EAI Diffusion Reading Group

Poisson Field Generative Models

01/04/2023

Summary

1. The Poisson Equation
2. PFGM Derivation
3. Results
4. PFGM++

The Poisson Equation

Given some *source* density $\rho : \mathbb{R}^N \to \mathbb{R}$ (with *compact support* *)

$$\Delta\varphi = -\rho$$

- $\Delta = \sum_{\mu=1}^{N} \frac{\partial^2}{\partial x_\mu^2}$ Laplacian operator (again!)
- Poisson Field $E(x) = -\nabla\varphi(x)$ satisfies *Gauss' Law*

$$\nabla \cdot E = \rho$$

One of the most basic PDEs in physics (yields Newtonian gravity, Electrostatic theory, ...)

* ρ is zero outside of some bounded set

Analytical solution

Given by *Green's function*

$$\varphi(x) = \int G(x,y)\rho(y)dy$$

- $G(x,y) = \frac{1}{(N-2)S_{N-1}}\frac{1}{\|x-y\|^{N-2}}$ (Green's function: solution for a single point source)
- $$E(x) = -\int \nabla_x G(x,y)\rho(y)dy$$
- $\nabla_x G(x,y) = \frac{1}{S_{N-1}}\frac{x-y}{\|x-y\|^{N}}$

Gradient Flow

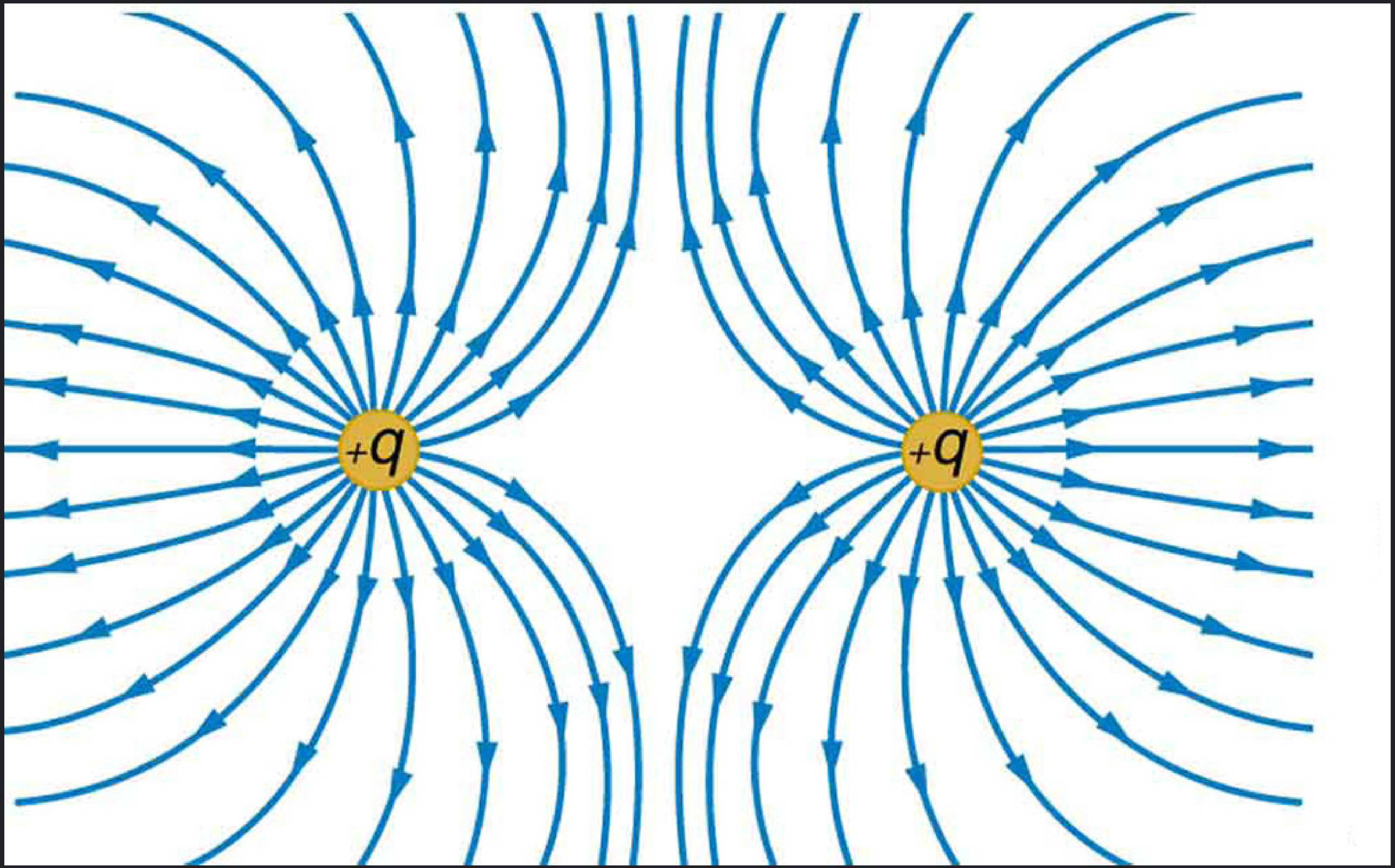
Consider a charged particle dropped in the field E, its trajectory is given by

