Graph Flow Matching: Enhancing Image Generation with Neighbor-Aware Flow Fields

Md Shahriar Rahim Siddiqui[1], **Moshe Eliasof**[2,3], **Eldad Haber**[1]
[1]The University of British Columbia, Vancouver, Canada
[2]University of Cambridge, Cambridge, United Kingdom
[3]Ben-Gurion University of the Negev, Beer-Sheva, Israel
shahriar.siddiqui@ubc.ca, me532@cam.ac.uk, ehaber@eoas.ubc.ca

Abstract

Flow matching casts sample generation as learning a continuous-time velocity field that transports noise to data. Existing flow matching networks typically predict each point's velocity independently, considering only its location and time along its flow trajectory, and ignoring neighboring points. However, this pointwise approach may overlook correlations with points along the generation trajectory that could enhance velocity predictions, thereby improving downstream generation quality. To address this, we propose Graph Flow Matching (GFM), a lightweight enhancement that decomposes the learned velocity into a reaction term – any standard flow matching network – and a diffusion term that aggregates neighbor information via a graph neural module. This reaction-diffusion formulation retains the scalability of deep flow models while enriching velocity predictions with local context, all at minimal additional computational cost. Operating in the latent space of a pretrained variational autoencoder, GFM consistently improves Fréchet Inception Distance (FID) and recall across five image generation benchmarks (LSUN Church, LSUN Bedroom, FFHQ, AFHQ-Cat, and CelebA-HQ at 256 × 256), demonstrating its effectiveness as a modular enhancement to existing flow matching architectures.

1 Introduction

Flow matching has recently gained traction as a promising generative modeling paradigm, framing sample synthesis as the integration of a learned (interpolated) continuous-time velocity field that deterministically or stochastically transforms noise into data (Lipman et al. 2023; Chen et al. 2018). Given two distributions π_0 and π_1, the task is to learn a velocity field $\mathbf{v}(\mathbf{x}, t)$ that, when integrated, solves:

$$\dot{\mathbf{x}} = \mathbf{v}(\mathbf{x}, t), \quad \mathbf{x}(0) \sim \pi_0, \quad \mathbf{x}(1) \sim \pi_1. \tag{1}$$

We may view learning the field $\mathbf{v}(\mathbf{x}, t)$ as an interpolation problem over the joint domain of $\mathbf{x}$ (namely $\mathcal{X} \subseteq \mathbb{R}^d$), and t (namely $[0, 1]$): $\mathcal{X} \times [0, 1]$, where we supervise the model at discrete $(\mathbf{x}, t)$ pairs sampled from a chosen transport plan, and require it to generalize to a velocity field everywhere in $\mathcal{X}$. Given $\mathbf{x}(0)$, which is typically a sample from a Gaussian distribution, the path in $\mathcal{X} \times [0, 1]$ obtained by solving Equation (1) is what we will term a *trajectory*. In the context of image generation, each **point** $\mathbf{x}(t)$ along such a trajectory is an image. While highly effective, current flow matching architectures predict each point's velocity *pointwise*, conditioning only on its $(\mathbf{x}, t)$ coordinates without considering neighboring points in adjacent trajectories. This overlooks an important structural property shown in the literature (Gao, Huang, and Jiao 2024; Gao et al. 2024; Fukumizu et al. 2024) but not explicitly leveraged: the *smoothness* of $\mathbf{v}(\mathbf{x}, t)$ over $\mathcal{X} \times [0, 1]$. We observe that this regularity implies that neighboring points along nearby trajectories typically exhibit correlated behavior, which could be harnessed to improve flow estimation.

In this paper, we leverage this observation, and propose **Graph Flow Matching (GFM)**, which enhances standard flow networks with neighbor awareness through a reaction-diffusion decomposition:

$$\mathbf{v}(\mathbf{x}, t) = \mathbf{v}_{\text{react}}(\mathbf{x}, t) + \mathbf{v}_{\text{diff}}(\mathbf{x}, t; \mathcal{N}(\mathbf{x}, t)), \tag{2}$$

where $\mathbf{v}_{\text{react}}$ is any standard flow network, and $\mathbf{v}_{\text{diff}}$ is a lightweight graph-based correction term (a graph neural network) that aggregates information from the set of neighbors of $\mathbf{x}$ at time t, denoted by $\mathcal{N}(\mathbf{x}, t)$. Figure 1 illustrates this process, where the flow trajectory from π_0 to π_1 is augmented by incorporating neighborhood structure via graphs at intermediate time steps. Designing a flow matching pipeline requires two key decisions: *network architecture* and *training strategy* (e.g., Rectified Flow (Liu, Gong, and Liu 2022), Consistency FM (Yang et al. 2024)). We focus on the *architecture* aspect, proposing an enhancement compatible with any training strategy and existing base flow network $\mathbf{v}_{\text{react}}$, pretrained or otherwise.

