Poisson Flow Generative Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

We propose a new "Poisson flow" generative model that maps a uniform distribution on a high-dimensional hemisphere into any data distribution. We interpret the data points as electrical charges on the $z = 0$ hyperplane in a space augmented with an additional dimension z, generating a high-dimensional electric field (the gradient of the solution to Poisson equation). We prove that if these charges flow upward along electric field lines, their initial distribution in the $z = 0$ plane transforms into a distribution on the hemisphere of radius r that becomes *uniform* in the $r \to \infty$ limit. To learn the bijective transformation, we estimate the normalized field along the trajectories of its associated forward ODE. For sampling, we devise a backward ODE that is anchored by the physically meaningful additional dimension: the samples hit the (unaugmented) data manifold when the z reaches zero. Experimentally, the backward ODE of Poisson flow is on par with the state-of-the-art score-based MCMC and SDE approaches while achieving $10\times$ to $20\times$ acceleration on image generation tasks. The backward ODE also outperforms popular ODE methods using the same NCSNv2 architecture [26] and demonstrates superior stability. Finally, we showcase the utility of the invertibility of Poisson flow on likelihood evaluation, meaningful latent representations and adaptive sampling.

1 Introduction

Deep generative models are a prominent approach for data generation, and have been used to produce high quality samples in image [1], text [2] and audio [28], as well as improve semi-supervised learning [17], domain generalization [21] and imitation learning [14]. However, current deep generative models also have limitations, such as unstable training objectives (GANs [1, 11]) and low sample quality (VAEs [18], normalizing flows [5]). Although recent advances on diffusion [15] and scored-based models [27] achieve comparable sample quality to GAN's without adversarial training, these models have a slow stochastic sampling process, or an unstable deterministic reverse-time ODE.

We present a new "Poisson flow" generative model (**PFGM**), exploiting a remarkable physics fact that generalizes to N dimensions. As illustrated in Fig. 1(a), motion in a viscous fluid transforms any planar charge distribution into a uniform angular distribution. Specifically, we interpret N-dimensional data points $\mathbf{x}$ (images, say) as positive electric charges in the $z = 0$ plane of an $N + 1$-dimensional space (see Fig. 1(a)) filled with a viscous liquid (say honey). A positive charge with $z > 0$ will be repelled by the other charges and move in the direction of their repulsive force, eventually crossing an imaginary hemisphere of radius r. We show that, remarkably, if the the original charge distribution is let loose just above $z = 0$, this law of motion will cause a *uniform* distribution for their hemisphere crossings in the $r \to \infty$ limit.

Our Poisson flow generative process reverses the forward process: we generate a uniform distribution of negative charges on the hemisphere, then track their motion back to the $z = 0$ plane, where they will be distributed as the training data. A Poisson flow is similar to a normalizing flow [5, 16] in the sense that it maps between an arbitrary distribution and an easily sampled one: in the former case an N-dimensional normal distribution and in the latter case a uniform distribution on an N-dimensional hemisphere. In practice, we implement the Poisson flow by solving a pair of forward/backward

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

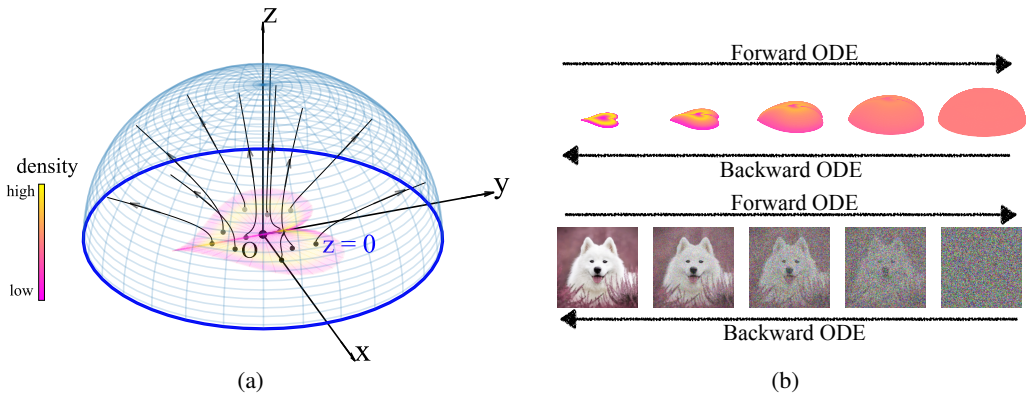


Figure 1: **(a)** 3D Poisson field trajectories for a heart-shaped distribution **(b)** The bijections between the heart-shaped distribution (**top**) or (augmented) data distribution (**bottom**) and an uniform distribution on hemisphere. The forward/backward ODEs pertain to the Poisson field.

ordinary differential equations (ODEs) induced by the electric field (Fig. 1(b)) given by the N-dimensional version of Coulombs law (the gradient of the solution to the Poisson's equation with the data as sources). We will interchangeably refer to this gradient as the *Poisson field*, since electric fields normally refer to the special case $N = 3$.

The proposed generative model PFGM has a stable training objective and produces high-quality samples on par with the the score-based MCMC or SDE methods [26, 27] on the same architecture. PFGM has two advantanges compared to score-based methods [26, 27]. First, PFGM's deterministic backward ODE sampler achieves faster sampling speeds than the score-based stochastic sampler [26, 27]. Second, our backward ODE exhibits greater stability than other ODE baselines, *e.g.*, the backward ODEs of VE/VP SDEs [27]. The rationale is that the time variables in these ODE baselines are strongly correlated with the sample norms because of the Gaussian perturbation kernels during training, resulting in a less error-tolerant inference-time ODE simulation. In contrast, the tie between the anchored variable and the sample norm in the backward ODE of PFGM is much weaker.

Experimentally, we show that PFGM has competitive performance with state-of-the-art method on the NCSNv2 architecture on CIFAR-10, and outperforms all the baselines on CelebA 64×64 dataset in terms of FID score [12]. Meanwhile, the backward ODE gives $10\times$ to $20\times$ speed up compared to stochastic score-based samplers across datasets. Notably, the backward ODE in PFGM is the *only* ODE-based sampler that can produce decent samples on its own on the NCSNv2 architecture, while other ODE baselines fail without correction. PFGM still performs better than other ODE-based samplers with correctors. We further showcase the utility of the invertible forward/backward ODE of the Poisson field on likelihood evaluation, meaningful latent representations, and adaptive sampling.