$$\frac{dx}{dt} = E(x)$$

For a distribution of particles:

$$\frac{\partial p_t}{\partial t}(x) = -\nabla \cdot (p_t(x)E(x))$$

Field Effects



Finally putting the "consciousness" in large Neural Nets

Rescalable dynamics

The Gradient flow ODE is stiff :(. Luckily, for $f \in \mathcal{C}^1$, strictly positive

$$\frac{dx}{dt} = \pm f(x)E(x)$$

follows the same trajectories as the original gradient flow

Forward Process

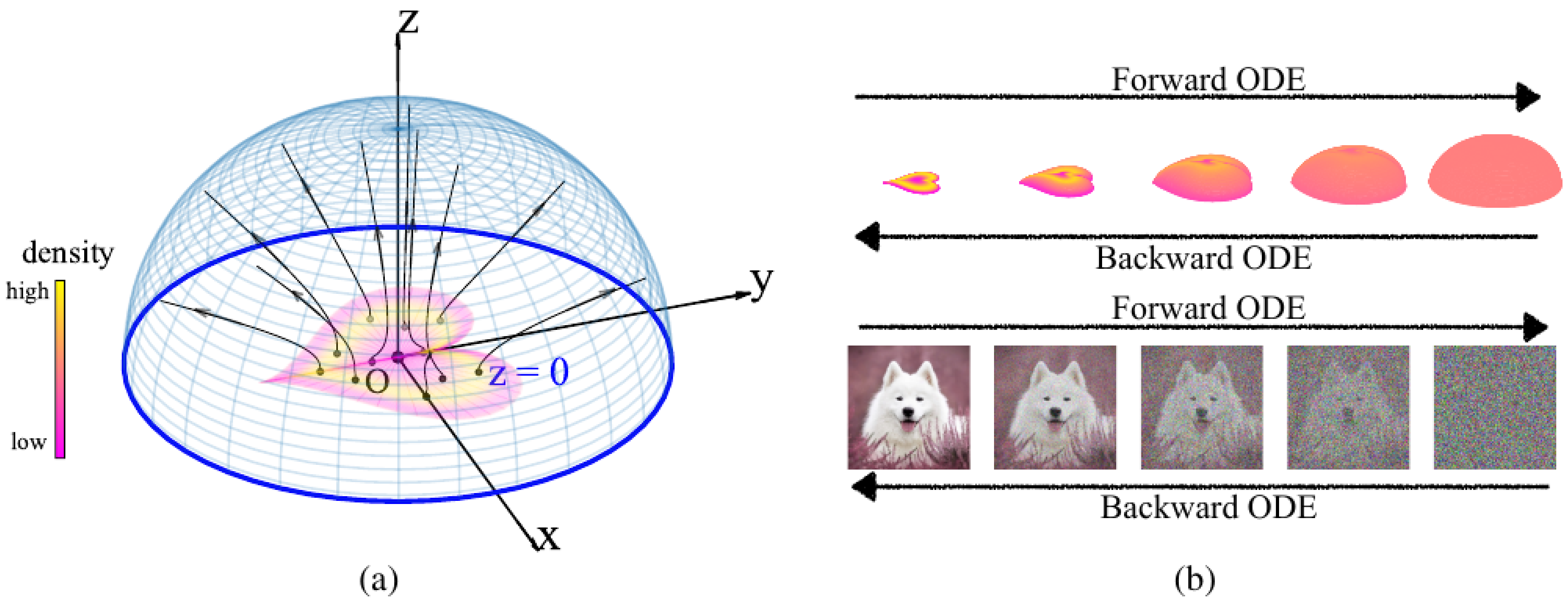


Figure 1: **(a)** 3D Poisson field trajectories for a heart-shaped distribution **(b)** The evolvements of a distribution (**top**) or an (augmented) sample (**bottom**) by the forward/backward ODEs pertained to the Poisson field.