In this work, we adopt the latent-space generative modeling paradigm, popularized by work such as Stable Diffusion (Rombach et al. 2022), and recently used for flow matching as in LFM (Dao et al. 2023), where image generation is performed in a perceptually compressed latent space produced by a pretrained variational autoencoder (VAE) (Kingma and Welling 2022). This choice reduces computational cost as the latent space defines meaningful distance metrics for constructing graphs, which are easy to compute. Our contributions therefore operate entirely within this latent framework. Therefore, in this work, each "point" $\mathbf{x}$ is the VAE latent encoding $\mathbf{x} \in \mathbb{R}^{4\times32\times32}$ of an image (in this paper, the latent encoding of a 256×256 RGB im-

age); all graphs operate over these latent encodings (otherwise known as *latent codes*). We shall denote these latent codes as $\mathbf{x}$ in this paper. Figure 2 and Figure 3 (in the Appendix) qualitatively demonstrate our method's effectiveness across three diverse datasets. Quantitatively, we validate GFM on five unconditional image generation benchmarks—LSUN Church (Yu et al. 2015), LSUN Bedroom (Yu et al. 2015), FFHQ (Karras, Laine, and Aila 2019), AFHQ-Cat (Choi et al. 2020), and CelebA-HQ (Karras et al. 2018) at 256×256 resolution, showing consistent improvements in Fréchet Inception Distance (FID) and recall while adding minimal computational overhead ($\lesssim 10\%$ additional parameters).

Our Contributions (i) We introduce Graph Flow Matching (GFM), a modular reaction–diffusion framework that enhances flow networks with neighbor-aware velocity correction. GFM decomposes the velocity field into a standard pointwise term (using any existing flow network) and a graph-based diffusion term, integrating seamlessly with existing backbones and training strategies without requiring modifications to losses or solvers; (ii) We validate GFM in the latent space of a pretrained VAE, demonstrating consistent improvements in unconditional image generation across five standard benchmarks. We use two architectures for $\mathbf{v}_{\text{diff}}$, which we call MPNN and GPS, to empirically validate that the use of a graph module in the flow model this way is a promising enhancement to existing flow architectures.

2 Background and Related Work

A central challenge in generative modeling is transporting a simple base distribution (e.g., Gaussian noise) into a complex, high-dimensional target distribution. This underpins models such as generative adversarial networks (GANs) (Goodfellow et al. 2014), variational autoencoders (VAEs) (Kingma and Welling 2022), normalizing flows (Papamakarios et al. 2021), and diffusion models (Song and Ermon 2019; Song et al. 2021; Ho, Jain, and Abbeel 2020). Recent trends increasingly adopt continuous-time formulations. Continuous normalizing flows (CNFs) (Chen et al. 2018) for example, generalize discrete-time flows by learning transformations via ODEs. Diffusion models, originally discrete-time (Ho, Jain, and Abbeel 2020), have evolved into continuous-time score-based models defined through stochastic differential equations (SDEs) (Song et al. 2021). Flow matching (Lipman et al. 2023) offers a deterministic, continuous-time alternative that avoids score estimation by directly supervising a velocity field that transports noise samples (that is, samples from a Gaussian distribution) to data. It combines fast sampling and simplified training, making it an attractive framework for generative modeling. Our work builds upon this foundation by introducing a graph-based architectural enhancement that enriches velocity predictions with information from neighboring points along the flow trajectory. On a different note, while recent works use flow matching *for* graph generation (Eijkelboom et al. 2024; Song et al. 2023; Qin et al. 2024; Scassola, Saccani, and Bortolussi 2025; Pombala, Grossmann, and Wolf 2025), our GFM approach focuses on a different goal; it uses graph learning techniques *for* enhancing flow matching techniques.

2.1 Flow Matching Preliminaries

Flow matching formulates sample generation as solving an ordinary differential equation (ODE) that transports a source distribution π_0 to a target distribution π_1 over a finite time horizon $t \in [0, 1]$. Given two random variables $X_0 \sim \pi_0$ and $X_1 \sim \pi_1$ with support in $\mathcal{X} \subseteq \mathbb{R}^d$, the goal is to learn a velocity field $\mathbf{v}_\theta : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$ such that the solution $\mathbf{x}(t)$ to the initial value problem

$$\dot{\mathbf{x}} = \mathbf{v}_\theta(\mathbf{x}, t), \quad \mathbf{x}(0) \sim \pi_0, \tag{3}$$

satisfies $\mathbf{x}(1) \sim \pi_1$. Learning the velocity field from data fundamentally constitutes an interpolation problem over the joint domain $\mathcal{X} \times [0, 1]$. Given velocity supervision at a finite set of spatiotemporal points $(\mathbf{x}_{t_i}, t_i)$, the model must generalize to unseen points throughout this domain. Flow matching methods create synthetic supervision by sampling pairs $(\mathbf{x}_0, \mathbf{x}_1) \sim (\pi_0, \pi_1)$, selecting an interpolation time $t \in [0, 1]$, constructing an interpolation point $\mathbf{x}_t$ between $\mathbf{x}_0$ and $\mathbf{x}_1$, and defining a target velocity $\mathbf{v}^*(\mathbf{x}_t, t)$ according to a transport plan. For example, under constant-velocity transport:

$$\mathbf{v}^*(\mathbf{x}_t, t) = \mathbf{x}_1 - \mathbf{x}_0, \quad \mathbf{x}_t = (1 - t)\mathbf{x}_0 + t\mathbf{x}_1. \tag{4}$$

The training objective minimizes the squared error between predicted and target velocities:

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t,\mathbf{x}_0,\mathbf{x}_1}\left[\left\|\mathbf{v}_\theta\left(\mathbf{x}_t, t\right) - \mathbf{v}^*(\mathbf{x}_t, t)\right\|^2\right]. \tag{5}$$

This approach transforms sample generation into a supervised learning problem over velocity samples $(\mathbf{x}_t, t, \mathbf{v}^*)$ drawn from the joint space $\mathcal{X} \times [0, 1]$. At inference time, one generates new samples by numerically solving the learned ODE Equation 3 starting from $X_0 \sim \pi_0$.

