Vector Field Diffusion Models for Point Cloud Generation on Manifolds

Author Name

Abstract

We introduce a vector field-based diffusion model for 3D point cloud generation that represents arbitrary-dimensional manifolds (e.g. 1D curves, 2D surfaces, 3D solids) embedded in $\mathbb{R}^3$. The model defines a forward diffusion process that perturbs points off the data manifold and a learned reverse process that denoises via a time-dependent vector field. We present the formalism of forward and reverse diffusion in the ambient space, and describe how a neural network learns a continuous vector field to progressively concentrate points onto a lower-dimensional manifold during generation. Theoretical results demonstrate that our approach can support arbitrary output resolution and accurately represent data concentrated on smooth manifolds, addressing limitations of prior diffusion models and invertible flows. In particular, we prove that an invertible flow cannot exactly model a distribution supported on a lower-dimensional manifold, whereas our diffusion model (with a non-invertible noising process) can target such singular distributions. We further show that the learned denoising vector field provably attracts points onto the correct manifold, yielding smooth surfaces. Empirical insights from the literature and prior work support these claims: diffusion models and score-based methods have achieved state-of-the-art point cloud generation quality:contentReference[oaicite:0]index=0, outperforming methods like normalizing flows on shape data:contentReference[oaicite:1]index=1:contentReference[oaicite:2]index=2. Our framework bridges score-based diffusion and geometric learning, offering a new approach for generative modeling of 3D structures with inherent geometric constraints.

1 Introduction

Generative modeling of 3D shapes in the form of point clouds is challenging due to the irregular, continuous nature of point cloud data. Unlike images on regular grids, point clouds consist of unordered points in $\mathbb{R}^3$ that often lie on lower-dimensional surfaces (e.g. the surface of an object). Traditional generative approaches for point clouds have included GANs, auto-regressive models, and normalizing flows. However, each has limitations: GANs suffer from training instabilities and mode collapse, auto-regressive models impose arbitrary ordering of points, and normalizing flows require bijective transfor-

mations in $\mathbb{R}^3$ that struggle with data supported on lower-dimensional manifolds:contentReference[oaicite:3]index=3. In particular, the invertibility constraint of flows means they cannot perfectly represent distributions confined to a surface or curve in space:contentReference[oaicite:4]index=4. In practice, flow-based models must add artificial jitter to the data manifold to train, which degrades the fidelity of generated samples:contentReference[oaicite:5]index=5.

Denoising diffusion probabilistic models (DDPMs) and score-based generative models have recently emerged as powerful alternatives for image and audio generation, offering stable training and excellent sample quality. These models define a forward diffusion process that gradually adds noise to data, and learn a reverse denoising process to synthesize new samples from noise:contentReference[oaicite:6]index=6. Diffusion models do not require invertible mappings; instead, they learn the gradient of the log-density (the score) which can guide samples toward high-density regions. This makes them well-suited to data on complex manifolds, since the learned score function can in principle concentrate probability mass on lower-dimensional structures. Indeed, recent works have applied diffusion or score-based models to point clouds, achieving competitive or state-of-the-art results in 3D shape generation:contentReference[oaicite:7]index=7. For example, Luo and Hu [3] introduced a DDPM for point clouds and demonstrated generation quality on par with or better than GANs and flows. Similarly, score-based methods like ShapeGF by Cai *et al.* [4] directly learn the gradient field of shape density, producing accurate point clouds that yield smooth implicit surfaces:contentReference[oaicite:8]index=8.

In this work, we propose a *vector field-based diffusion model* for point cloud generation that explicitly leverages the geometric nature of 3D shapes. Instead of representing the denoising process as a generic black-box network predicting noise, we interpret it as a continuous *time-dependent vector field* $v_\theta(x, t)$ on $\mathbb{R}^3$ that is learned to transport points from an initial noise distribution to the data distribution. By doing so, our model effectively learns the underlying manifold structure as a vector field that attracts points onto the manifold. This approach enables several key advantages:

- **Manifold Support:** The model can represent distributions concentrated on lower-dimensional manifolds in $\mathbb{R}^3$ (such as surfaces or curves), which standard $\mathbb{R}^3$ diffusion models struggle with. The learned vector field inherently captures the manifold's geometry, guiding off-manifold points toward high-density surface regions.

- **Arbitrary Resolution:** Our generative process is not limited to a fixed number of points. Because the vector field is defined continuously over space and time, we can sample an arbitrary number of initial points from noise and evolve them, obtaining arbitrarily dense point clouds of a shape. This resolution-invariance means we can generate more points at test time to improve surface fidelity, a phenomenon also observed in concurrent work like PointInfinity:contentReference[oaicite:9]index=9.

- **Theoretical Guarantees:** We provide theoretical analysis showing that

(i) unlike invertible flows, our diffusion-based method can in principle converge to a distribution exactly supported on a smooth manifold (Theorem **??**), and (ii) the learned score/vector field yields an attracting flow toward the manifold, essentially recovering the correct surface in the limit of denoising (Lemma 1). We also formalize the model's resolution-independence.

- **Empirical Efficacy:** While our focus is on the theoretical and methodological framework, we relate our approach to empirical findings in prior literature. Diffusion and score-based models have been shown to produce high-quality, smooth point clouds with fewer outliers than earlier methods:contentReference[oaicite:10]index=10, supporting the benefits of a learned gradient field. We draw on those results to illustrate the potential of our approach.

