PGFM: Poisson Flow Generative Models, Anaylsis and Improvements

Khush Jain
Mathematics
IIT Bombay
200010040@iitb.ac.in

Kushal S M
Aerospace Eng
IIT Bombay
200010078@iitb.ac.in

Mohd Taha Abbas
Aerospace Eng
IIT Bombay
200010045@iitb.ac.in

Ninad Chaphekar
Aerospace Eng
IIT Bombay
200010016@iitb.ac.in

***Abstract*—“Poisson flow” generative model (PFGM) that maps a uniform distribution on a high-dimensional hemisphere into any data distribution. 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). The authors 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 $\rightarrow \infty$ limit. To learn the bijective transformation, they estimate the normalized field in the augmented space. For sampling, they 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 authors claim, PFGM achieves current state-of-the-art performance among the normalizing flow models on CIFAR-10, with an Inception score of 9.68 and a FID score of 2.35. It also performs on par with the state-of-the-art SDE approaches while offering 10× to 20× acceleration on image generation tasks. We are gonna test their claims, train the model on medical datasets and see the generation.

I. Introduction

Several families of generative models have evolved throughout the development of AI. Some approaches, like Normalizing Flows, offer explicit estimates of sample likelihoods. Other approaches, like GANs, cannot explicitly calculate likelihoods, but can generate very high-quality samples.

Over the past couple of years, significant research efforts have been undertaken to develop Diffusion Models. Diffusion Models draw inspiration from physics, in particular thermodynamics/statistical physics, noting that e.g. any localized distribution of a gas will eventually spread out to fill an entire space evenly simply through random motion.

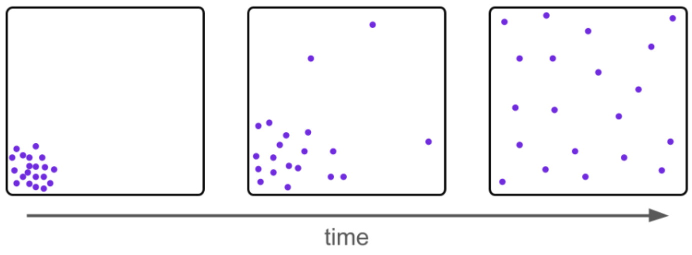


Fig. 1: A localized gas (purple particles) will spread out to evenly fill a room over time simply through random motion

A localized gas (purple particles) will spread out to evenly fill a room over time simply through random motion That is, any starting distribution is transformed into a uniform distribution over time. Diffusion Models draw inspiration from this idea and place e.g. images in a ”high dimensional room”, corrupting them with random noise in a similar way.

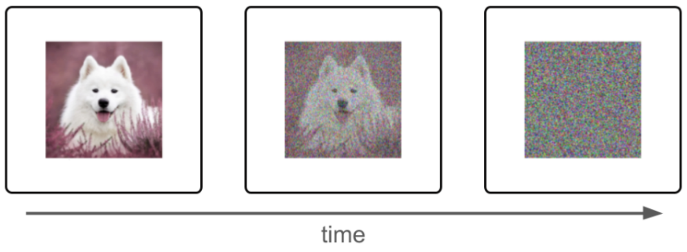


Fig. 2: The pixels in an image can be viewed as a localized cloud of particles that we ”diffuse” into random noise over time

The pixels in an image can be viewed as a localized cloud of particles that we ”diffuse” into random noise over time The goal is to train a Machine Learning model that learns how to go backwards through time, reversing this corruption process. If we can successfully learn such a mapping, then we have a transformation from a simple distribution (Gaussian, in the case of Diffusion Models) to the data distribution. We can then generate data simply by sampling from a Gaussian distribution and applying the learned transformation.

Poisson Flow Generative Models are inspired in a very similar manner to Diffusion Models, but they pull from the field of electrostatics rather than thermodynamics. The fundamental and foundational finding is that any distribution of electrons in a hyperplane generates an electric field that transforms the distribution into a uniform angular distribution as the distribution evolves through time according to the dynamics defined by the field. Treating a data distribution as a charge distribution defines an electric field that transforms the distribution into a uniform hemisphere over time If we know the electric field (aka Poisson field) generated by a distribution, then we can start with points uniformly sampled on a hemisphere and run the dynamics in reverse time to recover the original data distribution. The laws of physics, therefore, provide an invertible mapping between a simple