2.2 Interpolation Perspective

The flow velocity field $\mathbf{v}(\mathbf{x}_t, t)$ is smooth, typically Lipschitz continuous in $\mathbf{x}$ and t (Gao, Huang, and Jiao 2024; Gao et al. 2024). Moreover, the learned field satisfies the continuity equation, ensuring conservative mass transport (Albergo and Vanden-Eijnden 2022; Lipman et al. 2023). Learning $\mathbf{v}$ from a finite set of samples thus amounts to a scattered-data interpolation problem: one observes velocities at sparse $(\mathbf{x}_t, t)$ locations and seeks to fit a smooth vector field throughout the domain. Classical interpolants, such as radial basis functions, splines, and moving least squares, recover smooth functions by fitting to nearby samples with local support (Micchelli 1984; De Boor 1978; Levin 1998).

By contrast, most flow networks predict each $\mathbf{v}(\mathbf{x}_t, t)$ independently, ignoring correlations among neighboring points along adjacent flow trajectories. To explicitly instill this inductive bias into the architecture, we build an attention-weighted graph over the batch at each timestep and use a lightweight GNN to allow each velocity prediction to draw on its neighbors, mimicking how interpolation stencils blend local information. We view the GNN term, $\mathbf{v}_{\text{diff}}$, as a learnable analogue to classical interpolation kernels, generalizing fixed support functions to trainable networks. Further discussion is included in the Appendix.

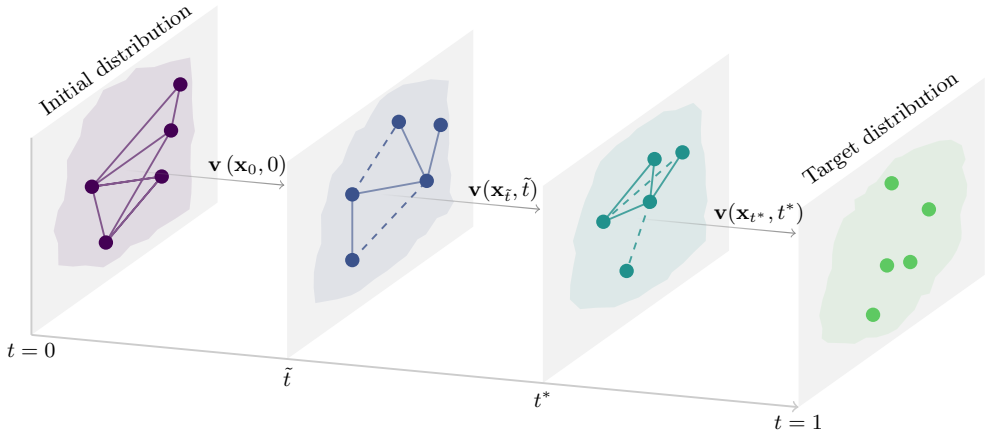

Figure 1: Graph flow matching enriches the flow trajectory from the initial distribution ($t = 0$) to the target distribution ($t = 1$) by connecting, at each intermediate time t (shown as slices) nodes $\mathbf{x}$ (shown as dots) that are *latent vectors* (VAE codes) of distinct images using attention-based similarity. The flow network output for each node $\mathbf{x}$ is its flow velocity $\mathbf{v}(\mathbf{x}, t)$. Each dot (node) at $t = 0$ represents a Gaussian noise image, while each dot (node) at $t = 1$ represents a generated image. Dashed edges indicate lower attention weights.

2.3 Graph-based Approaches in Deep Learning

The limitations of pointwise processing have motivated significant research into graph-based architectures that explicitly model relationships between data points. Graph Neural Networks (GNNs) have emerged as a powerful framework for incorporating local structure, enabling each point to aggregate information from its neighbors through message passing mechanisms (Kipf and Welling 2017; Veličković et al. 2018).

In geometric deep learning, Graph Convolutional Networks (GCNs) (Kipf and Welling 2017), Graph Attention Networks (GATs) (Veličković et al. 2018), and Message Passing Neural Networks (MPNNs) (Gilmer et al. 2017) have demonstrated the value of neighborhood aggregation across diverse domains. These architectures naturally encode inductive biases about local structure, leading to improved generalization in tasks ranging from molecular property prediction to social network analysis.

Recent work has also explored the connection between GNNs and partial differential equations (PDEs). (Eliasof, Haber, and Treister 2021) showed that the dynamics of PDEs can be encoded into GNN architectures, with the implicit PDE structure greatly influencing their suitability for specific tasks. This PDE-GNN connection provides a principled framework for designing graph architectures that capture desired physical dynamics, which is a key inspiration for the **MPNN** graph correction module used in our experiments, as outlined in Section 4.

2.4 Physical Inspiration: Reaction-Diffusion Systems

Reaction-diffusion systems (Turing 1990), which model how substances spread and interact in space, provide a natural framework for incorporating local interactions into flow matching. In these systems, the evolution of a quantity $u(\mathbf{x}, t)$ is governed by:

$$\frac{\partial u}{\partial t} = D\Delta u + R(u), \tag{6}$$

where $D\Delta u$ represents diffusion and $R(u)$ represents reaction (pointwise dynamics).

This decomposition inspires our approach: by treating the standard flow matching velocity network as a reaction term that captures pointwise dynamics and adding a graph based diffusion term that allows neighbor to neighbor communication, we enable velocity predictions to adapt based on local flow patterns. In particular, the reaction-diffusion equation is known to generate complex patterns (Turing 1990) that can explain many natural processes. Previous work has successfully utilized this equation in the context of GNNs (Haber et al. 2019; Choi et al. 2023; Eliasof, Haber, and Treister 2023, 2024) for solving graph related problems.