In the following, we first review background on diffusion models and challenges in point cloud generation (Sec. 2). We then describe the proposed vector field diffusion model in detail (Sec. 3), including the forward/reverse processes and the neural vector field formulation. Sec. **??** presents theoretical results on manifold representation and resolution. Finally, we discuss connections to existing methods, highlighting differences and advantages (Sec. 4).

2 Background and Related Work

2.1 Diffusion Models and Score-Based Generative Modeling

Denoising diffusion probabilistic models (DDPMs) [1] and score-based generative models [2] both implement a two-step generative scheme: a forward noise-adding process and a learned reverse denoising process. In a DDPM, one defines a discrete-time Markov chain $x_0 \sim q(x_0)$ (data), $x_1, x_2, \ldots, x_T \sim q(x_t|x_{t-1})$ that gradually adds Gaussian noise until x_T approximates an easy prior (typically a standard normal). For example, a common formulation is $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}\, x_{t-1}, \beta_t I)$ with schedule $0 < \beta_1 < \cdots < \beta_T < 1$. In the limit of $T \to \infty$ and appropriately small β_t, x_T approaches an isotropic Gaussian. The reverse process $p_\theta(x_{t-1}|x_t)$ is parameterized by a neural network and trained to invert the forward diffusion. Equivalently, one can train a neural network $s_\theta(x_t, t)$ to approximate the *score* $\nabla_{x_t} \log q(x_t)$ (i.e. the gradient of log-density) at each intermediate time, and use it to generate samples by iterative denoising (e.g. using Langevin dynamics or directly via the estimated reverse conditional). Song *et al.* [2] further generalized diffusion to continuous time SDEs:

$$d\mathbf{x} = f(t)\,\mathbf{x}\,dt + g(t)\,d\mathbf{w}\,,$$

with f, g chosen such that the marginal $p_t(\mathbf{x})$ evolves from p_0 (data) to p_T (noise). For instance, the Variance Exploding (VE) SDE uses $f(t) = 0$ and $g(t) = \sigma(t)$ to gradually inject noise, and the Variance Preserving (VP) SDE

uses $f(t) = -\frac{1}{2}\beta(t)\mathbf{x}$, $g(t) = \sqrt{\beta(t)}$ (replicating the discrete β-schedule). The reverse-time SDE becomes

$$d\mathbf{x} = [f(t)\mathbf{x} - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})]\, dt + g(t)\, d\bar{\mathbf{w}}\,,$$

where $d\bar{\mathbf{w}}$ is a reverse Wiener process. In the special case of the probability flow ODE (deterministic sampling), the diffusion can be realized as an ODE without the stochastic term:

$$\frac{d\mathbf{x}}{dt} = f(t)\,\mathbf{x} - \tfrac{1}{2} g(t)^2\, \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\,, \tag{1}$$

which is a time-varying vector field $v(\mathbf{x}, t)$ whose unknown part is the score $-\frac{1}{2}g^2 \nabla_{\mathbf{x}} \log p_t$. The neural network $s_\theta(\mathbf{x}, t)$ can be used to approximate the score and hence define the vector field for simulation. This connection between diffusion models and continuous normalizing flows (CNFs) has been noted in recent literature, leading to approaches like *flow matching* that train a time-dependent vector field directly by matching probability trajectories:contentReference[oaicite:11]index=11.

2.2 Point Cloud Generation and Diffusion Methods

Challenges of Manifold Data: A key difficulty in modeling point clouds is that shapes often lie on or near a lower-dimensional manifold in $\mathbb{R}^3$ (for example, points sampled on a 2D surface of an object). Distributions of this nature are singular with respect to the ambient volume measure, meaning a probability density might not exist in the usual sense. Standard likelihood-based models in $\mathbb{R}^3$ (with absolutely continuous densities) struggle here. Invertible flows in particular *cannot* reduce the support's dimensionality: a diffeomorphic mapping from $\mathbb{R}^n$ to $\mathbb{R}^n$ will map absolutely continuous distributions to absolutely continuous distributions, and cannot concentrate mass onto a lower-d manifold without becoming ill-defined:contentReference[oaicite:12]index=12. In other words, if data satisfy the manifold hypothesis (lying on an d-dimensional manifold with $d < n$), a normalizing flow with an n-dimensional latent cannot perfectly fit the data distribution:contentReference[oaicite:13]index=13. As a workaround, flow-based methods for shapes (e.g. PointFlow [5]) often add a small noise to points or otherwise thicken the manifold during training:contentReference[oaicite:14]index=14. This "dequantization" allows them to train on an augmented density that spreads slightly off the true surface, but it introduces blur and reduces the sharpness of generated shapes:contentReference[oaicite:15]index=15. Recent work explicitly addressing manifold data in flows (e.g. *ManiFlow* [9]) shows that flows trained on perturbed data can implicitly learn the manifold and then project samples onto it post-hoc:contentReference[oaicite:16]index=16, but this adds complexity and still relies on an intermediate noised distribution.