From the Physics to the actual forward process

Several steps needed

1. Add extra dimension z (avoid mode collapse)
2. Normalize field E (numerical stability)
3. Replace time with z (for easier batching)

Add extra dimension

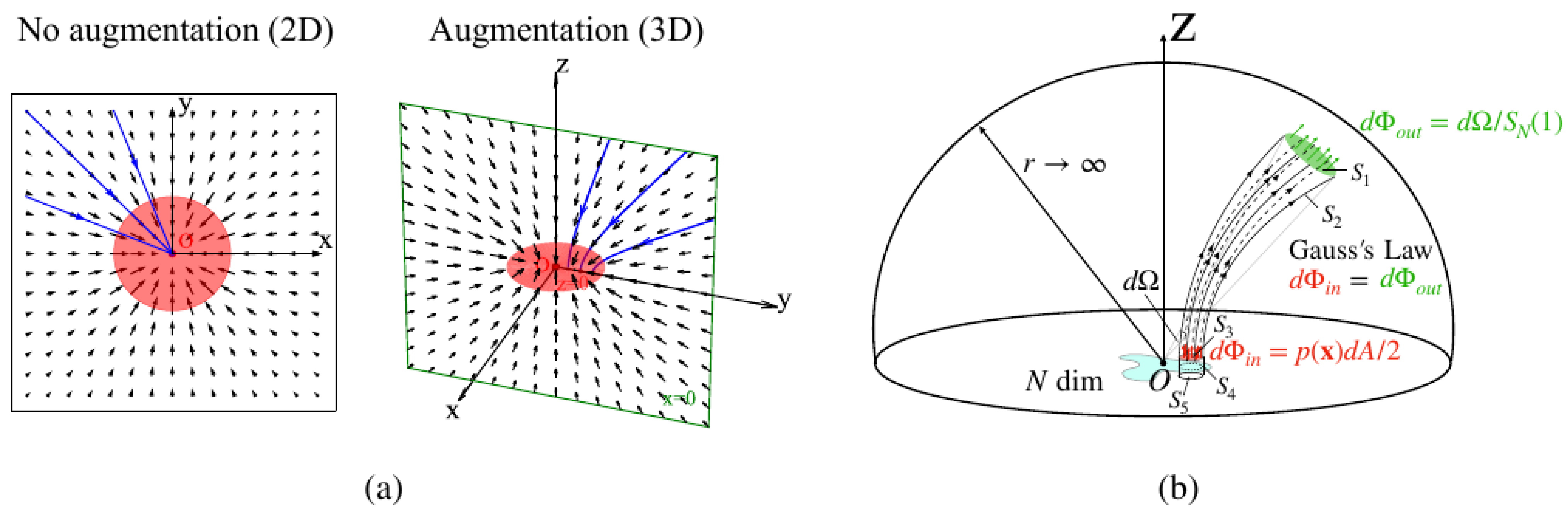


Figure 2: **(a)** Poisson field (black arrows) and particle trajectories (blue lines) of a 2D uniform disk (red). **Left** (no augmentation, 2D): all particles collapse to the disk center. **Right** (augmentation, 3D): particles hit different points on the disk. **(b)** Proof idea of Theorem 1. By Gauss's 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 $z > 0$.