2.5 Latent-Space Generative Modeling

Generating high-resolution images directly in pixel space is computationally expensive and often burdens generative models with the need to reconstruct low-level details that are perceptually insignificant. Latent-space generative modeling mitigates this challenge by performing generation in a lower-dimensional, semantically meaningful space. This approach, popularized by models like Stable Diffusion (Rombach et al. 2022), uses a two-stage process: a VAE first compresses images into latent representations, and a generative model then operates within this latent space to model data distributions.

Formally, given a pre-trained VAE with encoder $E : \mathcal{I} \to \mathcal{X}$ and decoder $D : \mathcal{X} \to \mathcal{I}$, latent-space generative modeling learns to transform a prior distribution (e.g., Gaussian noise) into the distribution of encoded data points in $\mathcal{X}$. Here $\mathcal{I} = [-1, 1]^{H \times W \times C}$ denotes the pixel-space of

Figure 2: FFHQ samples (256×256) generated using the same random seed by: **(top)** baseline ADM U-Net (Dao et al. 2023), **(middle)** ADM with MPNN-based correction, and **(bottom)** ADM with a GPS-based correction module (Rampášek et al. 2022). *GFM variants produce more coherent facial features and sharper details compared to the baseline model.*

$[-1, 1]$-normalized $H \times W \times C$ images (as used by the Stable Diffusion VAE), and $\mathcal{X} \subseteq \mathbb{R}^d$ its latent representation. The decoder D then reconstructs image samples from this latent trajectory in $\mathcal{X}$ to $\mathcal{I}$. This formulation provides several critical advantages:

- **Dimensionality Reduction.** Working in latent space reduces spatial resolution and channel complexity (e.g., from $256 \times 256 \times 3$ images to $32 \times 32 \times 4$ latent tensors), significantly lowering memory and compute requirements.
- **Semantic Compression.** The VAE, trained with perceptual and reconstruction losses, discards high-frequency noise and retains structurally and semantically relevant content. This reduces the burden on the generative model to learn low-level detail.
- **Decoupled Design.** Generation and reconstruction are modular: the VAE handles image fidelity, while the generative model focuses solely on learning distributional transformations in latent space.

Building on this design, the *Latent Flow Matching (LFM)* framework (Dao et al. 2023) extends flow matching to this perceptually aligned latent space. Rather than learning stochastic score-based dynamics as in diffusion models, LFM learns a deterministic velocity field that transports noise into encoded data samples by solving an ODE in latent space. This enables faster inference, fewer function evaluations, and simplified training compared to pixel-space diffusion.

For GFM to be effective, it must connect "neighboring" points to a graph. An embedded space is, therefore, natural to us as it defines a distance in latent space, which is easy to compute and is meaningful for complex data. Therefore, we build GFM directly on this latent space setting. We adopt the same VAE architecture as LFM - namely, the Stable Diffusion VAE - which maps 256×256 RGB images into $32 \times 32 \times 4$ latent tensors. We retain all components of the LFM pipeline, including the constant-velocity transport plan, flow matching loss, and ODE solver. Our contribution lies exclusively in augmenting the velocity field $\mathbf{v}_\theta(\mathbf{x}, t)$ with a neighbor-aware graph based correction term. While there are many possible architectures for this part, here, we experimented with two architectures discussed below (Section 3).

3 Graph Flow Matching

We propose *Graph Flow Matching (GFM)*, a modular enhancement to flow matching networks that integrates local neighborhood structure through a lightweight graph module. GFM operates under a reaction–diffusion framework and is model-agnostic, where the velocity field is decomposed into a standard reaction term which could be any off-the-shelf pointwise acting flow network, and a graph-based message passing term (the so-called "diffusion" term that we propose).

3.1 Reaction–Diffusion Decomposition

Given interpolation triplets $(\mathbf{x}, t, \mathbf{v})$ constructed via a transport plan during the training phase, GFM predicts the velocity at each point as:

$$\mathbf{v}_\theta(\mathbf{x}, t) = \mathbf{v}_{\text{react}}(\mathbf{x}, t) + \mathbf{v}_{\text{diff}}(\mathbf{x}, t; \mathcal{N}(\mathbf{x}, t)), \qquad (7)$$

where $\mathbf{v}_{\text{react}}$ is any off-the-shelf flow matching architecture, and $\mathbf{v}_{\text{diff}}$ is a graph-based correction term informed by a neighborhood $\mathcal{N}(\mathbf{x}, t)$ of surrounding samples at time t.

Graph Generation. At each intermediate timestep t, we treat the VAE latent code $\mathbf{x}_t \in \mathbb{R}^{4\times32\times32}$ of every sample in the minibatch as a graph node. We compute pairwise attention scores between these nodes. The attention weights form the adjacency matrix $\mathbf{A}$, so that $\mathcal{N}(\mathbf{x}_t, t) = \{\mathbf{x}_j \mid \mathbf{A}_{ij} > 0\}$ is exactly the set of latent codes connected to $\mathbf{x}_t$, with node index i in the graph. In Appendix C we provide an ablation where K-nearest-neighbors with learnable weights was used to construct the graph.

Reaction Term: Standard Flow Networks. The reaction term $v_{\text{react}}(\mathbf{x}, t)$ can be any off-the-shelf flow matching network that predicts velocity from spatial and temporal coordinates. It typically takes the form of a U-Net (Ronneberger, Fischer, and Brox 2015; Dhariwal and Nichol 2021), Vision Transformer (Dosovitskiy et al. 2020; Peebles and Xie 2023), or any pointwise neural interpolator. This component models pointwise transport based on global training dynamics.