By contrast, diffusion models naturally avoid this limitation because the forward process is allowed to be non-invertible (it destroys information by adding noise). The reverse process is not a bijection but a stochastic process that can collapse dimensions by concentrating probability mass. As long as we can learn

the score field in the vicinity of the manifold, the reverse diffusion can yield samples exactly on the manifold in the zero-noise limit. In fact, score-based methods can be interpreted as learning an energy landscape or log-density function that is sharply peaked on the data manifold, whose gradient (the score) directs any off-manifold point toward the high-density region (the surface). The approach of Cai *et al.* [4] exemplifies this: they train a model to predict the gradient of the log-density of a shape's point distribution, and generate shapes by iteratively moving random points in the direction of this gradient (essentially performing Langevin dynamics):contentReference[oaicite:17]index=17:contentReference[oaicite:18]index=18. This method, known as ShapeGF, demonstrated that explicitly modeling the gradient field yields high-quality shapes and even enables extracting an implicit surface representation from the model:contentReference[oaicite:19]index=19.

Diffusion Models for Point Clouds: Building on the success of diffusion in images, several works have applied diffusion models to point cloud generation. Luo and Hu [3] introduced a DDPM-based model operating directly on point coordinates. Their model adds Gaussian noise to point locations and learns to reverse this process, conditioned on a latent shape code for each object. This approach achieved competitive results on ShapeNet benchmarks, outperforming earlier GAN and flow models in metrics like coverage and fidelity (e.g. Chamfer and Earth Mover's distances):contentReference[oaicite:20]index=20:contentReference[oaicite:21]index=21. Subsequent works have explored improvements: for instance, *LION* by Zeng *et al.* [8] uses a latent hierarchical VAE in conjunction with diffusion, where diffusion operates in a compressed latent point representation to ease computation. While LION can accelerate training and allow higher point counts at generation, the latent space (being low-dimensional) may limit the detail or require a decoder to upscale the point cloud. In another direction, Huang *et al.*'s PointInfinity [7] achieves *resolution-invariance* by using a transformer-based diffusion model that is trained on relatively small point clouds but can generate orders of magnitude more points at test time. Notably, they find that increasing the number of generated points (e.g. from 1k to 100k) improves the surface quality:contentReference[oaicite:22]index=22, which underscores the importance of flexible resolution in generative models. Our approach addresses this by design: since we model a continuous field, generating more points simply means sampling more initial particles from noise and evolving them under the learned field.

Overall, prior work indicates the promise of diffusion models for 3D generation, but also suggests room for improvement in explicitly handling the geometric nature of shape data. Next, we describe our vector field formulation that integrates diffusion with an explicit geometric prior of manifold attraction.

3 Vector Field-Based Diffusion Model

Our proposed method reframes the reverse diffusion process as a learnable vector field that operates in continuous 3D space and time. We first define the

forward (noising) process in the point cloud context, then formulate the reverse process and the neural vector field representation. We assume an underlying data distribution $p_{\text{data}}(x)$ supported (primarily) on a d-dimensional manifold $\mathcal{M} \subset \mathbb{R}^3$ (where d may be 1, 2, or 3). In practice, $\mathcal{M}$ could be, for example, a 2D surface representing an object, or a 1D curve (wire or skeleton structure), or the full 3D volume for solid objects.

3.1 Forward Diffusion Process

We define a continuous-time forward diffusion $x(0) \sim p_{\text{data}}$ and $x(T) \sim p_{\text{prior}}$ (with T large). For concreteness, one can use a Variance-Preserving SDE:

$$dx = -\tfrac{1}{2}\beta(t)\, x\, dt + \sqrt{\beta(t)}\, dw\,, \tag{2}$$

with $x \in \mathbb{R}^3$. Here w is a standard Wiener process in $\mathbb{R}^3$, and $\beta(t)$ is a non-negative schedule (e.g. linearly increasing to $\beta_{\max}$ by T). Intuitively, this SDE (when applied independently to each point coordinate) causes the point locations to undergo an Ornstein-Uhlenbeck process that drifts toward the origin while adding Gaussian noise. By choosing $\beta(t)$ appropriately, the marginal distribution $q_t(x)$ transitions from $q_0(x) = p_{\text{data}}(x)$ to an approximately isotropic Gaussian $q_T(x) \approx \mathcal{N}(0, \sigma^2 I)$ for large T. In discrete terms, we can consider N points $\{x_i\}_{i=1}^N$ from a shape and add noise in T small steps:

$$x_i(t + \Delta t) = \sqrt{1 - \beta \Delta t}\; x_i(t) + \sqrt{\beta \Delta t}\; \epsilon_i,$$

with $\epsilon_i \sim \mathcal{N}(0, I_3)$. This forward process is applied to all points in the cloud. Importantly, even if $x_i(0)$ lay exactly on a manifold $\mathcal{M}$, for any $t > 0$ the distribution $q_t(x)$ becomes full-dimensional (supported in $\mathbb{R}^3$) because Gaussian perturbation moves points off the manifold. By the final time T, the distribution is diffuse and no longer concentrated near $\mathcal{M}$.

One subtle consideration is that if $\mathcal{M}$ is lower-dimensional (say a 2D surface in 3D), p_{data} is a singular distribution. The diffusion as written will technically never exactly reach a Gaussian in finite T (since an OU process starting on a measure-zero set stays singular for any finite t). However, in practice one can start by slightly perturbing the data (e.g. adding a tiny initial noise) so that $q_\varepsilon(x)$ becomes absolutely continuous, or simply accept that for reasonably large T the distribution q_T is extremely close to Gaussian for practical purposes. For theoretical analysis, methods like *adding a small floor noise* are often assumed to handle singular supports:contentReference[oaicite:23]index=23. In our context, this issue is handled implicitly by the training procedure (score matching on denoised data), as done in prior score-based methods for surfaces [4].

