** (EFMs)

Input: Dataset $\mathcal{D}$, insertion schedule α, time schedule $t(\cdot)$, initialized networks $\hat{\psi}, \hat{\mathcal{I}}$, insertion loss weight λ_{insert}, maximum length L
while not converged **do**
  $\boldsymbol{x}_1 \sim \mathcal{D}, \boldsymbol{x}_0 \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I}_{L\times V})$
  $\tau_s, \tau_t \sim \mathcal{U}(0,1)$ with $\tau_s \le \tau_t$, $\tau_u \leftarrow \frac{1}{2}(\tau_s + \tau_t)$ ▷ *ordered global times*
  $s, u, t \leftarrow t(\tau_s), t(\tau_u), t(\tau_t)$ ▷ *map through the time schedule*
  for i in $1, \dots, L$ **do**
    $t_i^{\text{ins}} \sim \dot{\alpha}_s ds$ ▷ *sample per-token insertion time*
    $s_i \leftarrow \max\left(0, \frac{s - t_i^{\text{ins}}}{1 - t_i^{\text{ins}}}\right)$ ▷ *local time induced by global s*
    $\boldsymbol{x}_{s_i}^i \leftarrow (1 - s_i)\boldsymbol{x}_0^i + s_i \boldsymbol{x}_1^i$ ▷ *sample expanding interpolant*
  end for
  $\boldsymbol{x}_s \leftarrow (\boldsymbol{x}_{s_i}^i : t_i^{\text{ins}} \le s)$ ▷ *active subsequence, $|\boldsymbol{x}_s| = d(s)$*
  $\boldsymbol{x}_s^\epsilon \leftarrow (\boldsymbol{x}_{s_i}^i : t_i^{\text{ins}} \le t)$ ▷ *expand once to $d(t)$ with gaps over $(s,t]$ as noise (s_i=0)*
  $\omega_{s,u,t} \leftarrow \frac{(u-s)(1-t)}{(t-s)(1-u)}$
  $\hat{\Phi}_{s,u}^{(t)}(\boldsymbol{x}_s^\epsilon) \leftarrow \frac{1-u}{1-s}\boldsymbol{x}_s^\epsilon + \frac{u-s}{1-s}\hat{\psi}_{s,u}^{(t)}(\boldsymbol{x}_s^\epsilon)$ ▷ *flow map over $[s,u]$ at dimension $d(t)$*
  $\bar{\psi}_{s,t}(\boldsymbol{x}_s^\epsilon) \leftarrow \text{sg}\left[\omega_{s,u,t}\hat{\psi}_{s,u}^{(t)}(\boldsymbol{x}_s^\epsilon) + (1 - \omega_{s,u,t})\hat{\psi}_{u,t}\left(\hat{\Phi}_{s,u}^{(t)}(\boldsymbol{x}_s^\epsilon)\right)\right]$
  for i in $0, \dots, d(s)$ **do**
    $\mathcal{I}_s(\boldsymbol{x}_s)[i] \leftarrow \mathbb{E}[\text{ind}_i(s) - \text{ind}_{i-1}(s) - 1 \mid \boldsymbol{x}_s]$ ▷ *per-gap missing count*
    $\mathcal{I}_{s,t}(\boldsymbol{x}_s)[i] \leftarrow \frac{\alpha_t - \alpha_s}{1 - \alpha_s}\mathcal{I}_s(\boldsymbol{x}_s)[i]$ ▷ *expected insertions over $(s,t]$*
  end for
  $\mathcal{L}_{\text{insert}} \leftarrow \sum_{i=0}^{d(s)} \phi\left(\mathcal{I}_{s,t}(\boldsymbol{x}_s)[i], \hat{\mathcal{I}}_{s,t}(\boldsymbol{x}_s)[i]\right)$
  $\mathcal{L}_{\text{DEFM}}(\hat{\mathcal{E}}, \hat{\psi}) \leftarrow \sum_{i=1}^{d(t)}\left[-(\bar{\psi}_{s,t} \cdot \log \hat{\psi}_{s,t})(\boldsymbol{x}_s^\epsilon)_i - (D_t \cdot \log \hat{\psi}_{t,t})(\boldsymbol{x}_t^i)\right]$
  $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{DEFM}} + \lambda_{\text{insert}}\mathcal{L}_{\text{insert}}$
  Update $\hat{\psi}, \hat{\mathcal{I}}$ using $\nabla\mathcal{L}_{\text{total}}$
end while
return $\hat{\psi}, \hat{\mathcal{I}}$