Add extra dimension

- Instead of solving the Poisson equation $\Delta\varphi = -p$ in $\mathbb{R}^N$, solve in augmented space $\tilde{x} = (x, z) \in \mathbb{R}^{N+1}$
- Embed original data with $x \mapsto (x, 0)$

Asymptotics as $\|x\| \to \infty$

Since p has bounded support, as x follows $\frac{dx}{dt} = E(x)$, E becomes spherically symmetric

This means that for a sufficiently large radius r, the distribution p_t is effectively uniform on the hemi-sphere of radius r

Normalize E

Given a dataset $\mathcal{D} = \{x_i\}_{i=1}^n$, the empirical Poisson field is

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

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

Further normalization yields the *negative normalized field*

$$v(\tilde{x}) = -\frac{\sqrt{N}\hat{E}(\tilde{x})}{\|\hat{E}(\tilde{x})\| + \gamma}$$

Notation: fields calculated from batch $\mathcal{B}$: $\hat{E}_{\mathcal{B}}$, $v_{\mathcal{B}}$

NB: γ: small constant to avoid division by zero

Perturbing the augmented training data

Given training datapoint $x \in \mathbb{R}^N$, add noise to $\tilde{x} = (x, 0)$ to obtain (y, z) :

$$y = x + \|\epsilon_x\|(1+\tau)^m u, \quad z = |\epsilon_z|(1+\tau)^m$$

- $\epsilon = (\epsilon_x, \epsilon_z) \sim \mathcal{N}(0, \sigma^2 I)$
- $u \sim \mathcal{U}(S_N)$
- $m \sim \mathcal{U}[0, M]$

Hyperparameters: τ, σ, M

Loss

Given mini-batch data $\mathcal{B} = \{x_i\}_{i=1}^{|\mathcal{B}|}$ and a larger batch $\mathcal{B}_L$ (for estimating the normalized field), we train a network f_θ to minimize

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

Training Algorithms

Algorithm 1 Learning the normalized Poisson Field

Input: Training iteration T, Initial model f_θ, dataset $\mathcal{D}$, constant γ, learning rate η.
for $t = 1 \dots T$ **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: $\{\tilde{\mathbf{y}}_i = \text{perturb}(\mathbf{x}_i)\}_{i=1}^{|\mathcal{B}|}$
  Calculate the normalized field by $\mathcal{B}_L$: $\mathbf{v}_{\mathcal{B}_L}(\tilde{\mathbf{y}}_i) = -\sqrt{N}\hat{\mathbf{E}}_{\mathcal{B}_L}(\tilde{\mathbf{y}}_i)/(\| \hat{\mathbf{E}}_{\mathcal{B}_L}(\tilde{\mathbf{y}}_i) \|_2 + \gamma), \forall i$
  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

Algorithm 2 perturb($\mathbf{x}$)

Sample the power $m \sim \mathcal{U}[0, M]$
Sample the initial noise $(\epsilon_{\mathbf{x}}, \epsilon_z) \sim \mathcal{N}(0, \sigma^2 I_{(N+1)\times(N+1)})$
Uniformly sample the vector from the unit ball $\mathbf{u} \sim \mathcal{U}(S_N(1))$
Construct training point $\mathbf{y} = \mathbf{x} + \| \epsilon_{\mathbf{x}} \| (1+\tau)^m \mathbf{u}$, $z = |\epsilon_z|(1+\tau)^m$
return $\tilde{\mathbf{y}} = (\mathbf{y}, z)$

Backward ODE

Instead of solving backward ODE in time, solve the equivalent ODE in z:

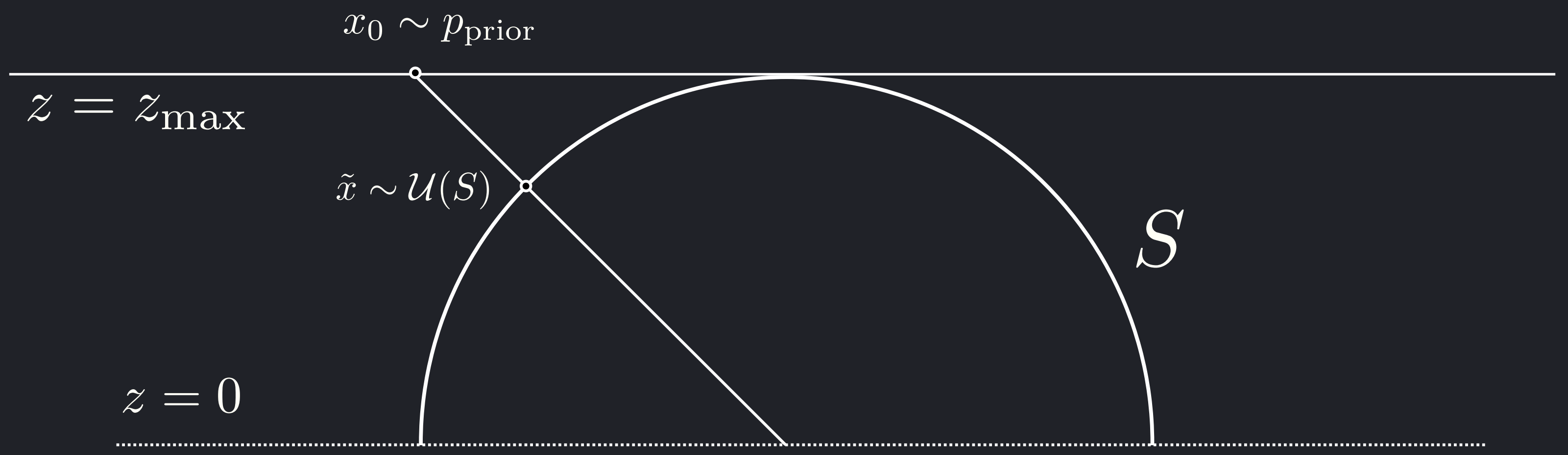
$$\frac{d(x,z)}{dz} = (\frac{dx}{dt}\frac{dt}{dz}, 1) = (v(\tilde{x})_x v(\tilde{x})_z^{-1}, 1)$$

Start from some $z \leq z_{\max}$, end at $z = 0$

Map hemisphere to $z = z_{\max}$ hyperplane

To allow processing batches, map the prior on the hemisphere $r = z_{\max}$ to the hyperplane

$$p_{\text{prior}}(x) = \frac{2z_{\max}}{S_N(1)(\|x\|_2^2 + z_{\max}^2)^{\frac{N+1}{2}}}$$