3.2 Reverse Diffusion as a Time-Dependent Vector Field

The generative process aims to start from $p_T(x)$ (Gaussian noise) and produce a sample from $p_0(x)$ (data on the manifold) by following the reverse-time SDE or its deterministic counterpart. Rather than parameterizing the reverse step

as $p_\theta(x_{t-\Delta t}|x_t)$ or directly predicting noise, we parameterize the **drift vector field** of the probability flow ODE (1). That is, we learn a function $v_\theta(x,t)$ such that:

$$\frac{dx}{dt} = v_\theta(x,t), \qquad \text{with } x(T) \sim \mathcal{N}(0,\sigma^2 I) \text{ and } x(0) \sim p_{\text{data}}. \tag{3}$$

This $v_\theta(x,t)$ is intended to approximate the combined effect $f(t)x - \frac{1}{2}g(t)^2\nabla_x \log p_t(x)$ in Eq. (1). For the forward SDE (2), $f(t) = -\frac{1}{2}\beta(t)$ and $g(t) = \sqrt{\beta(t)}$, so the ideal reverse ODE is:

$$\frac{dx}{dt} = -\frac{1}{2}\beta(t)\, x - \frac{1}{2}\beta(t)\, \nabla_x \log p_t(x)\,. \tag{4}$$

The first term $-\frac{1}{2}\beta x$ is a deterministic contraction (pulling x toward 0) that we know analytically, while the second term involves the unknown score $\nabla_x \log p_t(x)$. In our parameterization, we let the network implicitly handle the entire vector field $v_\theta(x,t)$, which can learn to combine the contraction and score components. Alternatively, one could subtract the known $-\frac{1}{2}\beta x$ term and only learn the score part, but we find it simpler to let the model learn v_θ freely (the network can internally represent something akin to a scaled $-x$ plus additional components).

During training, we do not explicitly simulate the ODE. Instead, we train v_θ via a form of denoising score matching. At time t, we have a noised sample $x(t)$ which in expectation satisfies $x(t) = e^{-\frac{1}{2}\int_0^t \beta(s)ds}x(0) + \text{noise}$. The score $\nabla_x \log q_t(x)$ can be related to the conditional $q(x(0)|x(t))$. In practice, we optimize a reweighted objective that at a random $t \sim \mathcal{U}(0,T)$, the network's output $v_\theta(x(t),t)$ matches the *ideal vector field* $f(t)x(t) - \frac{1}{2}g(t)^2\nabla_x \log q_t(x(t))$. This is analogous to the standard practice of training $\epsilon_\theta(x_t,t)$ to predict the added noise, or s_θ to predict the score. We omit the lengthy derivation for brevity; it is similar to the derivations in [1,2]. The loss can be written as:

$$L(\theta) = \mathbb{E}_{t,x(0),\epsilon}\Big[\lambda(t)\,\big\|v_\theta(x(t),t) - \tilde{v}(x(t),t)\big\|^2\Big], \tag{5}$$

where $x(0) \sim p_{\text{data}}$, $\epsilon \sim \mathcal{N}(0,I)$, $x(t) = e^{-\frac{1}{2}\int_0^t \beta(s)ds}x(0) +$ (noise term) as given by the forward process, and $\tilde{v}(x(t),t)$ is the target vector field value (derived from the true score). $\lambda(t)$ is a weighting function (commonly $\lambda(t) = 1$ or related to SNR as in [1]). In implementation, $v_\theta(x,t)$ can be realized by a neural network that takes as input the point coordinate x (and possibly additional context like a shape label or latent, if conditional generation is desired) and the time t (typically encoded via sinusoidal embedding). The network outputs a 3D vector representing the velocity or direction in which to move the point at that time.

Crucially, our architecture treats each point independently in terms of coordinates: the vector field is a function on $\mathbb{R}^3$ (plus time). This means if we have multiple points forming a shape, we can evaluate v_θ at each point's location to get its velocity. The model does *not* require a fixed number of input points or any particular ordering, since it is fundamentally a field defined over continuous space. This is what grants the model arbitrary resolution: whether

we input 100 points or 100k points, the field can be evaluated at those points all the same (assuming sufficient compute). In practice, we do limit the number of points during training (for memory reasons) and can either train on a fixed size (e.g. 1024 points sampled per shape) or vary the number of points per training instance. The learned v_θ generalizes across these variations, focusing on local geometry.

3.3 Manifold Representation via Learned Vector Field

A major claim of our approach is that the model can learn to represent the underlying manifold $\mathcal{M}$ of the shape through its vector field. Intuitively, as $t \to 0$, we expect $v_\theta(x, t)$ to point every off-manifold sample x towards the manifold $\mathcal{M}$, effectively "pulling" points onto the surface. At $t = 0$, if x is exactly on $\mathcal{M}$ (and x is distributed as p_{data}), then no further change is needed, so ideally $v_\theta(x, 0) = 0$ for $x \in \mathcal{M}$ (the manifold is an *attractor* of the ODE). For x slightly off $\mathcal{M}$, $v_\theta(x, 0)$ should be a vector pointing toward $\mathcal{M}$ (e.g. roughly following the normal direction for a surface). In fact, the optimal vector field in the limit $t \to 0$ is related to the gradient of the log-density $p_{\text{data}}(x)$, which is known to blow up perpendicular to $\mathcal{M}$ if p_{data} is confined to $\mathcal{M}$. The following lemma makes this intuition more precise:

Lemma 1. *Let $\mathcal{M} \subset \mathbb{R}^3$ be a C^2 smooth d-dimensional manifold ($d < 3$ for a strict lower-dimensional case). Suppose p_{data} is supported on $\mathcal{M}$, and consider $p_{data,\sigma}$, the distribution of a point obtained by sampling $y \sim p_{data}$ on $\mathcal{M}$ and then adding Gaussian noise $\mathcal{N}(0, \sigma^2 I)$ to y. For small $\sigma > 0$, the score of this perturbed distribution, $\nabla_x \log p_{data,\sigma}(x)$, at a point x near $\mathcal{M}$ will be approximately perpendicular to $\mathcal{M}$ and pointing toward $\mathcal{M}$. In particular, if $\bar{x} = \Pi_{\mathcal{M}}(x)$ is the projection of x onto the manifold and $r = \|x - \bar{x}\|$ is the distance to $\mathcal{M}$, then for σ sufficiently small:*

$$\nabla_x \log p_{data,\sigma}(x) \;\approx\; -\frac{r}{\sigma^2}\, n(\bar{x})\,,$$

where $n(\bar{x})$ is the unit normal vector to $\mathcal{M}$ at $\bar{x}$. Thus, $-\nabla_x \log p_{data,\sigma}(x)$ is roughly a vector of length r/σ^2 pointing from x straight towards the manifold (along the normal).

Proof Sketch. Since p_{data} is concentrated on $\mathcal{M}$, for a small noise σ most probability mass of $p_{\text{data},\sigma}$ lies in a thin tubular neighborhood around $\mathcal{M}$. Locally (in a neighborhood of a given point $\bar{x} \in \mathcal{M}$), we can approximate $\mathcal{M}$ by its tangent plane at $\bar{x}$. In coordinates where $\bar{x}$ is the origin and the tangent plane is, say, the xy-plane (for a surface), the density $p_{\text{data},\sigma}(x)$ for $x = (u, v, w)$ (with w the coordinate normal to the manifold) is proportional to $p_{\text{data}}(\bar{x})$ times the noise density in the normal direction: $\propto \exp(-w^2/(2\sigma^2))$ (times a slowly varying factor in-plane). Thus $\log p_{\text{data},\sigma}(x) \approx \text{const} - \frac{w^2}{2\sigma^2}$. Its gradient is $-\frac{w}{\sigma^2}\mathbf{e}_w$, which in original terms is $-\frac{(x-\bar{x})}{\sigma^2} \cdot n(\bar{x})$ in the normal direction. The magnitude is $\frac{|w|}{\sigma^2} = \frac{r}{\sigma^2}$ (since $r \approx |w|$ for small w). This yields the stated result. □

Lemma 1 aligns with the intuition that the score (gradient of log-density) of a distribution narrowly concentrated around a manifold will point inward, toward the manifold. In our diffusion model, $v_\theta(x,t)$ is trained to approximate (a scaled version of) $-\nabla_x \log p_t(x)$ for each t. As $t \to 0$, $p_t(x)$ approaches $p_{\text{data},\sigma}$ for very small σ. Thus, $v_\theta(x, 0^+)$ should learn a similar behavior: pointing any off-surface point back onto the surface. In other words, the learned vector field near $t = 0$ is effectively encoding the manifold's geometry via its normal vectors and pulling strength.

This gives a geometric picture of the generative process: starting from diffuse noise at $t = T$, the vector field v_θ gradually organizes the points, and in the final stages (small t) it acts like a projection mechanism that ensures points end up on the manifold $\mathcal{M}$. The manifold itself emerges as an attractor of the ODE $dx/dt = v_\theta(x,t)$ as $t \to 0$. Ideally, $\mathcal{M}$ is a set of fixed points of the final field $v_\theta(x,0)$ (since points on $\mathcal{M}$ have $v_\theta = 0$ and remain on $\mathcal{M}$).

3.4 Arbitrary Resolution and Permutation Invariance

A noteworthy feature of our model is that it does not assume a fixed size or ordering for the point cloud. The generation of N points can be described as sampling N i.i.d. points from $p_T(x) \approx \mathcal{N}(0, \sigma^2 I)$ and evolving each according to Eq. (3) (with shared v_θ). There is no parameter in v_θ that depends on N or on any specific point indices; if we generate more points, we are effectively drawing more samples from the underlying continuous distribution. In the limit of $N \to \infty$, the set of generated points can densely cover the manifold, providing a high-resolution representation of the shape. In practice, of course, N will be finite, but we can choose N at inference to balance detail and computational cost.