Algorithm 3 `ExpandOperator`: Insert tokens between tokens at parameterized rate

Input: Intermediate subsequence $\boldsymbol{x}_s$, global initial time s, global jump time t, maximum length L
$n_s \leftarrow \texttt{length}(\boldsymbol{x}_s)$ ▷ *get number of inserted tokens*
$\hat{\mathcal{I}}_{s,t}(\boldsymbol{x}_s) \leftarrow \text{InsertHead}(\boldsymbol{x}_s, s, t)$
$\ell_i \sim \text{Binomial}\left(L - n_s, \frac{\hat{\mathcal{I}}_{s,t}(\boldsymbol{x}_s)[i]}{L - n_s}\right)$ ▷ *sample number of tokens to insert at each gap*
$\boldsymbol{x}_0^{\ell_i} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I}_{\ell_i})$ ▷ *sample noisy tokens*
$\boldsymbol{x}_s^\epsilon \leftarrow \left[\bigoplus_{i=1}^{d(s)}\left(\boldsymbol{x}_0^{\ell_i} \oplus \boldsymbol{x}_{s_i}^i\right)\right] \oplus \boldsymbol{x}_0^{\ell_{d(s)+1}}$ ▷ *insert noisy tokens into sequence*
return $\boldsymbol{x}_s^\epsilon$

Algorithm 4 Sampling **Discrete Expanding Flow Maps** (EFMs)

Input: Trained expanding operator $\hat{\mathcal{E}}$, mean denoiser $\hat{\psi}$, time grid $\{t_k\}_{k=1}^K$, maximum length L
$\boldsymbol{x}_0 \leftarrow$ (empty)
$\boldsymbol{x}_0 \leftarrow \hat{\mathcal{E}}_{0, t_{\text{denoise-min}}}(\boldsymbol{x}_0)$
$\boldsymbol{x}_{t_1} \leftarrow \hat{\psi}_{t_{\text{denoise-min}}, t_1}(\boldsymbol{x}_0)$ ▷ *insert from minimum time*
for k in $1, \dots, K-1$ **do**
  $\boldsymbol{x}_{t_k}^\epsilon \leftarrow \texttt{ExpandOperator}(\boldsymbol{x}_{t_k}, t_k, t_{k+1}, L)$ ▷ *insert noisy tokens into sequence*
  $\boldsymbol{x}_{t_{k+1}} \leftarrow \hat{\psi}_{t_k, t_{k+1}}(\boldsymbol{x}_{t_k}^\epsilon)$
end for
$\boldsymbol{x}_1 \leftarrow \arg\max(\boldsymbol{x}_{t_K})$ ▷ *sample one-hot argmax tokens*
return $\boldsymbol{x}_1$

H Example Generations

steps = 64

EFlow (Ours)

Gen. PPL: **54.92**
Entropy: **4.04**

[CLS] the side. [SEP] i'm guessing that it is are based on a and point. [SEP] he called the old life,'bo the, of andal'to mr. [SEP] and his father. [SEP] the hon in given patients to an does not or a test. [SEP] i was not to for her. [SEP] by way, it is not to be a, but available to be funny as well - - just as a party. [SEP] i am waiting for it to happen for each firm. [SEP] craig ward made 30 saves for the defending st louisny and antoine vermo on its 13out, including six - [SEP]

steps = 128

EFlow

Gen. PPL: **54.19**
Entropy: **4.21**

[CLS] is scheduled to be monday. [SEP] they are appealing against two key yesterday. [SEP] i'm never sure how much i mean. [SEP] he said, " a friend has come a little down and has been making the most into again. " [SEP] by the time, last week, some 5, 000 tamil there were living in the, which on have year, an even the s at found. [SEP] the price of gasoline was 2. 7 % compared with the previous year. [SEP] the next morning mr cameron said that he would not be able to give london to for the expense of defence. [SEP] they study the verys so that [SEP]

steps = 264

EFlow (Ours)