2 Background and Related works

Poisson equation Let $\mathbf{x} \in \mathbb{R}^N$ and $\rho(\mathbf{x}) : \mathbb{R}^N \to \mathbb{R}$ be a *source* function. The Poisson equation is

$$\nabla^2 \varphi(\mathbf{x}) = -\rho(\mathbf{x}), \tag{1}$$

where $\varphi(\mathbf{x}) : \mathbb{R}^N \to \mathbb{R}$ is called the *potential function*, and $\nabla^2 \equiv \sum_{i=1}^N \frac{\partial^2}{\partial x_i^2}$ is the Laplacian operator. It is usually helpful to define the gradient field $\mathbf{E}(\mathbf{x}) = -\nabla\varphi(\mathbf{x})$ and rewrite the Poisson equation as $\nabla \cdot \mathbf{E} = \rho$, known in physics as Gauss's law [10]. The Poisson equation is widely used in physics, giving rise to Newton's gravitational theory [8] and the electrostatic theory [10], when $\rho(\mathbf{x})$ is interpreted as mass density or electric charge density, respectively. $\mathbf{E}$ is the N-dimensional analog of the electric field. The Poisson equation Eq. (1) (with zero boundary condition at infinity) admits a simple integral solution:

$$\varphi(\mathbf{x}) = \int G(\mathbf{x}, \mathbf{y})\rho(\mathbf{y})d\mathbf{y}, \quad G(\mathbf{x}, \mathbf{y}) = \frac{1}{S_{N-1}(1)} \frac{1}{||\mathbf{x} - \mathbf{y}||^{N-2}}, \tag{2}$$

where $G(\mathbf{x}, \mathbf{y})$ is the Green's function, and $S_{N-1}(1)$ is a geometric constant representing the surface area of the unit $(N-1)$-sphere[1]. The negative gradient field of $\varphi(\mathbf{x})$, referred as *Poisson field* of the source ρ, is

$$\mathbf{E}(\mathbf{x}) = -\nabla\varphi(\mathbf{x}) = -\int \nabla_{\mathbf{x}} G(\mathbf{x}, \mathbf{y})\rho(\mathbf{y})d\mathbf{y}, \quad \nabla_{\mathbf{x}} G(\mathbf{x}, \mathbf{y}) = -\frac{N-2}{S_{N-1}(1)} \frac{\mathbf{x}-\mathbf{y}}{||\mathbf{x}-\mathbf{y}||^N}. \tag{3}$$

Qualitatively, the Poisson field $\mathbf{E}(\mathbf{x})$ points away from sources, or equivalently $-\mathbf{E}(\mathbf{x})$ points towards sources, as illustrated in Fig. 1(a). It is straightforward to check that when $\rho(\mathbf{x}) \to \delta(\mathbf{x}-\mathbf{y})$, we get $\varphi(\mathbf{x}) \to G(\mathbf{x}, \mathbf{y})$ and $\mathbf{E}(\mathbf{x}) \to -\nabla_{\mathbf{x}} G(\mathbf{x}, \mathbf{y})$. This implies that $G(\mathbf{x}, \mathbf{y})$ and $-\nabla_{\mathbf{x}} G(\mathbf{x}, \mathbf{y})$ can be interpreted as the potential function and the gradient field generated by a unit point source, *e.g.*, a point charge, located at $\mathbf{y}$. When $\rho(\mathbf{x})$ takes general forms but has bounded support, simple asymptotics exist for $||\mathbf{x}|| \gg ||\mathbf{y}||$. To the lowest order, $\mathbf{E}(\mathbf{x}) = \nabla_{\mathbf{x}} G(\mathbf{x}, \mathbf{y})|_{\mathbf{y}=\mathbf{0}} \sim \mathbf{x}/||\mathbf{x}||^N$ behaves as if it were generated by a unit point source at $\mathbf{y} = 0$. In physics, the power law decay is considered to be long-range (compared to exponential decay) [10].

Particle dynamics in a Poisson field The Poisson field has two associated forward and backward ODEs. To obtain physical intuition, we imagine a particle with mass $m = 1$ and charge $q = 1$ pushed by an electric field $\mathbf{E}$ through an an extremely viscous fluid (e.g., honey), causing its velocity (as opposed to its acceleration as would be the case in vacuum) to be proportional to $\mathbf{E}$.

$$[q = 1, \text{forward ODE}] \quad \frac{d\mathbf{x}}{dt} = \mathbf{E}(\mathbf{x}), \quad [q = -1, \text{backward ODE}] \quad \frac{d\mathbf{x}}{dt} = -\mathbf{E}(\mathbf{x}). \tag{4}$$

which trace the electric lines in the forward and backward way, respectively. The dynamics is also *rescalable* in the sense that the particle trajectory remains the same for $\frac{d\mathbf{x}}{dt} = \pm f(\mathbf{x})\mathbf{E}(\mathbf{x})$ when $f(\mathbf{x}) > 0$, because the time rescaling $dt \to f(\mathbf{x}(t))dt$ recovers $\frac{d\mathbf{x}}{dt} = \pm\mathbf{E}(\mathbf{x})$. Note that the dynamics defined in Eq. (4) is stiff due to the power law factor in the denominator in Eq. (3), posing computational challenges. Luckily the rescalablility allows us to rescale $\mathbf{E}(\mathbf{x})$ properly to get new ODEs (formally defined later in Section 3.3) that are more amenable for sampling.

Generative Modeling via ODE Generative modeling can be done by transforming a base distribution to a data distribution via mappings defined by ODEs. The ODE-based samplers allow for adaptive sampling, exact likelihood evaluation and modeling of continuous-time dynamics [4, 27]. Previous works broadly fall into two lines. [4, 3] introduce a continues-time normalizing flow model that can be trained with maximum likelihood by the instantaneous change-of-variables formula [4]. For sampling, they directly integrate the learned invertiable mapping over time. Another work [27] unifies the scored-based model [25, 26] and diffusion model [15] into a general diffusion process, and uses the reverse-time ODE of the diffusion process for sampling. They show that the reverse-time ODE produces high quality samples with improved architecture.