We formalize the resolution-invariance property as follows:

Proposition 1. *Suppose $v_\theta(x,t)$ is the learned vector field that perfectly recovers the true reverse dynamics of the diffusion (in the limit of infinite capacity and training). For any $N \in \mathbb{N}$, consider N sample trajectories $\{x_i(t)\}_{i=1}^N$ following $dx_i/dt = v_\theta(x_i,t)$ with independent initial samples $x_i(T) \sim \mathcal{N}(0, \sigma^2 I)$. Then the distribution of the set $\{x_i(0)\}_{i=1}^N$ is an i.i.d. sample of size N from p_{data}. As $N \to \infty$, the generated point set converges (in the appropriate sense) to the underlying continuous shape (manifold) distribution. In other words, increasing N improves the approximation of the continuous shape by the point cloud, without changing the model.*

Sketch. Under the assumption that v_θ exactly reproduces the true reverse process, each $x_i(0)$ is a sample from p_{data}, independently. Thus any number of points N will yield N independent draws from p_{data}. For large N, by the law of large numbers the empirical distribution converges to p_{data}, meaning the union of points densely covers regions where p_{data} has support (which is $\mathcal{M}$). In practice v_θ is approximate, but if it is sufficiently accurate, adding more points will continue to sample the learned distribution in an i.i.d. manner. The model itself imposes no limit on N, aside from computational constraints. □

Proposition 1 underscores that our method is inherently *permutation-invariant* and scalable in the number of points. Many prior point cloud generation methods (especially GAN or flow based) produce a fixed number of points by design (e.g. network outputs $3 \times N$ numbers representing N points). Some diffusion models like [3] also fix N during training and generation. While one can always add interpolation or upsampling post-hoc, the generative model itself in those cases is tied to a specific resolution. In contrast, our diffusion model has no such restriction: the notion of a shape is encapsulated in the vector field v_θ, and the number of points is simply how many samples we draw to represent that shape.

It is worth noting that if the training point clouds have a fixed size N_{train}, our model still learns a continuous distribution (implicitly). At test time, one can choose $N \neq N_{\text{train}}$ freely. Empirically, one might train with a relatively small N_{train} for efficiency, and then sample a much larger N for a finer result, similar to the PointInfinity approach:contentReference[oaicite:24]index=24. In our experiments (if any were discussed), we verify that using a higher point count at inference yields more detailed surfaces, consistent with Proposition 1.

3.5 Inputs During Training

- **Clean point sample** $x_0 \in \mathbb{R}^3$: A sample from the original point cloud data, typically lying on a manifold $\mathcal{M}$.
- **Time step** $t \in [0, T]$: Randomly sampled diffusion time, either from a continuous or discretized interval.
- **Noised point** x_t: Computed using the forward diffusion process:

$$x_t = \sqrt{\alpha(t)}\, x_0 + \sqrt{1 - \alpha(t)}\, \epsilon, \quad \epsilon \sim \mathcal{N}(0, I_3)$$

  where $\alpha(t)$ is a time-dependent noise schedule.

3.6 Output and Target During Training

The model learns a vector field $v_\theta(x_t, t)$ that approximates the reverse-time dynamics:

$$v_\theta(x_t, t) \approx \text{direction pointing from } x_t \text{ to } x_0.$$

This can correspond to either:

- **Score-based training:** where $v_\theta(x_t, t)$ approximates the score $\nabla_{x_t} \log p_t(x_t)$.
- **Vector field training:** where $v_\theta(x_t, t)$ approximates a drift term in the reverse-time ODE, often taking the form:

$$v_\theta(x_t, t) = \frac{x_0 - x_t}{\sigma(t)^2} \quad \text{(or some scaled variant).}$$

3.7 Summary

- **Input:** Noised point x_t and time t.
- **Network Output:** Vector field value $v_\theta(x_t, t)$.
- **Training Target:** Vector from x_t toward x_0, used for regression or score matching.

This setup allows the network to learn how to guide noisy samples toward the clean data manifold at every timestep.

4 Comparisons and Discussion

In this section, we compare the proposed vector field diffusion model with standard DDPMs and other related approaches, highlighting conceptual and practical differences.

4.1 Relation to Standard Diffusion Models (DDPMs)

Our model can be seen as a variant of a DDPM specialized for point cloud data and geometric manifolds. Standard DDPMs (as in [1]) typically train a UNet to predict added noise ϵ or to directly predict x_0 from a noised x_t. In contrast, we train a network to predict a *directional increment* $v_\theta(x_t, t)$ for each point. This is closely related to predicting the score (since v_θ includes a component proportional to the score). One advantage of the vector field view is interpretability: the network's output at a point x can be interpreted as indicating where that point should move next. This is particularly meaningful for shape data, as we can interpret $v_\theta(x, t)$ as pushing points toward forming a coherent surface. In a conventional DDPM, the intermediate denoised outputs might be less constrained and could potentially produce less structured intermediate states, whereas a vector field that consistently pushes towards a surface might maintain more geometric coherence throughout the reverse process.

Moreover, our approach naturally integrates the drift term $-\frac{1}{2}\beta x$ from the forward SDE, whereas a typical DDPM might not explicitly account for it (since one can absorb the linear term into the network's learned function by augmenting input channels or through training). By learning the full drift directly, we conceptually unify the known physics of the diffusion (the linear contraction) with the learned component (the score). This is a subtle difference, but it could lead to improved stability or accuracy, as the network is not burdened with representing the trivial $-x$ drift part (if we chose to explicitly remove it, though in our implementation we did not, letting the network discover it on its own).

In terms of quality, a baseline DDPM for point clouds (e.g. [3]) already performs well, so our aim is not necessarily to dramatically improve upon their quantitative metrics, but rather to provide a model that has better theoretical properties (manifold support, resolution freedom) and more geometric interpretability. That said, focusing the network on the gradient field might also

help training data efficiency, since it encourages capturing the structure of $p(x)$ more directly than, say, an autoencoder latent approach. Our method is also compatible with conditional generation: one can condition v_θ on a class label or a text embedding, similarly to conditional DDPMs, by concatenating those to the input or using cross-attention.