In our experiments (Sec. 4), we instantiate v_{react} with a UNet (ADM) (Dhariwal and Nichol 2021) and a transformer model (DiT) (Peebles and Xie 2023), following the settings in (Dao et al. 2023). However, GFM is compatible with any flow matching backbone and training strategy.

Diffusion Term: Neighbor-Aware Velocity Correction. While the diffusion term $\mathbf{v}_{\text{diff}}(\mathbf{x}, t; \mathcal{N}(\mathbf{x}, t))$ can be implemented using any graph-based neural architecture that aggregates information from neighboring samples, here we focus on two such architectures: the Message Passing Neural Network (MPNN) (Gilmer et al. 2017) and a graph transformer architecture (GPS) (Rampášek et al. 2022).

MPNN Architecture. The first diffusion term we used is a custom MPNN architecture where we use a graph gradient $\mathbf{G}$ (sometimes referred to as an incidence matrix) (Eliasof, Haber, and Treister 2021) to compute the difference between node features. This yields an edge-based quantity, or edge features. We then apply a nonlinearity to the edge features and aggregate them back to the node using the transpose of the incidence matrix. We then apply a non-linear network to the result. The network (Equation (7)) can thus be summarized by the following ODE

$$\begin{aligned}\mathbf{v}_\theta(\mathbf{x}_t, t) &= \frac{d\mathbf{x}_t}{dt} \\ &= -N_\theta^{(2)}\big(G^\top[\sigma(G\,N_\theta^{(1)}(\mathbf{x}_t, t))],\, t\big) + R(\mathbf{x}_t, t)\,.\end{aligned} \tag{8}$$

where $R(\mathbf{x}_t, t)$ denotes the reaction term v_{react}, σ denotes a nonlinearity and $N_\theta^{(1)}$ and $N_\theta^{(2)}$ are lightweight convolutional networks. This network generalizes the classical MPNN that uses Multi-Layer Perceptrons (MLPs) for $N_\theta^{(1)}$ and $N_\theta^{(2)}$. While the node data for classical graphs is unstructured, in our experiments, each node in the graphs represents an image. Hence, we utilize convolutional architectures (such as UNets) for $N_\theta^{(1)}$ and $N_\theta^{(2)}$ in this paper.

GPS Architecture. Our second instantiation of the graph correction module employs the General, Powerful, Scalable (GPS) graph transformer architecture (Rampášek et al. 2022). We adapt the GPS framework for flow matching as follows: (i) recasting each latent tensor in the batch as a node in a fully connected graph; (ii) integrating temporal information via learned time embeddings that are projected and concatenated with node features; (iii) incorporating random walk positional encodings (RWPE) generated at runtime using Pytorch Geometrics's implementation; and (iv) replacing the standard GINEConv blocks with recurrent GatedGraphConv units. The GPS architecture alternates local message passing and global multi-head attention, allowing nodes to aggregate information from the entire batch. Finally, the network projects the enriched node representations back to the original latent tensor dimensions, producing the diffusion velocity correction term. The adjacency matrix of this setup is thus attention-based.

Complexity. Graph Flow Matching (GFM) augments standard flow matching models with a diffusion term computed via a GNN, introducing an additional forward pass per ODE step. Each batch item corresponds to a node, yielding a graph with B nodes per step. The computational overhead depends on the graph topology: fully connected graphs scale as $\mathcal{O}(B^2)$ due to dense attention, while k-nearest neighbor (KNN) graphs scale as $\mathcal{O}(Bk)$ in both compute and memory. The reaction and diffusion terms are parameterized by separate networks, with the diffusion module adding only $\sim$ 5-10% to the total parameter count in our experiments. Importantly, GFM does not alter training objectives, solvers (e.g., Runge-Kutta, dopri5), or integration schedules. It preserves architectural compatibility while offering improved locality awareness at minimal additional cost.

4 Experiments

We evaluate Graph Flow Matching (GFM) on five standard unconditional image generation benchmarks: LSUN Church, LSUN Bedroom, FFHQ, AFHQ-Cat, and CelebA-HQ, all at 256×256 resolution. The goal is to assess the effect of incorporating neighbor-aware graph based correction into flow matching pipelines under consistent, state-of-the-art settings. We also conduct experiments with sparse K-nearest neighbor graphs (Appendix C), which demonstrate that GFM provides benefits even with sparse graphs.

Experimental Settings. To isolate the effect of the graph correction term, we retain all architecture and training settings from the latent flow matching (LFM) models of (Dao et al. 2023). We therefore perform our experiments in the latent space of the pretrained VAE used by the aforementioned authors. Specifically, (i) **Backbones**: The $\mathbf{v}_{react}(\mathbf{x}_t, t)$ terms we use are the ADM U-Net (Dhariwal and Nichol 2021) and DiT Transformer (Peebles and Xie 2023) (specifically, the ADM variants and DiT-L/2 variant from (Dao et al. 2023)), representing leading convolutional and attention-based flow architectures; (ii) **Latent space**: The Stable Diffusion VAE (Rombach et al. 2022), which maps 256×256 RGB images into $32 \times 32 \times 4$ latent tensors. All models operate exclusively in this latent space[1]; (iii) **Training strategy**: Constant-velocity flow matching loss (Equation (5)), the Dormand–Prince (dopri5) ODE integrator with

[1]The pretrained VAE used by (Dao et al. 2023): https://huggingface.co/stabilityai/sd-vae-ft-mse