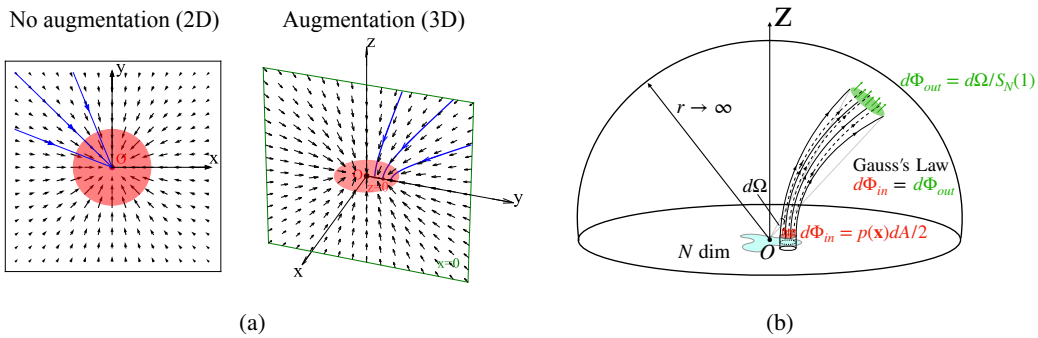
3 Poisson Flow Generative Models

In this section, we start with the properties of the Poisson flow in the augmented space and show how to draw samples from the data distribution by following the backward ODE of the Poisson flow (Section 3.1). We then discuss how to actually learn a normalized Poisson field from data samples through simulations of the forward ODE (Section 3.2) and present an equivalent backward ODE that allows for exponentially decay on z (Section 3.3).

3.1 Augment the data with additional dimension

We wish to generate samples $\mathbf{x} \in \mathbb{R}^N$ from a distribution $p(\mathbf{x})$ supported on a bounded region. We may set the source $\rho(\mathbf{x}) = p(\mathbf{x})$ and compute the resulting gradient field $\mathbf{E}(\mathbf{x})$ from Eq. (3). Since $-\mathbf{E}(\mathbf{x})$ points towards sources, the backward ODE $d\mathbf{x}/dt = -\mathbf{E}(\mathbf{x})$ will take samples close to the sources. One may naively hope that the backward ODE is a generative model that recovers $p(\mathbf{x})$. Unfortunately, the backward ODE has the problem of mode collapse. We illustrate this phenomenon with a 2D uniform disk. The reverse Poisson field $-\mathbf{E}(\mathbf{x})$ on the 2D (x, y)-plane points towards the center of the disk O (Fig. 2(a) left), so all particle trajectories (blue lines) will eventually hit O. If we instead add an additional dimension z (Fig. 2(a) right), particles can hit different points on the disk and faithfully recover the data distribution.

[1] The N-sphere with radius r is defined as $\{\mathbf{x} \in \mathbb{R}^{N+1}, ||\mathbf{x}|| = r\}$



(a) (b)

Figure 2: **(a)** Poisson field (black arrows) and particle trajectories (blue lines) of a 2D uniform disk (red). **Left** (no augmentation, 2D): all particles collapse to the disk center. **Right** (augmentation, 3D): particles hit different points on the disk. **(b)** Proof idea of Theorem 1. By Gauss's Law, the outflow flux $d\Phi_{out}$ equals the inflow flux $d\Phi_{in}$. The factor of two in $p(\mathbf{x})dA/2$ is due to the symmetry of Poisson fields in $z < 0$ and $z > 0$.

Consequently, instead of solving the Poisson equation $\nabla^2\varphi(\mathbf{x}) = -p(\mathbf{x})$ in the original data space, we solve the Poisson equation in an augmented space $\tilde{\mathbf{x}} = (\mathbf{x}, z) \in \mathbb{R}^{N+1}$ with an additional variable $z \in \mathbb{R}$. We augment the training data $\tilde{\mathbf{x}}$ in the new space by setting $z = 0$ such that $\tilde{\mathbf{x}} = (\mathbf{x}, 0)$. As a consequence, the data distribution in the augmented space is $\tilde{p}(\tilde{\mathbf{x}}) = p(\mathbf{x})\delta(z)$, where δ is the Dirac delta function. By Eq. (3), the Poisson field by solving the new Poisson equation $\nabla^2\varphi(\tilde{\mathbf{x}}) = -\tilde{p}(\tilde{\mathbf{x}})$ has an analytical form:

$$\forall \tilde{\mathbf{x}} \in \mathbb{R}^{N+1}, \mathbf{E}(\tilde{\mathbf{x}}) = -\nabla\varphi(\tilde{\mathbf{x}}) = \frac{N-1}{S_N(1)} \int \frac{\tilde{\mathbf{x}} - \tilde{\mathbf{y}}}{||\tilde{\mathbf{x}} - \tilde{\mathbf{y}}||^{N+1}} \tilde{p}(\tilde{\mathbf{y}}) d\tilde{\mathbf{y}} \tag{5}$$

The associated forward/backward ODEs of the Poisson field are $d\tilde{\mathbf{x}}/dt = \mathbf{E}(\tilde{\mathbf{x}}), d\tilde{\mathbf{x}}/dt = -\mathbf{E}(\tilde{\mathbf{x}})$. Intuitively, theses ODEs uniquely define trajectories of particles between the $z = 0$ hyperplane and an enclosing hemisphere (cf. Fig. 1(a)). In the following theorem, we show that the backward ODE defines a transformation between the uniform distribution on an infinite hemisphere and the data distribution $\tilde{p}(\tilde{\mathbf{x}})$ in the $z = 0$ plane. We present the formal proof to Appendix A, illustrated by Fig. 2(b). The proof is based on the idea that when the radius of hemisphere $r \to \infty$, the data distribution $\tilde{p}(\tilde{\mathbf{x}})$ can be effectively viewed as a delta distribution at origin. Consequently, the Poisson field points in the radial direction at $r \to \infty$, perpendicular to $S_N^+(r)$ (Green arrows in Fig. 2(b)).