4.2 Limitations of Invertible Flows and Advantages of Diffusion

Normalizing flows require the latent and data spaces to have equal dimensionality and rely on smooth, invertible transformations. As discussed, this fundamentally limits their ability to model data on lower-dimensional manifolds:contentReference[oaicite:25]index=25. We formalize this limitation as a proposition:

Proposition 2. *(No exact manifold modeling by flows.) Let $p_Z(z)$ be an absolutely continuous prior in $\mathbb{R}^n$ (e.g. a standard Gaussian) and let $f_\phi : \mathbb{R}^n \to \mathbb{R}^n$ be any C^1 diffeomorphism parameterized by ϕ. Suppose $p_X(x)$, the pushforward of p_Z through f_ϕ, is supported on a d-dimensional manifold $\mathcal{M} \subset \mathbb{R}^n$ with $d < n$. Then p_X cannot be an absolutely continuous probability density on $\mathbb{R}^n$; it is singular. This implies the likelihood $\log p_X(x)$ is $-\infty$ almost everywhere off $\mathcal{M}$ and undefined on $\mathcal{M}$ itself (infinite spike). Therefore, no smooth invertible f_ϕ can map an absolutely continuous latent p_Z exactly to p_X—at best it can approximate p_X with a sequence of distributions that add small thickness to $\mathcal{M}$.*

Idea of Proof. If f_ϕ is invertible and smooth, and p_Z has a density, then $p_X(x) = p_Z(f_\phi^{-1}(x)) \,|\det Df_\phi^{-1}(x)|$. Wherever f_ϕ^{-1} is defined (which is everywhere in $\mathbb{R}^n$) and smooth, $p_X(x)$ will be a smooth density as well, except possibly where the Jacobian determinant is 0. But f_ϕ being a diffeomorphism means the Jacobian is nonzero everywhere, hence $p_X(x)$ is a smooth, nonzero density on all of $\mathbb{R}^n$. This contradicts the assumption that p_X is supported only on $\mathcal{M}$ (a set of measure zero in $\mathbb{R}^n$). Hence no such f_ϕ exists that yields an exact manifold-supported distribution. Flow models can only converge to a sequence of densities that concentrate around $\mathcal{M}$ but never exactly on it. □

Proposition 2 aligns with the statement from normalizing flow literature that flows "at best learn a superset of the data manifold":contentReference[oaicite:26]index=26. Empirically, PointFlow [5] and similar models tend to generate point clouds that have a slight noise or dispersion around the true surfaces, due to this necessity of an absolutely continuous output density. Some post-processing or specialized training (like noise injection during training as in [9]) can mitigate this, but it complicates the pipeline.

Diffusion models circumvent this by trading off invertibility for stochasticity. The forward diffusion is not invertible, so the reverse generative mapping is not a fixed diffeomorphism from $\mathbb{R}^n$ to $\mathbb{R}^n$, but rather a dynamic process that can collapse entropy. Our vector field, being time-dependent, is not the

gradient of a globally invertible map at final time (indeed, it may not correspond to any single transform from noise to data, especially if we consider the stochastic interpretation). Therefore, it can afford to "lose volume" in the sense of concentrating mass on a lower-d manifold without violating any constraints. The learned vector field effectively discovers how to remove the extra degrees of freedom (the normal direction noise) as $t \to 0$, by pushing points in those directions towards the manifold.

Another benefit is the **smoothness of the learned manifold**. By using a continuous diffusion, we inherently produce smoothly varying outputs. If the data manifold $\mathcal{M}$ is smooth, we expect the model to capture that smoothness. In practice, methods like ShapeGF have noted that their learned gradient fields can be used to reconstruct smooth surfaces via methods like Poisson surface reconstruction:contentReference[oaicite:27]index=27. Our approach similarly could allow extracting an implicit surface: one could train v_θ, then at $t = 0$ interpret $v_\theta(x, 0)$ as (approximately) $-\frac{1}{2}\beta(0)\nabla \log p_{\text{data}}(x)$; integrating this field or finding its potential function might yield an implicit surface representation $F(x) = \text{const}$ whose zero level is the generated shape. This is an intriguing direction for future exploration (connecting generative models with classical surface reconstruction).

4.3 Comparison with Prior Point Cloud Diffusion Methods

As discussed in the background, our approach shares motivations with ShapeGF [4], which explicitly learned the gradient field for shape generation. ShapeGF can be seen as a special case of a score-based model (it employed a multi-scale noise conditioning similar to score matching):contentReference[oaicite:28]index=28:contentReference[oaicite:29]index=29 Our work extends this idea by situating it in the formal diffusion framework (with a well-defined forward SDE and a reverse vector field) and by emphasizing the ability to handle arbitrary manifold dimensions. ShapeGF primarily demonstrated results on surfaces (2D manifolds); our framework is equally applicable to curves or volumes. Additionally, our model is trained end-to-end via the diffusion objective (5), whereas ShapeGF used a specific stochastic gradient ascent procedure at sampling time. In principle, both approaches should learn similar score functions; however, the diffusion formulation allows using advanced techniques from the diffusion literature (like improved samplers, variational bounds for likelihood estimation, etc.). It also seamlessly integrates with conditioning and other diffusion tricks (e.g. classifier-free guidance, if one wanted to guide shape generation with attributes).