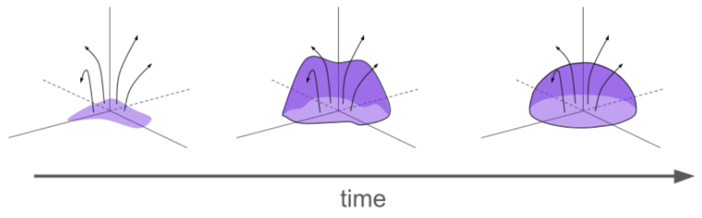


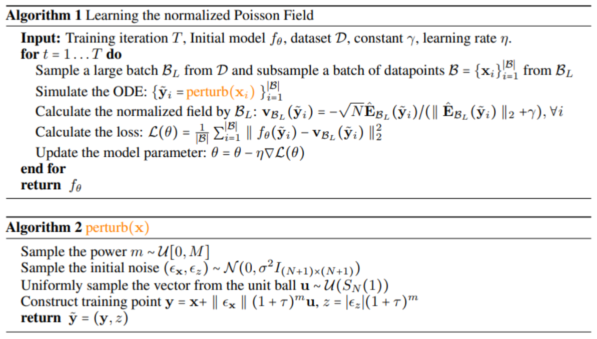
Fig. 3: Treating a data distribution as a charge distribution defines an electric field that transforms the distribution into a uniform hemisphere over time

distribution and the data distribution, yielding a means to generate novel data akin to normalizing flows

Instead of solving the Poisson equation in the original data space, it is solved in an augmented space x˜ = (x, z) with an additional variable z in the same subspace as x. We augment the training data x˜ in the new space by setting z = 0 such that x˜ = (x, 0). As a consequence, the data distribution in the augmented space is p˜(x˜) = p(x)delta(z), where delta is the Dirac delta function

II. Learning the normalized Poisson Field

Given a set of training data $D = x_{i\,i=1}^{\,n}$ i.i.d sampled from the data distribution p(x), gradient field is calculated on n augmented data points with numerical stability. For training, Fit the neural network to the more amenable negative normalized field calculated on batch data B by E^B as $v_B(x) = -\sqrt{N}\frac{E_B(x)}{||E_B(x)||_2}$. Denoting the set of perturbed points as $y_{i\,i=1}^{\,|B|}$, the neural network fis trained on these points to estimate the negative normalized field by minimizing the following loss: $L(\theta) = \frac{1}{|B|}\sum_{i=1}^{|B|}||f_\theta(y_i) - v_{BL}(y_i)||_2^2$



```
Algorithm 1 Learning the normalized Poisson Field
  Input: Training iteration $T$, Initial model $f_\theta$, dataset $\mathcal{D}$, constant $\gamma$, learning rate $\eta$.
  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 $f_\theta$

Algorithm 2 perturb(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 $\tilde{\mathbf{y}} = (\mathbf{y}, z)$
```

Fig. 4: The main algorithm to learn the Poisson Field

III. Architecture

A. Layer Composition

Each layer in a PFGM is composed of two fundamental components:

1. **Learnable Transformation**. The learnable transformation is a neural network that is optimized during training to map the input data to a new representation.

2. **Poisson Process**: The Poisson process component is responsible for modeling the dependency structure of the data. It is used to scale and shift the input data along specific directions.

B. Layer Forward Pass

During the forward pass, each layer applies the following operations:

1. **Learnable Transformation**: The learnable transformation is applied to the input data, resulting in a transformed representation.

2. **Poisson Process**: The Poisson process is applied to the transformed representation, resulting in a scaled and shifted representation along specific directions.

The output of the layer is the concatenation of these two components, resulting in a transformed and scaled representation.

C. Layer Backward Pass

During the backward pass, each layer applies the following operations:

1. **Learnable Inverse Transformation**: The inverse of the learnable transformation is applied to the output of the previous layer, resulting in an inverse transformed representation.

2. **Poisson Process Inverse**: The inverse of the Poisson process is applied to the inverse transformed representation, resulting in an unscaled and unshifted representation along specific directions.

IV. Advantages

A. *10x sampling*

- By recursively applying the Poisson process inverse, PFGM reduces the number of inverse transformations needed to compute the original input data by a factor of L.
- This is because the Poisson process inverse can be computed efficiently using the recursive formula, eliminating the need to compute the inverse transformation layer by layer.

B. *Recursive Formulation*

Assuming L layers, the recursive formulation of the inverse transformation can be written as:

- $I(z_L) = z_0$ (original input data)
- $I(z_{L-1}) = z_L \cdot P^{-1}$ (Poisson process inverse)
- $I(z_{L-2}) = I(z_{L-1}) \cdot T^{-1}(z_{L-2})$
- $\dots$
- $I(z_0) = I(z_1) \cdot T^{-1}(z_0)$

where T^{-1} denotes the inverse of the learnable transformation, and P^{-1} denotes the inverse of the Poisson process.

C. *Practical Implementation*

In practice, the recursive computation of the inverse transformation can be implemented efficiently using a stack-based approach. The stack is used to store the intermediate results of the inverse transformation, allowing for efficient computation of the inverse transformation.

V. GOAL

- Comparison of physics based diffusion model with other diffusion models on different datasets.
- Advantages/Disadvantages of using each model
- Differences in architectures, sampling techniques and other technical differences
- DIfference in generation quality, speed, memory efficiency and ease of use

VI. Approach

- Dataset Preparation: Obtain bloodMNIST, DermaMNIST, and RetinaMNIST datasets, preprocess for uniformity, and split into train/validation/test sets.
- Diffusion Model Architecture: PFGM++
- Training: Train the diffusion model, monitor with validation data, and apply techniques like data augmentation and regularization.
- Generation: Use the trained model to generate samples from each dataset.
- Evaluation: Compute Fréchet Inception Distance (FID) for generated samples to measure similarity to original datasets.