Theorem 1. *Suppose particles are sampled from a uniform distribution on the upper ($z > 0$) half of the sphere of radius r and evolved by the backward ODE $\frac{d\tilde{\mathbf{x}}}{dt} = -\mathbf{E}(\tilde{\mathbf{x}})$ until they reach the $z = 0$ hyperplane, where the Poisson field $\mathbf{E}(\tilde{\mathbf{x}})$ is generated by the source $\tilde{p}(\tilde{\mathbf{x}})$. in the $r \to \infty$ limit, this process generates a particle distribution $\tilde{p}(\tilde{\mathbf{x}})$, i.e., a distribution $p(\mathbf{x})$ in the $z = 0$ hyperplane.*

Proof sketch. Suppose the flux of the backward ODE connects a solid angle $d\Omega$ (on $S_N^+(r)$) with an area dA (on $\text{supp}(p(\mathbf{x}))$), and the stationary distribution on $\text{supp}(p(\mathbf{x}))$ by $q(\mathbf{x})$. According to Gauss's law, the outflow flux $d\Phi_{out} = d\Omega/S_N(1)$ on the hemisphere (Green arrows in Fig. 2(b)) equals the inflow flux $d\Phi_{in} = p(\mathbf{x})dA/2$ on $\text{supp}(p(\mathbf{x}))$ (Red arrows in Fig. 2(b)). Furthermore, particle conservation gives $q(\mathbf{x})dA = \mathcal{U}(S_N^+(r))r^N d\Omega = 2d\Omega/S_N(1)$. Together, we conclude that the final distribution in the $z = 0$ plane is $q(\mathbf{x}) = p(\mathbf{x})$. □

The theorem states that starting from an infinite hemisphere, one can recover the data distribution $\tilde{p}$ by following the inverse Poisson field $-\mathbf{E}(\tilde{\mathbf{x}})$. The property allows generative modeling by following the Poisson flow of $\nabla^2\varphi(\tilde{\mathbf{x}}) = -\tilde{p}(\tilde{\mathbf{x}})$.

3.2 Learning the normalized Poisson Field

Given a set of training data $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^n$ i.i.d sampled from the data distribution $p(\mathbf{x})$, we define the empirical version of the Poisson field (Eq. (5)) as follows:

$$\hat{\mathbf{E}}(\tilde{\mathbf{x}}) = c(\tilde{\mathbf{x}}) \sum_{i=1}^{n} \frac{\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_i}{||\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_i||^{N+1}}$$

where the gradient field is calculated on n *augmented* datapoints $\{\tilde{\mathbf{x}}_i = (\mathbf{x}_i, 0)\}_{i=1}^n$, and $c(\tilde{\mathbf{x}}) = \frac{N-1}{nS_N(1)} \min_j ||\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_i||^{N+1}$ is the multiplier for numerical stability. We further normalize the field to resolve the variations in the magnitude of the norm $\| \hat{\mathbf{E}}(\tilde{\mathbf{x}}) \|_2$, and fit the neural network to the more amenable normalized field $\mathbf{v}(\tilde{\mathbf{x}}) = \hat{\mathbf{E}}(\tilde{\mathbf{x}})/\| \hat{\mathbf{E}}(\tilde{\mathbf{x}}) \|_2$. The Poisson field is rescalable (cf. Section 2) and thus trajectories of its forward/backward ODEs are invariant under normalization. We denote the empirical field calculated on batch data $\mathcal{B}$ by $\hat{\mathbf{E}}_{\mathcal{B}}$ and the normalized field as $\mathbf{v}_{\mathcal{B}}(\tilde{\mathbf{x}}) = \hat{\mathbf{E}}_{\mathcal{B}}(\tilde{\mathbf{x}})/\| \hat{\mathbf{E}}_{\mathcal{B}}(\tilde{\mathbf{x}}) \|_2$.

We sample training points on the trajectories of the forward ODE $d\tilde{\mathbf{x}}/dt = \mathbf{v}(\tilde{\mathbf{x}})$, so that the neural approximations are tailored to capture the normalized field well in areas likely traversed along backward ODE trajectories. Given a training point $\mathbf{x} \in \mathcal{D}$, we discretize the forward ODE by Euler method with the step sizes as a geometric sequence. Because the empirical Poisson fields are zero on $\{\tilde{\mathbf{x}}_i = (\mathbf{x}_i, 0)\}_{i=1}^n$, we perturb the augmented data $\tilde{\mathbf{u}}_0 = (\mathbf{x}, 0)$ by a small Gaussian noise to construct the initial value $\tilde{\mathbf{u}}_1$:

$$\tilde{\mathbf{u}}_1 = (\mathbf{x}, 0) + (\epsilon_{\mathbf{x}}, |\epsilon_z|), \quad \tilde{\mathbf{u}}_t = \tilde{\mathbf{u}}_{t-1} + \| \tilde{\mathbf{u}}_1 - \tilde{\mathbf{u}}_0 \|_2 \, \tau(1+\tau)^{t-1} \mathbf{v}(\tilde{\mathbf{u}}_{t-1}) \, (2 \leq t \leq M) \tag{6}$$

where $\epsilon = (\epsilon_{\mathbf{x}}, \epsilon_z) \sim \mathcal{N}(0, \sigma^2 I_{N+1 \times N+1})$. The terminal time M, standard deviation σ and τ are hyper-parameters. By the nature of the step size $\| \tilde{\mathbf{u}}_1 - \tilde{\mathbf{u}}_0 \|_2 \, \tau(1+\tau)^{t-1}$, one can show that the sequence of arc lengths in the discretization above, *i.e.*, $\{\sum_{i=1}^t \| \tilde{\mathbf{u}}_i - \tilde{\mathbf{u}}_{i-1} \|_2\}_{t=1}^M$, also form a geometric sequence (Appendix E.1). The rationale behind the design is that training points farther away from the data support play a less important role in generative modeling, sharing a similar spirit with the choice of noisy scales in score-based models [26, 27].

In practice, we simulate the forward ODE for a mini-batch data $\mathcal{B} = \{\mathbf{x}_i\}_{i=1}^{|\mathcal{B}|}$ in each iteration. We uniformly sample the terminal time M in $\{0, 1, \ldots, \text{MAX_STEP}\}$ for each datapoint. We select a large constant MAX_STEP as maximum number of steps in the forward ODE (Eq. (6)) (typically around 300) to ensure the terminal points can reach a large enough hemisphere. Since the empirical normalized field is biased, we use a larger batch $\mathcal{B}_L$ for the estimation, which empirically gives better results. Denoting the set of terminal points as $\{\tilde{\mathbf{y}}_i\}_{i=1}^{|\mathcal{B}|}$, we train the neural network f_θ on the terminal points to estimate the normalized field by minimizing the following loss:

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \| f_\theta(\tilde{\mathbf{y}}_i) - \mathbf{v}_{\mathcal{B}_L}(\tilde{\mathbf{y}}_i) \|_2^2$$

We summarize the training process in Algorithm 1. In practice, we add a small constant γ to the denominator of the normalized field to overcome the numerical issue when $\exists i, ||\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_i|| \approx 0$. The calculation of normalized fields in the forward ODE simulation (Algorithm 2) are based on a batch of samples, and can be easily parallelized on deep learning libraries like Pytorch [23].