Gen. PPL: **41.29**
Entropy: **4.09**

[CLS] spent 15 years in prison at a $ 1 million military camp in afghanistan. [SEP] there's a face on it, we can't look like our else. [SEP] he never lost the great of the before. [SEP] court's bemourned her on tuesday. [SEP] two teenagers have died from a serious wound in hospital. [SEP] the military alone was not enough to force the points. [SEP] adjusted for inflation, m. a. n. m. was up 4. 2 percent, but is up slightly. [SEP] " it is a job we have agreed for as well as with the new - community policing group. [SEP] [SEP]

steps = 512

EFlow (Ours)

Gen. PPL: **49.89**
Entropy: **4.19**

[CLS] - hitter, will win against the wall. [SEP] no wonder it was right? [SEP] the wall street agency said the biggest threat to the financial system is from individuals and the, caused of over $ 3 million in assets. [SEP] the public library at the university is having a lowdown on film. [SEP] japan's olympic committee will meet again on monday. [SEP] it is a difficult (and sometimes expensive) decision to make with us, but it also takes time and effort to do with the usual for ofs. [SEP] of the world's three countries with low - on economies face up to 22pc of gdp, most has [SEP]

Figure 4: Example LM1B generations from the EFlow teacher at 64, 128, 264, and 512 sampling steps. Generative perplexity (GPT-2-Large) and entropy are reported for the individual sample shown.