Dataset	Model	Total Parameters (M)	$\mathbf{v}_{diff}$ Parameters (M)	FID (↓) VAE→Flow	FID (↓) True→Flow	Recall (↑)
LSUN Church	ADM (Baseline)	356.38	0	–	7.70	0.39
	ADM+MPNN (Ours)	379.81	23.43	5.06	4.94	0.52
	ADM+GPS (Ours)	374.20	18.13	4.67	4.61	0.51
	DiT (Baseline)	456.80	0	–	5.54	0.48
	DiT+MPNN (Ours)	502.09	45.29	2.90	2.92	0.65
	DiT+GPS (Ours)	481.25	24.45	2.89	3.10	0.66
FFHQ	ADM (Baseline)	406.40	0	–	8.07	0.40
	ADM+MPNN (Ours)	429.82	23.43	5.61	5.90	0.62
	ADM+GPS (Ours)	424.53	18.13	4.80	5.09	0.62
	DiT (Baseline)	456.80	0	–	4.55	0.48
	DiT+MPNN (Ours)	480.23	23.43	3.80	4.52	0.66
	DiT+GPS (Ours)	481.25	24.45	3.95	4.48	0.65
LSUN Bedroom	ADM (Baseline)	406.40	0	–	7.05	0.39
	ADM+MPNN (Ours)	429.82	23.43	4.51	4.18	0.54
	ADM+GPS (Ours)	424.53	18.13	4.26	3.97	0.56
	DiT (Baseline)	456.80	0	–	4.92	0.44
	DiT+MPNN (Ours)	480.23	23.43	2.55	2.66	0.64
	DiT+GPS (Ours)	481.25	24.45	3.31	3.57	0.65

Table 1: Performance comparison against the baselines from (Dao et al. 2023) on LSUN Church, FFHQ, and LSUN Bedroom at 256×256 resolution. We report total parameters ($\mathbf{v}_{\text{react}} + \mathbf{v}_{diff}$), with $\mathbf{v}_{diff}$ parameters listed separately to show their minimal overhead, plus FID (lower is better) and recall (higher is better). Baseline parameter counts were computed from the (Dao et al. 2023) checkpoints on our machine. Corresponding *FID(True→VAE)* values: Church $= 1.01$, FFHQ $= 1.05$, Bedroom $= 0.64$.

$rtol = atol = 10^{-5}$, and unmodified training hyperparameters from LFM (Dao et al. 2023).

This ensures that any performance gain arises from our graph-based diffusion term (i.e., the graph correction module $\mathbf{v}_{\text{diff}}$) rather than hyperparameter tuning or architectural shifts. We evaluate the two implementations of $\mathbf{v}_{\text{diff}}$ discussed in Section 3. The *MPNN* implementation uses neural components $N_\theta^{(1)}$ and $N_\theta^{(2)}$, for which we employ identical U-Net (Ronneberger, Fischer, and Brox 2015) architectures derived from the U-Net design of (Huang, Lim, and Courville 2021). The *GPS* implementation, like MPNN, implements attention-based adjacency. Full architectural and hyperparameter details are provided in the Appendix.

To provide a nuanced assessment of generation quality, we introduce three FID metrics: (i) *FID(True→Flow)*, the standard FID computed between real and generated images; (ii) *FID(True→VAE)*, which measures the FID between real images and their reconstructions via the pretrained VAE (reported in table captions) to quantify reconstruction fidelity; and (iii) *FID(VAE→Flow)*, which compares VAE reconstructions with generated samples. These metrics allow us to distinguish generative fidelity from the limitations of the VAE encoder-decoder pipeline and to better interpret results within the latent space modeling framework.

Table 1 and Table 2 report Fréchet Inception Distance (FID), recall, and parameter counts. Across all datasets and backbones, GFM consistently improves sample quality, achieving FID reductions of up to and sometimes exceeding 40% relative to the base models. These improvements are achieved with only modest computational overhead—typically less than a 10% increase in total parameters.

Qualitatively, GFM-enhanced models generate better-structured furniture and spatial layouts on LSUN Bedroom, sharper and more anatomically plausible faces on FFHQ, and more coherent architectural forms on LSUN Church as shown in Figure 2 and Figure 3 (in the Appendix). The effectiveness of GFM across both convolutional (ADM) and transformer-based (DiT) models underscores its architectural generality.

To further assess the role of graph structure, we conduct an ablation in which the adjacency matrix of the GPS module is replaced with the identity matrix, thereby eliminating inter-node communication while preserving architectural and parameter parity. Note that for the MPNN module, setting the adjacency matrix to identity would lead to a zero gradient matrix $\mathbf{G}$. This would effectively cancel out the diffusion term, leaving only the reaction term $\mathbf{v}_{\text{react}}$, making the MPNN architecture unsuitable for this particular ablation study. This ablation reduces GPS to a non-graph network and thus isolates the effect of the graph from the addition of a second network. Results in Table 3 show a clear degradation in FID and recall when graph connectivity is removed, confirming that **performance gains stem from graph structure rather than parameter count or the addition of a second network alone.** This conclusion is further supported by our KNN ablation (Appendix C), where sparse graphs with controlled parameter counts still outperform wider baseline models, demonstrating that the benefits arise specifically from enabling inter-node communication, whether through dense or sparse connectivity, rather than simply from increased model capacity. Notably, GFM provides consistent performance gains across both convolutional (ADM) and transformer-based (DiT) architectures. This underscores the generality of GFM as an architectural enhancement that is agnostic to the flow matching backbone $\mathbf{v}_{\text{react}}$.