Algorithm 1 Learning the normalized Poisson Field

Input: Training iteration T, Initial model f_θ, dataset $\mathcal{D}$, constant γ, learning rate η.
for $t = 1 \ldots T$ **do**
  Sample a large batch $\mathcal{B}_L$ from $\mathcal{D}$ and subsample a batch of datapoints $\mathcal{B} = \{\mathbf{x}_i\}_{i=1}^{|\mathcal{B}|}$ from $\mathcal{B}_L$
  Simulate the ODE: $\{\tilde{\mathbf{y}}_i = \text{forwardODE}(\mathbf{x}_i, \mathcal{B}_L)\}_{i=1}^{|\mathcal{B}|}$
  Calculate the normalized field by $\mathcal{B}_L$: $\mathbf{v}_{\mathcal{B}_L}(\tilde{\mathbf{y}}_i) = \hat{\mathbf{E}}_{\mathcal{B}_L}(\tilde{\mathbf{y}}_i)/(\| \hat{\mathbf{E}}_{\mathcal{B}_L}(\tilde{\mathbf{y}}_i) \|_2 + \gamma), \forall i$
  Calculate the loss: $\mathcal{L}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \| f_\theta(\tilde{\mathbf{y}}_i) - \mathbf{v}_{\mathcal{B}_L}(\tilde{\mathbf{y}}_i) \|_2^2$
  Update the model parameter: $\theta = \theta - \eta \nabla \mathcal{L}(\theta)$
end for
return f_θ

3.3 Backward ODE anchored by the additional dimension

After estimating the normalized field $\mathbf{v}$, we can sample from the data distribution by the backward ODE $d\tilde{\mathbf{x}} = -\mathbf{v}(\tilde{\mathbf{x}})dt$. Nevertheless, the boundary condition of the above ODE is unclear: the starting and terminal time t of the ODE are both unknown. To remedy the issue, we propose an equivalent

Algorithm 2 forwardODE($\mathbf{x}, \mathcal{B}$)

Sample # of forward step $M \sim \mathcal{U}\{0, 1, \ldots, \text{MAX_STEP}\}$
Initialize $\tilde{\mathbf{u}}_0 = (\mathbf{x}, 0)$, $\tilde{\mathbf{u}}_1 = (\mathbf{x}, 0) + (\epsilon_{\mathbf{x}}, |\epsilon_z|)$, $(\epsilon_{\mathbf{x}}, \epsilon_z) \sim \mathcal{N}(0, \sigma^2 I_{N+1\times N+1})$
for $t = 2 \ldots M$ **do**
  $\tilde{\mathbf{u}}_t = \tilde{\mathbf{u}}_{t-1} + \parallel \tilde{\mathbf{u}}_1 - \tilde{\mathbf{u}}_0 \parallel_2 \tau(1+\tau)^{t-1}\mathbf{v}_{\mathcal{B}}(\tilde{\mathbf{u}}_{t-1})$
end for
return $\tilde{\mathbf{u}}_M$

backward ODE in which $\mathbf{x}$ evolves with the augmented variable z:

$$d(\mathbf{x}, z) = -(\frac{d\mathbf{x}}{dt}\frac{dt}{dz}dz, dz) = -(\mathbf{v}(\tilde{\mathbf{x}})_{\mathbf{x}}\mathbf{v}(\tilde{\mathbf{x}})_z^{-1}, 1)dz$$

where $\mathbf{v}(\tilde{\mathbf{x}})_{\mathbf{x}}, \mathbf{v}(\tilde{\mathbf{x}})_z$ are the corresponding components of $\mathbf{x}, z$ in vector $\mathbf{v}(\tilde{\mathbf{x}})$. In the new ODE, we replace the time variable t with the physically meaningful variable z, permitting explicit starting and terminal conditions: when $z = 0$, we arrive at the data distribution and we can freely choose a large z_{max} as the starting point in the backward ODE. The backward ODE is compatible with general-purpose ODE solvers, *e.g.*, RK45 method [20] and forward Euler method. The popular black-box ODE solvers, such as the one in Scipy library [30], typically use a common starting time for the same batch of samples. Since the distribution on the $z = z_{\text{max}}$ hyperplane is no longer uniform, we derive the prior distribution by radially projecting uniform distribution on the hemisphere with radius $r = z_{\text{max}}$ to the $z = z_{\text{max}}$ hyperplane:

$$p_{\text{prior}}(\mathbf{x}) = \frac{z_{\text{max}}^{N+1}}{S_N(z_{\text{max}})(\parallel \boldsymbol{x} \parallel_2^2 + z_{\text{max}}^2)^{\frac{N+1}{2}}} = \frac{z_{\text{max}}}{S_N(1)(\parallel \boldsymbol{x} \parallel_2^2 + z_{\text{max}}^2)^{\frac{N+1}{2}}}$$

where $S_N(r)$ is the surface area of N-sphere with radius r. The reason behind the radical projection is that the Poisson field points in the radial direction at $r \to \infty$. The new backward ODE defines a bijection between $p_{\text{prior}}(\mathbf{x})$ on the infinite hyperplane ($z_{\text{max}} \to \infty$) and the data distribution $\tilde{p}(\tilde{\mathbf{x}})$, analogous to Theorem 1. In order to sample from $p_{\text{prior}}(\mathbf{x})$, it is suffice to sample the norm from the distribution: $p_{\text{prior}}(\parallel \boldsymbol{x} \parallel_2) \propto \parallel \boldsymbol{x} \parallel_2^{N-2}/(\parallel \boldsymbol{x} \parallel_2^2 + z_{\text{max}}^2)^{\frac{N}{2}}$ and then uniformly sample its angle. Detailed derivations are deferred to Appendix D. We further achieve exponentially decay on the z dimension by introducing a new variable t':

$$[\text{Backward ODE}] \quad d(\mathbf{x}, z) = -(\mathbf{v}(\tilde{\mathbf{x}})_{\mathbf{x}}\mathbf{v}(\tilde{\mathbf{x}})_z^{-1}z, z)dt' \tag{7}$$

The z component in the backward ODE, *i.e.*, $dz = -zdt'$, can be solved by $z = e^{-t'}$. Since z reaches zero as $t' \to \infty$, we instead choose a tiny positive number z_{min} as the terminal condition. The corresponding starting/terminal time of the variable t' are $-\log z_{\text{max}}/-\log z_{\text{min}}$ respectively. Empirically, this simple change of variable leads to $2\times$ faster sampling with almost no harm to the sample quality. We defer more details to Appendix E.2.