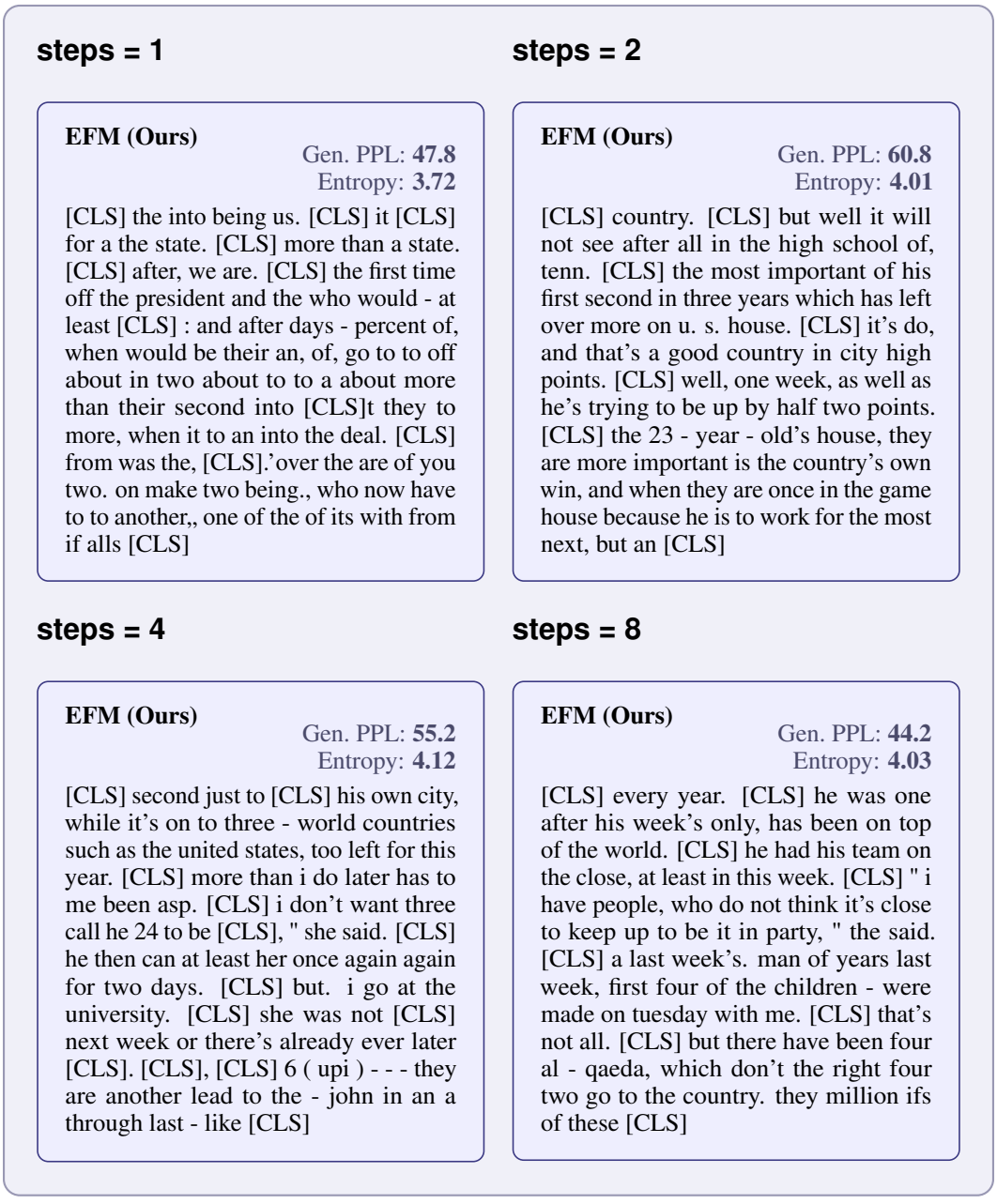

Figure 5: Example LM1B generations from the EFM student at 1, 2, 4, and 8 sampling steps. Generative perplexity (GPT-2-Large) and entropy are reported for the individual sample shown.