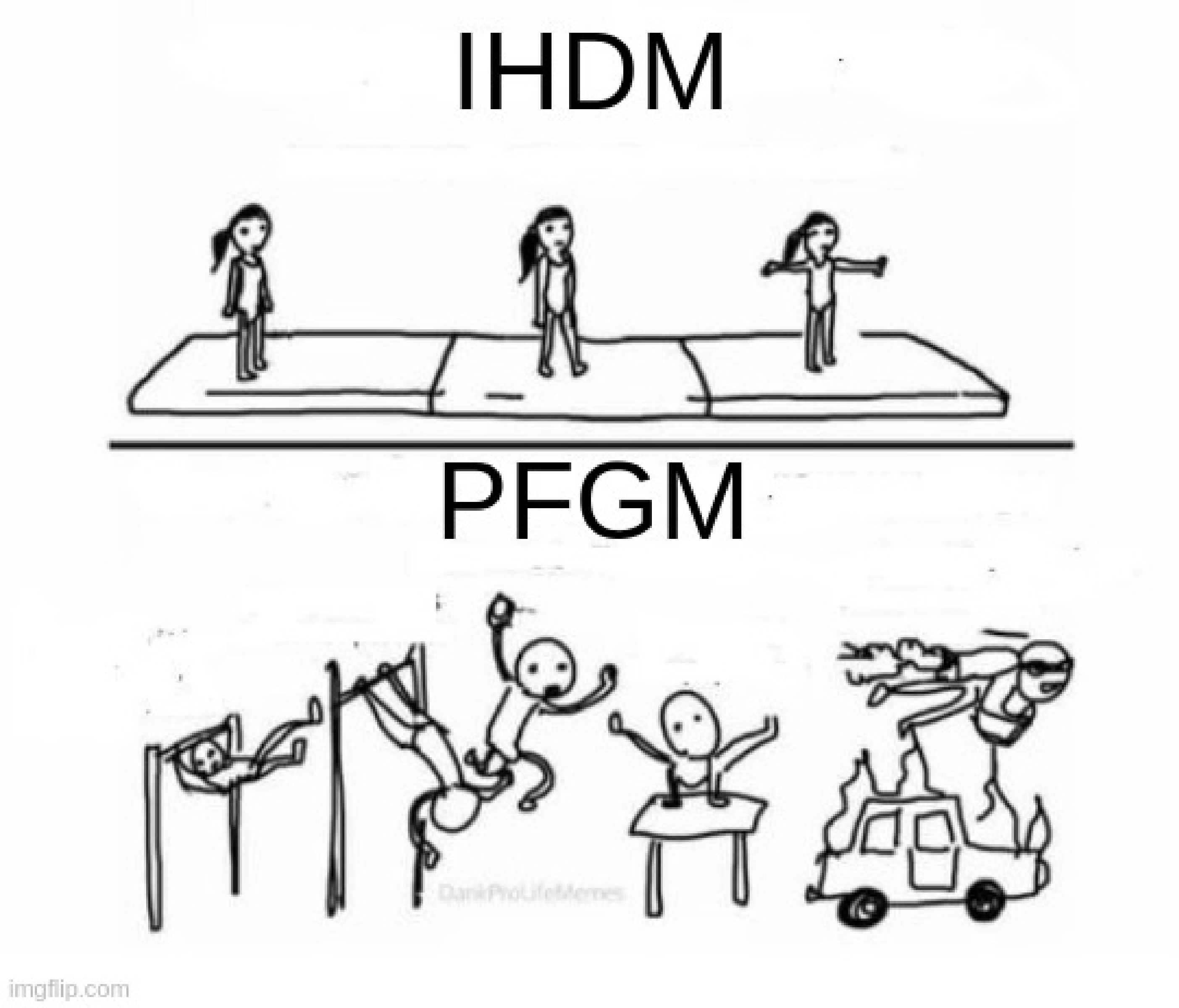
In practice: sample on hemisphere and project.

The actual Backward ODE for real

Add new time variable such that $dz = zdt'$

$$\frac{d(x,z)}{dt'} = (v(\tilde{x})_x v(\tilde{x})_z^{-1} z, z)$$

z now converges exponentially towards 0 in the backward ODE (actually makes solving the ODE faster)