4 Generative Modeling via the Backward ODE

In this section, we demonstrate the effectiveness of the backward ODE associated with PFGM on image generation tasks. We show that PFGM has competitive or even better generation quality than current state-of-the-art methods under the same architecture, and achieving $10\times \sim 20\times$ acceleration. Meanwhile, unlike existing ODE baselines that heavily rely on corrector to generate decent samples, the backward ODE of PFGM exhibits greater stability against error. In addition, we show that the generation process of PFGM inherits the merits of ODE samplers, such as likelihood evaluation, latent representations and adaptive sampling.

4.1 Efficient and stable image generation by Poisson Flow

Setup We consider the CIFAR-10 [19] and CelebA [31] dataset for image generation task. Following [26], we first center-crop the CelebA images and then resize them to 64×64. We choose MAX_STEP $= 300$, $\sigma = 0.01$ and $\tau = 0.03$ for the forward ODE during training, and $z_{\text{min}} = 2e-3$, $z_{\text{max}} = 40$ (CIFAR-10)/60 (CelebA 64×64) for the backward ODE. Due to computational constraints, we use the NCSNv2 architecture in [26] as our backbone, instead of the SOTA architecture NCSN++ in [27]. The architecture of NCSN++ is larger and takes significantly longer training time.

Table 1: CIFAR-10 sample quality (FID, Inception) and number of function evaluation (NFE). All the methods below the *NCSNv2 backbone* separator use the NCSNv2 [26] network architecture as the backbone.

	Inception ↑	FID ↓	NFE ↓
PixelCNN [29]	4.60	65.93	1024
IGEBM [7]	6.02	40.58	60
WGAN-GP [11]	$7.86 \pm .07$	36.4	1
SNGAN [22]	$8.22 \pm .05$	21.7	1
NCSN [25]	$\mathbf{8.87 \pm .12}$	25.32	1001
NCSNv2 backbone			
Langevin dynamics [26]	$8.40 \pm .07$	**10.87**	1161
VE-SDE (PC) [27]	$8.23 \pm .02$	10.94	1000
VP-SDE (PC) [27]	$6.85 \pm .01$	44.05	1000
VE-ODE (Euler w/ corrector)	$8.05 \pm .03$	11.33	1000
VP-ODE (Euler w/ corrector)	$7.33 \pm .07$	37.74	1000
PFGM (Euler)	$8.00 \pm .09$	11.78	200
PFGM (RK45)	$8.30 \pm .05$	11.22	**118**

Note that PFGM can be readily paired with the NCSN++ architecture for a better performance. We add the scalar z (resp. predicted direction on z) as input (resp. output) to accommodate the additional dimension. We use the same set of hyper-parameters, such as batch size, learning rate and training iterations in [26]. We provide more training details in Appendix E.1.

Baselines We compare PFGM to modern autoregressive models [29], GANs [11, 22] and EBMs [7]. We also compare with variants of score-based models, from the annealed Langevin dynamics sampler [25, 26] to state-of-the-art SDE/ODE sampler [27]. For fair comparison, PFGM and score-based models use the NCSNv2 [26] architecture as the backbone. We denote the methods that use forward-time SDEs such as Variance Exploding (**VE**) SDE and Variance Preserving (**VP**) SDE [27] and the corresponding backward SDE/ODE, as **A-B**, where A $\in$ {VE, VP} and B $\in$ {SDE, ODE}.

Numerical Solvers The backward ODE (Eq. (7)) is compatible with any general purpose ODE solver. In our experiments, we choose the forward Euler method (**Euler**) with fixed number of steps, and the black box ODE solver in the Scipy library [30] with the RK45 [6] method (**RK45**). For VE-SDE and VP-SDE, we use the predictor-corrector (**PC**) sampler introduced in [27]. We explicitly name the sampler, with forward Euler method as predictor and Langevin dynamics as corrector, as **Euler w/ corrector**.

Results For quantitative evaluation on CIFAR-10 and CelebA, we report the Inception [24] (higher is better) and FID [12] scores (lower is better) in Table 1 and Table 2. We omit the Inception score on CelebA as it is not widely reported. We do not report the results of VE/VP-ODE without corrector, such as VE-ODE (RK45), since they have worse-off sample quality and get FIDs greater than 90 on CIFAR-10 and 60 on CelebA. We measure the inference speed by the average NFE (number of function evaluation).

Table 2: FID/NFE on CelebA 64×64

	FID ↓	NFE ↓
NCSN [25]	26.89	1001
NCSNv2 backbone		
Langevin dynamics [26]	10.23	2501
VE-SDE (PC) [27]	8.15	1000
VP-SDE (PC) [27]	34.52	1000
VE-ODE (Euler) w/ corrector	8.30	200
VP-ODE (Euler) w/ corrector	41.81	200
PFGM (Euler)	**7.85**	**100**
PFGM (RK45)	7.93	110

Our main findings are: **(1)** Table 1 shows that PFGM with RK45 method has competitive FID/Inception scores with the Langevin dynamics, which was the best model on the NCSNv2 architecture before, and requires $10\times$ less NFE. In addition, PFGM performs better than all the other ODE samplers. **(2)** As shown in Table 2, PFGM achieves best FID scores than all the baselines on CelebA dataset, while accelerating the inference speed around $20\times$. Remarkably, PFGM outperforms the Langevin dynamics and reverse-time SDE samplers, which are usually considered better than their deterministic counterparts. **(3)** Our method is more tolerant of sampling error. Among the compared ODEs, our backward ODE (Eq. (7)) is the only one that successfully generates high quality samples while the VE/VP-ODE fail w/o the Langevin dynamics corrector. The backward ODE still beats the

(a) CIFAR-10 (b) CelebA 64×64

Figure 3: PFGM samples on CIFAR-10 and CelebA 64×64

baselines w/ corrector. In Fig. 3, we visualize the uncurated samples from PFGM for CIFAR-10 and CelebA 64×64.