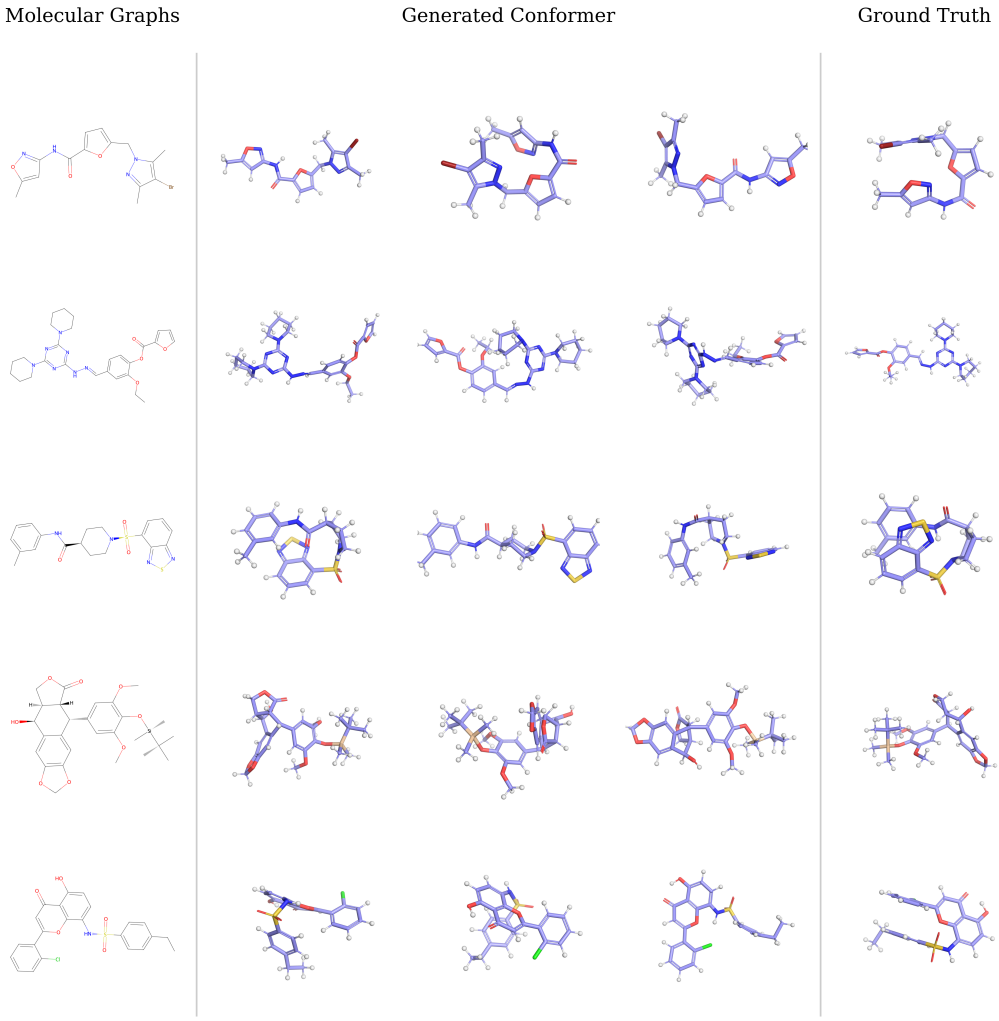

Figure 6: Molecular graph, three generated conformers with EFlow+R (30 steps), and ground-truth sample for 5 test molecules in GEOM-Drugs, visualized with PyMol (DeLano et al., 2002).

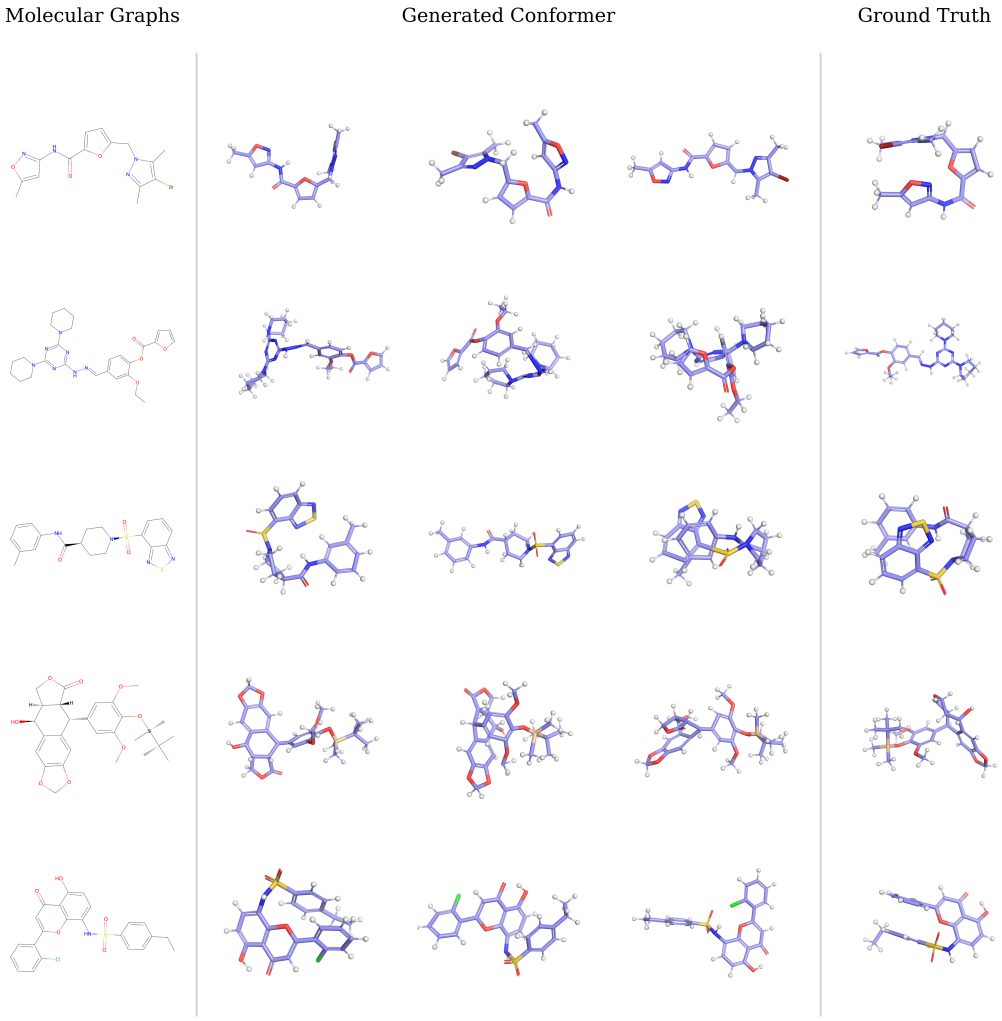

Figure 7: Molecular graph, three generated conformers with EFM+R (14 steps), and ground-truth sample for 5 test molecules in GEOM-Drugs, visualized with PyMol (DeLano et al., 2002).