Dataset	Model	Total Parameters (M)	$\mathbf{v}_{diff}$ Parameters (M)	FID (↓) VAE→Flow	FID (↓) True→Flow	Recall (↑)
AFHQ-Cat	ADM (Baseline)	153.10	0	7.40	7.34	0.32
	ADM+MPNN (Ours)	163.17	10.07	7.01	6.97	0.35
	ADM+GPS (Ours)	171.23	18.13	2.92	4.63	0.54
	DiT (Baseline)	456.80	0	8.24	9.05	0.42
	DiT+MPNN (Ours)	480.23	23.43	7.19	8.07	0.51
	DiT+GPS (Ours)	481.25	24.45	8.21	8.98	0.42
CelebA-HQ	ADM (Baseline)	153.10	0	7.54	6.80	0.50
	ADM+MPNN (Ours)	176.53	23.43	5.33	6.28	0.56
	ADM+GPS (Ours)	177.55	24.45	5.89	6.71	0.52
	DiT (Baseline)	456.80	0	6.85	6.23	0.55
	DiT+MPNN (Ours)	480.49	23.43	5.85	6.16	0.57
	DiT+GPS (Ours)	481.25	24.45	6.09	6.18	0.59

Table 2: Performance comparison on AFHQ-Cat and CelebA-HQ at 256×256 resolution, computed entirely on our machine with $\mathbf{v}_{react}$ architectures ADM and DiT-L/2 being taken from (Dao et al. 2023). We report total parameters ($\mathbf{v}_{\text{react}} + \mathbf{v}_{diff}$), with $\mathbf{v}_{diff}$ parameters listed separately to show their minimal overhead, plus FID (lower is better) and recall (higher is better). Corresponding *FID(True→VAE)* values for the datasets are: AFHQ-Cat $= 3.18$, CelebA-HQ $= 1.29$.

Dataset	Model	Total Parameters (M)	$\mathbf{v}_{diff}$ Parameters (M)	FID (↓) VAE→Flow	FID (↓) True→Flow	Recall (↑)
LSUN Church	ADM+GPS	374.20	18.13	4.67	4.61	0.51
	ADM+GPS (Adj=I)	374.20	18.13	6.24	5.71	0.51
	DiT+GPS	481.25	24.45	2.89	3.10	0.66
	DiT+GPS (Adj=I)	481.25	24.45	4.63	4.80	0.64
FFHQ	ADM+GPS	424.53	18.13	4.80	5.09	0.62
	ADM+GPS (Adj=I)	424.53	18.13	6.12	6.58	0.59
	DiT+GPS	481.25	24.45	3.95	4.48	0.65
	DiT+GPS (Adj=I)	481.25	24.45	4.02	4.53	0.64
AFHQ-Cat	ADM+GPS	171.23	18.13	2.92	4.63	0.54
	ADM+GPS (Adj=I)	171.23	18.13	7.51	7.20	0.35
	DiT+GPS	481.25	24.45	8.21	8.98	0.42
	DiT+GPS (Adj=I)	481.25	24.45	8.26	9.01	0.41

Table 3: Ablation study isolating graph structure benefits in LSUN Church, FFHQ, and AFHQ-Cat. We compare our full graph augmented networks against an identical-parameter baseline where the graph adjacency matrix is set to identity (Adj=I), eliminating inter-node communication while preserving all network parameters. Total parameters include the $\mathbf{v}_{diff}$ parameters. Performance degradation when graph connectivity is removed (Adj=I) demonstrates that gains arise from graph structure, not just parameter count.

5 Conclusion

We introduced Graph Flow Matching (GFM), a lightweight architectural enhancement that improves the expressiveness of flow matching generative models by incorporating local neighborhood structure via a graph-based "diffusion" term. Through a reaction–diffusion decomposition, GFM enables pointwise velocity predictors to aggregate contextual information from neighboring samples along the flow trajectory, offering a principled and scalable mechanism for improving the interpolation of the flow field.

Our results demonstrate that GFM consistently enhances generative quality across multiple datasets and architectures, reducing FID and increasing recall with minimal computational overhead. Importantly, these improvements hold across both convolutional and transformer-based backbones, highlighting the generality of our approach.

Ablation studies confirm that performance gains arise from incorporating a graph, rather than merely from the addition of parameters or a second network. These findings underscore a broader principle: local correlations along the flow trajectory can be leveraged to improve sample quality.

Looking forward, we believe GFM opens several avenues for exploration and extensions to conditional or multimodal generation. More broadly, our work suggests that combining continuous-time generative frameworks with discrete geometric priors offers a promising direction for robust, high-fidelity generative modeling.

Acknowledgments

MSRS was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC). We would like to thank Tamila Kalimullina for their valuable assistance with Figure 1.

References

Albergo, M. S.; and Vanden-Eijnden, E. 2022. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*.

Chen, R. T.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018. Neural ordinary differential equations. *Advances in neural information processing systems*, 31.

Choi, J.; Hong, S.; Park, N.; and Cho, S.-B. 2023. Gread: Graph neural reaction-diffusion networks. In *International Conference on Machine Learning*, 5722–5747. PMLR.

Choi, Y.; Uh, Y.; Yoo, J.; and Ha, J.-W. 2020. StarGAN v2: Diverse Image Synthesis for Multiple Domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Dao, Q.; Phung, H.; Nguyen, B.; and Tran, A. 2023. Flow matching in latent space. *arXiv preprint arXiv:2307.08698*.

De Boor, C. 1978. *A practical guide to splines*, volume 27. springer New York.