IHDM
PFGM
imgflip.com

Results

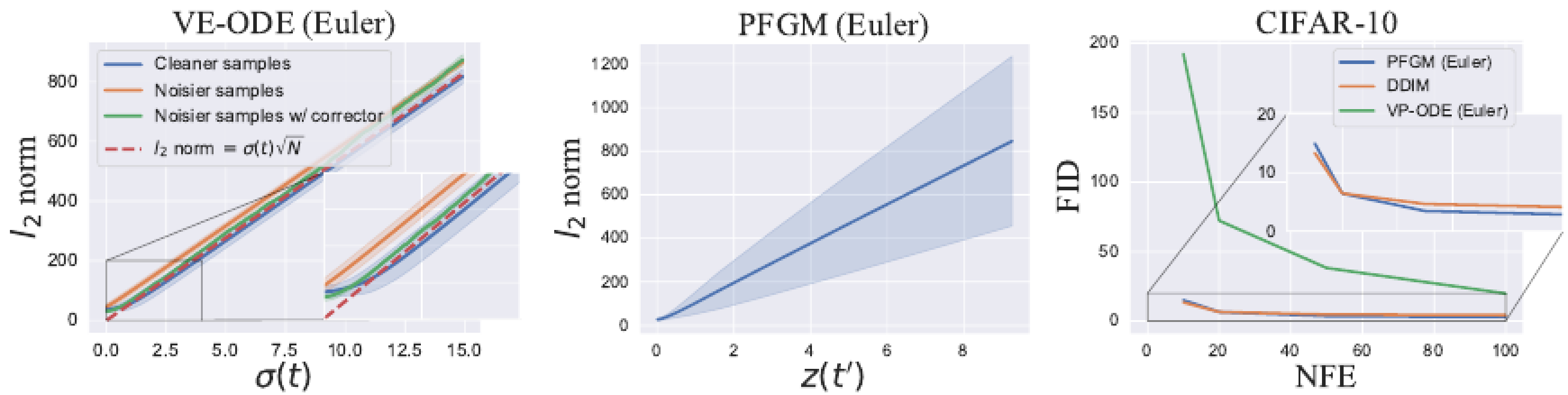
Table 1: CIFAR-10 sample quality (FID, Inception) and number of function evaluation (NFE).

	Invertible?	Inception ↑	FID ↓	NFE ↓
PixelCNN [36]	✗	4.60	65.9	1024
IGEBM [8]	✗	6.02	40.6	60
ViTGAN [24]	✗	9.30	6.66	1
StyleGAN2-ADA [17]	✗	9.83	2.92	1
StyleGAN2-ADA (cond.) [17]	✗	10.14	2.42	1
NCSN [31]	✗	8.87	25.32	1001
NCSNv2 [32]	✗	8.40	10.87	1161
DDPM [16]	✗	9.46	3.17	1000
NCSN++ VE-SDE [33]	✗	9.83	2.38	2000
NCSN++ deep VE-SDE [33]	✗	9.89	2.20	2000
Glow [19]	✓	3.92	48.9	1
DDIM, T=50 [30]	✓	-	4.67	50
DDIM, T=100 [30]	✓	-	4.16	100
NCSN++ VE-ODE [33]	✓	9.34	5.29	194
NCSN++ deep VE-ODE [33]	✓	9.17	7.66	194
DDPM++ backbone				
VP-SDE [33]	✗	9.58	2.55	1000
sub-VP-SDE [33]	✗	9.56	2.61	1000
VP-ODE [33]	✓	9.46	2.97	134
sub-VP-ODE [33]	✓	9.30	3.16	146
PFGM (ours)	✓	**9.65**	**2.48**	**104**
DDPM++ deep backbone				
VP-SDE [33]	✗	9.68	2.41	1000
sub-VP-SDE [33]	✗	9.57	2.41	1000
VP-ODE [33]	✓	9.47	2.86	134
sub-VP-ODE [33]	✓	9.40	3.05	146
PFGM (ours)	✓	**9.68**	**2.35**	**110**

Results Summary

- Achieves best Inception and FID scores among normalizing flow models
- (10x ~ 20x) Faster than SDE models with similar architectures

Euler method with low step size



(a) Norm-$\sigma(t)$ in VE-ODE (b) Norm-$z(t')$ in PFGM (c) FID vs. NFE on CIFAR-10

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. **(c)** Number of steps versus FID score.

Miscelaneous

- Likelihood evaluation
- Latent representation

Limitations

- The mini-batch normalized field estimator is biased
- need large training batch to compensate

These are fixed in PFGM++

PFGM++

- Generalize to $z \in \mathbb{R}^D$
- equivalent to Diffusion models for $D \to \infty$
- Dispenses with the biased normalized field estimator
- No need to actually solve with the D additional dimensions. One is enough
- Transfer Hyperparameters from Diffusion Models to arbitrary D