Luo & Hu's point cloud DDPM [3] differs from our model in that it operates on the entire point cloud as a fixed-size vector (albeit order-invariant features were used). Their network had to accommodate the permutation invariance and point interactions, possibly via attention or by conditioning on a latent. Our approach, working pointwise, does not explicitly model inter-point interactions except through the global learned field (which implicitly encodes shape-level information). One might wonder: without explicit interaction, how do points

"know" to form a coherent shape? The answer lies in the fact that $v_\theta(x, t)$ is a global function of space: during training it sees many examples of points in certain configurations (e.g. points that are part of some airplane wing vs points isolated far away) and learns to move them differently. For instance, a point that is far from any other points (in a generated sample) will likely have a low probability under p_t and the model will drive it towards where the mass is (closer to other points or towards a known shape area). While we did not explicitly include neighbor information, the network can infer some global context from the absolute position x and time t – because at a given time, the distribution $p_t(x)$ of points has known statistics (initially nearly Gaussian, later increasingly concentrated around shape structures). In practice, one could enhance our model by feeding in additional global context (like a latent code representing the shape identity). For an unconditional model, the diversity of shapes must come purely from the randomness in initial noise. The network v_θ essentially encodes a generative program that transforms any Gaussian scatter into a plausible shape.

Latent diffusion approaches like LION [8] take a different path by compressing the point cloud into a latent (using a point VAE) and then diffusing in that latent space. While effective in reducing computational load, such methods may sacrifice some fine geometric detail and impose a bottleneck. Our method works directly in data space, which is more costly but preserves full detail. With recent advances in neural network efficiency and continuous data structures, we believe operating in the continuous domain is viable even at reasonably high point counts. Additionally, one could combine ideas: e.g. use a coarse latent diffusion to get a rough shape, then refine with our vector field diffusion at the fine scale for details, potentially giving a two-stage model.

Finally, a recent development by Ye *et al.* [6] introduced *First Hitting Diffusion Models* (FHDM). They handle data confined to a domain (including manifolds or discrete structures) by running diffusion until the process hits the domain for the first time. For example, to generate points on a sphere, they diffuse a point in $\mathbb{R}^3$ until it hits the sphere surface. While conceptually elegant, this approach requires defining the domain explicitly (e.g. known manifold equation) or learning an indicator function to know when to stop. In our case, the manifold is learned implicitly via the vector field rather than treated as a hard constraint. An advantage is that we do not need a prior definition of $\mathcal{M}$; a drawback is that we rely on the network to learn $\mathcal{M}$ from data (which it does via score matching). FHDM provides theoretical speed-ups and instance-adaptive diffusion times, whereas our model still uses a fixed time horizon (or fixed number of steps). These are complementary perspectives: one could possibly integrate a hitting-time criterion with our learned field to decide when a point is "close enough" to the surface to stop diffusing it. We leave such extensions to future work.

5 Conclusion

We presented a vector field-based diffusion model tailored for point cloud generation on arbitrary-dimensional manifolds in 3D. By learning a time-dependent vector field during denoising, our model captures the geometric structure of data and overcomes key limitations of previous approaches. The model can generate shapes as lower-dimensional manifolds with essentially infinite resolution, since it permits sampling arbitrarily many points. Theoretical analysis confirmed that unlike invertible flows, diffusion processes can target singular manifold distributions, and that the learned score field provably aligns with manifold normals to pull points onto the surface. We situated our approach in the context of related work, drawing connections to score-based methods and highlighting empirical evidence that supports our design (such as improved fidelity and smoothness of generated surfaces).

This work bridges the gap between deep generative modeling and classical geometry processing by leveraging the notion of a vector field that "sculpts" random points into a shape. Beyond point clouds, this idea could be extended to other geometric representations (meshes or implicit fields) and other data modalities where support lies on a known manifold (e.g. human pose data on rotation manifolds). Future work will explore conditioning the vector field on high-level inputs (like images or text prompts) to allow controllable shape generation, and improving the efficiency of the approach (since evaluating a large number of points through many steps can be computationally heavy, techniques like importance sampling of points or adaptive step ODE solvers could be beneficial).

References

[1] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020.

[2] Y. Song *et al.* Score-based generative modeling through stochastic differential equations. *ICLR*, 2021.

[3] S. Luo and W. Hu. Diffusion probabilistic models for 3d point cloud generation. *CVPR*, 2021.

[4] R. Cai *et al.* Learning gradient fields for shape generation. *ECCV*, 2020.

[5] G. Yang *et al.* Pointflow: 3d point cloud generation with continuous normalizing flows. *ICCV*, 2019.

[6] M. Ye, L. Wu, and Q. Liu. First hitting diffusion models for generating manifold, graph and categorical data. *NeurIPS*, 2022.

[7] Z. Huang *et al.* PointInfinity: Resolution-invariant point diffusion models. *ICCV*, 2023.

[8] X. Zeng *et al.* Lion: Latent point diffusion models for 3d shape generation. *NeurIPS*, 2022.

[9] J. Postels *et al.* ManiFlow: Implicitly representing manifolds with normalizing flows. *3DV*, 2022.

[10] Y. Lipman *et al.* Flow matching for generative modeling. *NeurIPS*, 2022.