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction (Sec. 1) state the contributions: Expanding Generative Flows (Sec. 3), Expanding Flow Maps (Sec. 4), and their extension to the discrete simplex (Sec. 5), evaluated on molecular conformers, molecular graphs, and language modeling (Sec. 6).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

   Justification: Section B covers the increase-only construction, the upper bound on length, the competitive size-first baselines, the scope of our evaluation, and training cost.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: The paper provides assumptions and proofs for all theoretical results. Propositions C.2 and C.1 are proved in App. C.5 and App. C.4. Proposition 4.1 is proved in App. C.6 under Assumptions A1–A4. Proposition C.3 is proved in App. C.7. Proposition C.4 is proved in App. C.8 under Assumptions A1–A5. The proofs build on the fixed-dimensional flow map identities of Boffi et al. (2025b).

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: All information needed to reproduce the main experimental results is provided in App F.1 for the conformer experiment, App F.2 for the molecular graph experiment, and App F.3 for the language modeling experiment.

Guidelines:

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

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: Code is available at https://github.com/sophtang/ExpandingFlowMaps and https://huggingface.co/ChatterjeeLab/ExpandingFlowMaps. GEOM (Axelrod and Gomez-Bombarelli, 2022), QM9 (Wu et al., 2018), and LM1B (Chelba et al., 2014) are publicly available. App F.1, App F.2, and App F.3 document the architecture, hyperparameters, and training procedure.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: All training and test details are provided in App F.1 for the conformer experiment, App F.2 for the molecular graph experiment, and App F.3 for the language modeling experiment.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: Error bars are not reported.

   Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The conformer and molecular graph experiments were trained on a single NVIDIA A100 GPU, and the LM1B experiments on 4 NVIDIA B200 GPUs (App F.1, F.2, and F.3). App F.3 also reports training time per step.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics. The paper uses publicly available datasets (GEOM, QM9, LM1B) under their respective licenses and does not involve human subjects or sensitive data.

   Guidelines:

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

10. **Broader impacts**

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

Answer: [Yes]

Justification: The paper develops a general framework for few-step generative modeling with variable output dimensionality. Positive impacts include more efficient generation of variable-sized molecular graphs for drug discovery and variable-length text. Potential negative impacts mirror those of generative models more broadly, including the possibility of misuse for generating misleading text content; we view these as low risk for the small QM9 molecules and LM1B language models considered in this work, which are far from deployment scale.

Guidelines:

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

11. **Safeguards**

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

Answer: [N/A]

Justification: The paper does not release pretrained models or scraped datasets that pose a high risk for misuse. The QM9 molecular graph models and small-scale LM1B language models trained in this work are not of a scale or capability that would require additional safeguards for release.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: All existing assets are properly cited. The QM9 dataset (Wu et al., 2018) is publicly available under the CC0 license. The LM1B benchmark is publicly available. Baseline methods including DeFoG (Qin et al., 2025), FLM (Lee et al., 2026), FMLM (Lee et al., 2026), and CFM (Roos et al., 2026) are properly cited and the corresponding evaluation protocols are followed.

Guidelines:

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

13. **New assets**

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

Answer: [N/A]

Justification: We release the code and models at `https://github.com/sophtang/ExpandingFlowMaps` and `https://huggingface.co/ChatterjeeLab/ExpandingFlowMaps`, with documentation in those repositories.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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

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

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects. All experiments are conducted on publicly available molecular (QM9) and language (LM1B) datasets.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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

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

    Answer: [N/A]

    Justification: The paper does not involve research with 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.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: LLMs are not used as an important, original, or non-standard component of the core methodology. The language modeling experiments in Sec. 6 use LM1B as a benchmark dataset for evaluating the EFlow and EFM frameworks, but the methods themselves do not rely on pretrained LLMs as part of their design.

    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.