Failure of VE/VP ODEs Although [27] demonstrates that both VE/VP-ODE have high sample quality on NCSN++, we empirically observe that these backward ODEs perform poorly on the NCSNv2 backbone. For the score-based model, the l_2 norm of perturbed training data and the standard variance of Gaussian perturbation kernel $\sigma(t)$ have strong correlation, *e.g.*, l_2 norm $\approx \sigma(t)\sqrt{N}$ with large $\sigma(t)$ in VE [27]. We hypothesize that the VE/VP-ODE are more sensitive to estimation error because of such correlation. VE/VP-ODE can break down when the trajectories of backward ODEs deviate from the norm-$\sigma(t)$ relation. The smaller NCSNv2 backbone incurs larger errors and thus leads to their failure. Fig. 4 shows the ideal sample norm distributions with varying time variables of different ODEs. We observe that the PFGM has much weaker norm-$z(t')$ correlation while VE puts most mass around $\| x \|_2 = \sigma(t)\sqrt{N}$.

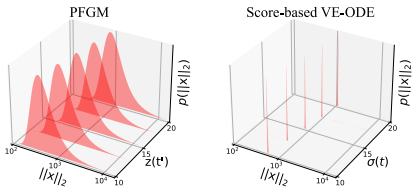
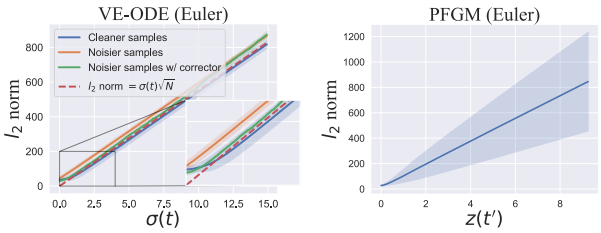


Figure 4: Sample norm distributions with varying time variables

In our experiments, we observe that many samples in VE-ODE have large noise. To verify the hypothesis above, we split a batch of VE-ODE samples into cleaner/noisier samples by their visual quality. In Fig. 5(a), we investigate the relation for cleaner/noisier samples during the simulation of VE-ODE (Euler) when $\sigma(t) < 15$. We can see that the trajectory of cleaner samples stays close to the norm-$\sigma(t)$ relation, while the noisier samples diverge from the relation. The Langevin dynamics corrector changes the trajectory of noisier samples to align with the relation. We also examine the norm-$z(t')$ relation during the backward ODE of PFGM. Fig. 5(b) shows that the anchored variable $z(t')$ and the norms in the backward ODE are not strongly correlated, giving rise to the robustness against the imprecise estimation on smaller backbone. We defer more details to Appendix F.



(a) VE-ODE (b) PFGM

Figure 5: **(a)** Norm-$\sigma(t)$ relation during the backward sampling of VE-ODE (Euler). **(b)** Norm-$z(t')$ relation during the backward sampling of PFGM (Euler). The shaded areas mean the standard deviation of norms.

Table 3: Bits/dim on CIFAR-10

	bits/dim ↓
Flow model	
RealNVP [5]	3.49
Glow [16]	3.35
Residual Flow [3]	3.28
FFJORD [9]	3.40
Flow++ [13]	3.29
DDPM backbone	
DDPM (L) [15]	$\leq 3.70^*$
DDPM VP-ODE [27]	3.21
DDPM sub-VP-ODE [27]	3.05
NCSNv2 backbone	
PFGM ODE	3.30

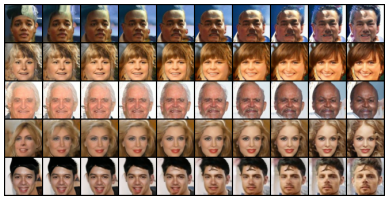
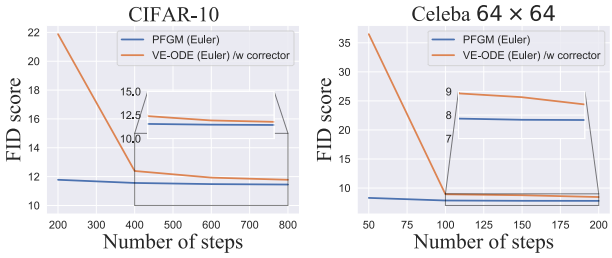


Figure 6: Interpolation on CelebA



(a) CIFAR-10 (b) CelebA 64×64

Figure 7: Number of steps versus FID score

4.2 Utilities of ODE: likelihood evaluation, latent representation and adapative sampling

Similar to the family of normalizing flows [5, 16, 13] and reverse-time ODE samplers [27], the forward ODE in PFGM defines an invertible mapping between the data space and latent space with a known prior. Formally, we define the invertible forward $\mathcal{M}$ mapping by integrating the corresponding forward ODE $d(\mathbf{x}, z) = (\mathbf{v}(\tilde{\mathbf{x}})_{\mathbf{x}}\mathbf{v}(\tilde{\mathbf{x}})_z^{-1}z, z)dt'$ of Eq. (7):

$$\mathbf{x}(\log z_{\text{max}}) = \mathcal{M}(\mathbf{x}(\log z_{\text{min}})) \equiv \mathbf{x}(\log z_{\text{min}}) + \int_{\log z_{\text{min}}}^{\log z_{\text{max}}} \mathbf{v}(\mathbf{x}(t'))_{\mathbf{x}}\mathbf{v}(\tilde{\mathbf{x}}(t'))_z^{-1}e^{t'}dt'$$

where $\log z_{\text{min}}$/$\log z_{\text{max}}$ are the starting/terminal time in the forward ODE. The forward mapping transfers the data distribution to the prior distribution p_{prior} on the $z = z_{\text{max}}$ hyperplane (cf. Section 3.3): $p_{\text{prior}}(\mathbf{x}(\log z_{\text{max}})) = \mathcal{M}(p(\mathbf{x}(\log z_{\text{min}})))$. The invertibility enables likelihood evaluation and creates a meaningful latent space on the $z = z_{\text{max}}$ hyperplane. In addition, we can adapt to the computational constraints by adjusting the precision in numerical ODE solvers.

Likelihood evaluation We evaluate the data likelihood by the instantaneous change-of-variable formula [4, 27]. In Table 3, we report the bits/dim on the uniformly dequantized CIFAR-10 test set. We compare with existing baselines that use the same setup. We observe that PFGM achieves a bits/dim on par with SOTA normalizing flow models, even without maximum likelihood training. We also compare with baselines using the backbone in DDPM [15], which is similar to NCSN++ and has much larger capacity than our NCSNv2 backbone. The PFGM still gets competitive likelihood with the VP-ODE with the DDPM backbone.