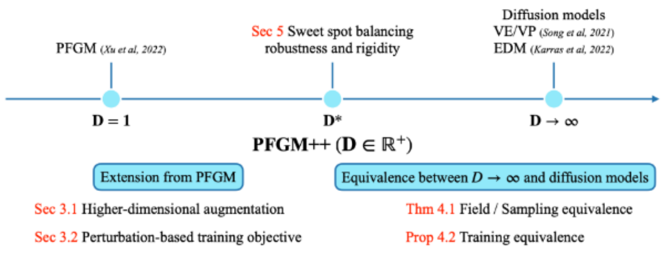
VII. Datasets

- BloodMNIST (samples 17092 classes: 8) : Images of Red Blood Cells

Scarcity of data in medical domain. Extremely complex modalities of data with skewed combinations and long-tailed class distributions. Thus, a good generative model can help overcome some of the difficulties

A. *Implementation Details*

- BloodMNIST - Images of size 28x28, 64z64, 128x128, 224x224
- Model Details - pfgmpp, D =128, batch size = 512, ticks = 150.


PFGM (Xu et al, 2022)
Sec 5 Sweet spot balancing robustness and rigidity
Diffusion models
VE/VP (Song et al, 2021)
EDM (Karras et al, 2022)
D = 1
D*
D → ∞
PFGM++ (D ∈ ℝ+)
Extension from PFGM
Equivalence between D → ∞ and diffusion models
Sec 3.1 Higher-dimensional augmentation
Sec 3.2 Perturbation-based training objective
Thm 4.1 Field / Sampling equivalence
Prop 4.2 Training equivalence

B. *Preliminary Results*

Succesfully Trained the Model on the medical MNIST dataset and here are some results.

Generated Images: Generated Images:

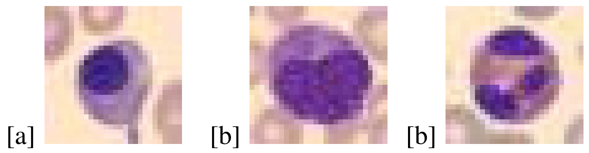


Fig. 5: Generated Images using PFGM

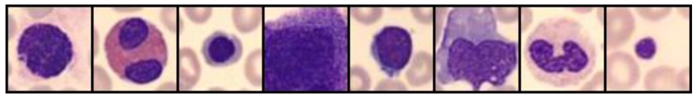


Fig. 6: Generated Images using DDPM

Training Data Images:

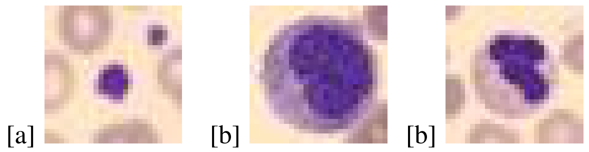


Fig. 7: Images from Training Data

C. *FID score comparision*

	PGFM	DDPM
28x28	2.14366	2.22919
64x64	5.02781	5.09118
128x128	6.50491	6.71045

TABLE I: FID scores

VIII. Work Done

- PFGM++ successfully trained on BloodMNIST: Resolved dependency issues, calculated FID score
- Train alternative diffusion models:
  - DDPM
  - Consistent setups for fair comparison.
- Compare and analyze FID scores:
  - Identify performance disparities.
  - Investigate contributing factors.

IX. Conclusion

In conclusion, while both Denoising Diffusion Probabilistic Models (DDPM) and Poisson Flow Generative Models (PFGM) offer promising approaches to high-quality sample generation, PFGM demonstrates superiority over DDPM in several key aspects. Firstly, PFGM excels in terms of sampling speed, achieving faster speeds compared to SDE samplers while maintaining high performance. This advantage is particularly pronounced in weaker architectures, where PFGM's backward ODE shows better generation performance and stability. Secondly, PFGM exhibits robustness and error tolerance due to weaker correlations between variables, making it a promising method for high-quality sample generation. Its scalability to higher resolution datasets, such as LSUN bedroom 256x256, with comparable performance to VE-SDE using fewer NFEs, further highlights its efficiency and effectiveness. Moreover, PFGM offers ease of use and implementation, with its backward ODE being compatible with architectures of varying capacities, offering stability against errors and robustness to step size variations. Overall, PFGM's combination of faster sampling speeds, better generation performance and stability, robustness, scalability, and ease of use position it as a superior choice over DDPM for high-quality sample generation tasks

References

[1] Poisson Flow Generative Models: https://arxiv.org/pdf/2209.11178 (2022)

[2] Denoising Diffusion Probabilistic Models: https://arxiv.org/pdf/2006.11239 (2020)