Dhariwal, P.; and Nichol, A. 2021. Diffusion Models Beat GANs on Image Synthesis. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*, volume 34, 8780–8794. Curran Associates, Inc.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Eijkelboom, F.; Bartosh, G.; Andersson Naesseth, C.; Welling, M.; and van de Meent, J.-W. 2024. Variational flow matching for graph generation. *Advances in Neural Information Processing Systems*, 37: 11735–11764.

Eliasof, M.; Haber, E.; and Treister, E. 2021. PDE-GCN: Novel Architectures for Graph Neural Networks Motivated by Partial Differential Equations. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*, volume 34, 3836–3849. Curran Associates, Inc.

Eliasof, M.; Haber, E.; and Treister, E. 2023. Adr-gnn: advection-diffusion-reaction graph neural networks. *arXiv preprint arXiv:2307.16092*, 108.

Eliasof, M.; Haber, E.; and Treister, E. 2024. Graph neural reaction diffusion models. *SIAM Journal on Scientific Computing*, 46(4): C399–C420.

Fukumizu, K.; Suzuki, T.; Isobe, N.; Oko, K.; and Koyama, M. 2024. Flow matching achieves almost minimax optimal convergence. *arXiv preprint arXiv:2405.20879*.

Gao, Y.; Huang, J.; and Jiao, Y. 2024. Gaussian interpolation flows. *Journal of Machine Learning Research*, 25(253): 1–52.

Gao, Y.; Huang, J.; Jiao, Y.; and Zheng, S. 2024. Convergence of continuous normalizing flows for learning probability distributions. *arXiv preprint arXiv:2404.00551*.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural Message Passing for Quantum Chemistry. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 1263–1272. PMLR.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27.

Haber, E.; Lensink, K.; Treister, E.; and Ruthotto, L. 2019. IMEXnet a forward stable deep neural network. In *International Conference on Machine Learning*, 2525–2534. PMLR.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33: 6840–6851.

Huang, C.-W.; Lim, J. H.; and Courville, A. C. 2021. A variational perspective on diffusion-based generative models and score matching. *Advances in Neural Information Processing Systems*, 34: 22863–22876.

Karras, T.; Aila, T.; Laine, S.; and Lehtinen, J. 2018. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *International Conference on Learning Representations*.

Karras, T.; Laine, S.; and Aila, T. 2019. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 4401–4410.

Kingma, D. P.; and Welling, M. 2022. Auto-Encoding Variational Bayes. arXiv:1312.6114.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

Levin, D. 1998. The approximation power of moving least-squares. *Mathematics of computation*, 67(224): 1517–1531.

Lipman, Y.; Chen, R. T. Q.; Ben-Hamu, H.; Nickel, M.; and Le, M. 2023. Flow Matching for Generative Modeling. arXiv:2210.02747.

Liu, X.; Gong, C.; and Liu, Q. 2022. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*.

Micchelli, C. A. 1984. Interpolation of scattered data: distance matrices and conditionally positive definite functions. In *Approximation theory and spline functions*, 143–145. Springer.

Papamakarios, G.; Nalisnick, E.; Rezende, D. J.; Mohamed, S.; and Lakshminarayanan, B. 2021. Normalizing Flows for Probabilistic Modeling and Inference. *Journal of Machine Learning Research*, 22(57): 1–64.

Peebles, W.; and Xie, S. 2023. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, 4195–4205.

Pombala, P.; Grossmann, G.; and Wolf, V. 2025. Exploring Molecule Generation Using Latent Space Graph Diffusion. *arXiv preprint arXiv:2501.03696*.

Qin, Y.; Madeira, M.; Thanou, D.; and Frossard, P. 2024. Defog: Discrete flow matching for graph generation. *arXiv preprint arXiv:2410.04263*.

Rampášek, L.; Galkin, M.; Dwivedi, V. P.; Luu, A. T.; Wolf, G.; and Beaini, D. 2022. Recipe for a general, powerful,

scalable graph transformer. *Advances in Neural Information Processing Systems*, 35: 14501–14515.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10684–10695.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, 234–241. Springer.

Scassola, D.; Saccani, S.; and Bortolussi, L. 2025. Graph Conditional Flow Matching for Relational Data Generation. *arXiv preprint arXiv:2505.15668*.

Song, Y.; and Ermon, S. 2019. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32.

Song, Y.; Gong, J.; Xu, M.; Cao, Z.; Lan, Y.; Ermon, S.; Zhou, H.; and Ma, W.-Y. 2023. Equivariant flow matching with hybrid probability transport for 3d molecule generation. *Advances in Neural Information Processing Systems*, 36: 549–568.

Song, Y.; Sohl-Dickstein, J.; Kingma, D. P.; Kumar, A.; Ermon, S.; and Poole, B. 2021. Score-Based Generative Modeling through Stochastic Differential Equations. arXiv:2011.13456.

Turing, A. M. 1990. The chemical basis of morphogenesis. *Bulletin of mathematical biology*, 52: 153–197.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. *International Conference on Learning Representations*. Accepted as poster.

Yang, L.; Zhang, Z.; Zhang, Z.; Liu, X.; Xu, M.; Zhang, W.; Meng, C.; Ermon, S.; and Cui, B. 2024. Consistency Flow Matching: Defining Straight Flows with Velocity Consistency. *arXiv preprint arXiv:2407.02398*.

Yu, F.; Zhang, Y.; Song, S.; Seff, A.; and Xiao, J. 2015. LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop. *arXiv preprint arXiv:1506.03365*.