Latent representation We demonstrate the utility of the latent representation by conducting image interpolation. As shown in Fig. 6, we can travel along the latent space, *i.e.*, the $z = z_{\text{max}}$ hyperplane, to obtain perceptually consistent interpolations between CelebA images. We defer the temperature scaling on the latent space [5, 16, 27] to Appendix G.2.

Adaptive sampling We can trade off sample quality for computational efficiency by adjusting the relative and absolute tolerances `rtol`, `atol` in the RK45 method, or the number of steps in the forward Euler method. As shown in Fig. 7, the sample quality of PFGM degrades gracefully as we reduce the number of steps in the forward Euler method. In addition, our method shows better adaptivity than the VE-ODE w/ corrector, especially when only taking few Euler steps. In Appendix G.1 we show similar results by enlarging the error tolerance in the RK45 method.

5 Conclusion

We present a new deep generative model by solving the Poisson equation whose source term is the data distribution. We estimate the normalized gradient field of the solution in an augmented space with an additional dimension. For sampling, we devise a backward ODE that exponential decays on the physically meaningful additional dimension. Empirically, our approach has competitive or better performance than baselines on image generation, while achieving $10\times$ to $20\times$ acceleration. Our backward ODE shows greater stability against error than popular ODE-based methods. We further demonstrate the utility of the forward ODE on likelihood evaluation, image interpolation and adaptive sampling. Future directions include improving the normalization of Poisson fields and speeding up the training process. More principled approaches can be used to get around the divergent near-field behavior. For example, we may exploit renormalization, a useful tool in physics, to make the Poisson field well-behaved in near fields. Another direction is to explore faster sampling algorithms to avoid sequential simulation in the forward ODE.

References

[1] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *ArXiv*, abs/1809.11096, 2019.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020.

[3] Ricky T. Q. Chen, Jens Behrmann, David Kristjanson Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. *ArXiv*, abs/1906.02735, 2019.

[4] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Kristjanson Duvenaud. Neural ordinary differential equations. *ArXiv*, abs/1806.07366, 2018.

[5] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *ArXiv*, abs/1605.08803, 2017.

[6] J. R. Dormand and P. J. Prince. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6:19–26, 1980.

[7] Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. *ArXiv*, abs/1903.08689, 2019.

[8] Herbert Goldstein, Charles Poole, and John Safko. Classical mechanics, 2002.

[9] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Kristjanson Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *ArXiv*, abs/1810.01367, 2019.

[10] David J Griffiths. Introduction to electrodynamics, 2005.

[11] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. In *NIPS*, 2017.

[12] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017.

[13] Jonathan Ho, Xi Chen, A. Srinivas, Yan Duan, and P. Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. *ArXiv*, abs/1902.00275, 2019.

[14] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *NIPS*, 2016.

[15] Jonathan Ho, Ajay Jain, and P. Abbeel. Denoising diffusion probabilistic models. *ArXiv*, abs/2006.11239, 2020.

[16] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *NeurIPS*, 2018.

[17] Diederik P. Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. *ArXiv*, abs/1406.5298, 2014.

[18] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.

[19] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[20] F Shampine Lawrence. Some practical runge-kutta formulas. *Mathematics of Computation*, 46:135–150, 1986.

[21] Daiqing Li, Junlin Yang, Karsten Kreis, Antonio Torralba, and Sanja Fidler. Semantic segmentation with generative models: Semi-supervised learning and strong out-of-domain generalization. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8296–8307, 2021.

[22] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *ArXiv*, abs/1802.05957, 2018.

[23] Adam Paszke, S. Gross, Francisco Massa, A. Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Z. Lin, N. Gimelshein, L. Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[24] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *ArXiv*, abs/1606.03498, 2016.

[25] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *ArXiv*, abs/1907.05600, 2019.

[26] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *ArXiv*, abs/2006.09011, 2020.

[27] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *ArXiv*, abs/2011.13456, 2021.

[28] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *SSW*, 2016.

[29] Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In *NIPS*, 2016.

[30] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, Ilhan Polat, Yu Feng, Eric W. Moore, J. Vanderplas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Daniel Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, Aditya Alessandro Pietro Alex Andreas Andreas Anthony Ant Vijaykumar Bardelli Rothberg Hilboll Kloeckner Sco, Aditya Vijaykumar, Alessandro Pietro Bardelli, Alex Rothberg, Andreas Hilboll, Andre Kloeckner, Anthony M. Scopatz, Antony Lee, Ariel S. Rokem, C. Nathan Woods, Chad Fulton, Charles Masson, Christian Häggström, Clark Fitzgerald, David A. Nicholson, David R. Hagen, Dmitrii V. Pasechnik, Emanuele Olivetti, Eric Martin, Eric Wieser, Fabrice Silva, Felix Lenders, Florian Wilhelm, Gert Young, Gavin A. Price, Gert-Ludwig Ingold, Gregory E. Allen, Gregory R. Lee, Hervé Audren, Irvin Probst, Jorg P. Dietrich, Jacob Silterra, James T. Webber, Janko Slavi, Joel Nothman, Johannes Buchner, Johannes Kulick, Johannes L. Schönberger, José Vinícius de Miranda Cardoso, Joscha Reimer, Joseph E. Harrington, Juan Luis Cano Rodríguez, Juan Nunez-Iglesias, Justin Kuczynski, Kevin Lee Tritz, Martin Dr Thoma, Matt Newville, Matthias Kümmerer, Maximilian Bolingbroke, Michael Tartre, Mikhail Pak, Nathaniel J. Smith, Nikolai Nowaczyk, Nikolay Shebanov, Oleksandr Pavlyk, Per Andreas Brodtkorb, Perry Lee, Robert T. McGibbon, Roman Feldbauer, Sam Lewis, Sam Tygier, Scott Sievert, Sebastiano Vigna, Stefan Peterson, Surhud More, Tadeusz Pudlik, Taku Oshima, Thomas J. Pingel, Thomas P. Robitaille, Thomas Spura, Thouis Raymond Jones, Tim Cera, Tim Leslie, Tiziano Zito, Tom Krauss, U. Upadhyay, Yaroslav O. Halchenko, and Y. Vázquez-Baeza. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature Methods*, 17:261 – 272, 2020.

[31] Shuo Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. From facial parts responses to face detection: A deep learning approach. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 3676–3684, 2015.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See the Section 5 and Appendix I
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix J
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Theorem 1
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A and Appendix D
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix E.1
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Table 1 and Fig 5(a), 5(b)
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix E.1
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]