Diffusion Models and PFGM++

Emmanouil Pantelis
Artificial Intelligence and Learning Systems Laboratory
Electrical and Computer Engineering School, NTUA
Athens, Greece
pantelios2001@gmail.com

***Abstract*—In this paper, we describe the diffusion models, highlighting the key mathematical ideas behind them and the most important models presented so far. We focus on the Poisson Flow Generative Model, on which we also perform some experiments, such as image generation on pretrained models and training on the MNIST dataset.**

I. Introduction

A. Generative Models

Let's assume we have a data distribution $p(x)$ where data points x are associated with labels y based on their classes. Hence, our system is modeled by the distribution $p(x, y)$. In a basic classification problem, for a specific sample x, we seek the probabilities $p(y|x)$ in order to classify it into the most probable class. Conversely, the goal of a generative model is to sample from the distribution $p(x)$ or $p(x|y)$ if sampling from a specific class is desired. Examples of such models include variational autoencoders, generative adversarial networks, flow and energy based models and diffusion models, which we will study more extensively.

B. Diffusion Models

Diffusion models constitute a relatively modern approach on generative modeling, and along with GANs, achieve the greatest performance to date in image synthesis and similar data generation tasks. Initially, we will present the fundamental principles behind these models, and then, we will examine their historical evolution by presenting the most important milestones in their development. We will consider the general problem of sampling from a single distribution $p(x)$, while also briefly mentioning techniques on sampling from a specific class.//

Diffusion models can be approached from three different but interconnected perspectives:

- As a process of noising and denoising the data. We consider the model as a Markov Chain which, initiating from the initial distribution of our data, introduces a small perturbation at each step. Thus, after a sufficiently many steps, we reach a distribution that can be regarded as a typical Gaussian (white noise). Subsequently, we try to model the reverse process, which aims to reconstruct the initial data distribution, starting from white noise and removing a small amount of noise at each step. More details will be provided later.
- As Score Based Generative Models. Under this perspective, we want to compute the Fisher Score of the distribution $\nabla \log p(x)$ using statistical methods. Once calculated, we can them sample from the distribution $p(x)$ without computing it analytically using Langevin Dynamics, a method similar to MCMC that is derived from physics. Starting from a random point in space, we move along $\nabla \log p(x)$ towards areas of higher probability density. Due to some challenges in the Fisher Score calculation and in the Langevin Dynamics convergence which we will see later, we resort to implementing the above method using noisy versions of our data, for various noise levels, in a process that resembles the initial noising and denoising approach.
- As trajectories of Stochastic Differential Equations. This final approach models the noising process as the trajectory of an SDE, which evolves in time and gradually adds noise to the data. This SDE can be then time-reversed, in a reverse SDE that depends on the Fisher Score of the distribution. After calculating the score function, we can use various techniques, such as numerical solvers for the SDE or the corresponding ODE, or Annealed Langevin Dynamics to sample from the initial data. The previous two approaches can be shown to be special cases of this more general approach.

II. Diffusion Models so far

A. Inception of Diffusion Models

Diffusion models were introduced by Sohl-Dickstein et al in [1]. The idea of the gradual destruction of data structure by adding noise originated from the concept of diffusion in thermodynamics. In the forward trajectory, a Markov chain develops in T steps. The initial distribution is $q(x_0)$, and the transition probabilities are given by the following relation:

$$q((x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

Then, the assumption is made that for small values of β_t, $q(x_{t-1}|x_t)$ will be Gaussian, with mean and variance that can be parameterized and computed. Thus, we consider the inverse process:

$$p(x_t) = \mathcal{N}(x_t; 0, I)$$

$$p(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; f_\mu(x_t, t), f_\Sigma(x_t, t))$$

Now we can calculate the probability of obtaining a sample:

$$p(x_0) = \int p(x_{0\ldots T})\, dx_{1\ldots T}$$
$$= \int q(x_{1\ldots T}|x_0)p(x_T) \prod \frac{p(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\, dx_{1\ldots T}$$

Our goal in training is to determine the means and variances of each step to maximize the log likelihood of the model:

$$L = \int \log(p(x_0)q(x_0))dx_0$$

Instead of this, optimization is chosen with respect to a lower bound of L. Regarding the diffusion coefficients β_t, they are chosen to be learned from the model as parameters.

Finally, if we want to use some a priori knowledge for the data we want to generate, we sample from the distribution:

$$\mathcal{P}(x_0) = \frac{1}{Z} p(x_0)r(x_0)$$

It is shown that the process we follow remains the same if we choose:

$$\mathcal{P}(x_t|x_{t+1}) = \frac{1}{Z_t(x_t)} p(x_t|x_{t+1})r(x_t)$$

The idea is that if the function $r(x_t)$ is smooth enough, as $p(x_t|x_{t+1})$ is Gaussian with very small variance, the product will remain almost Gaussian, with a different mean indicating a priori knowledge.

B. Denoising Diffusion Probabilistic Models

The Denoising Diffusion Probabilistic Model (DDPM) is a very powerful model that achieves the generation of samples of impressive quality, that was introduced by Ho et al in [2]. The model has the same form as before, but the training method changes. It is proven that:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{a_t}x_0, (1-a_t)I),$$

where $\alpha_t = 1 - \beta_t$ and $a_t = \prod_{s=1}^{t} \alpha_s$.

As before, we aim to optimize the log likelihood of the model. We will use the Expectation Lower Bound (ELBO), as in the training of Variational Autoencoders.

$$\mathbb{E}_q\left[-\log p_\theta(x_0)\right] \leq \mathbb{E}_q\left[-\log p(x_T) - \sum \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right]$$
$$= \mathbb{E}_q[D_{KL}(q(x_T|x_0)\|p(x_T))$$
$$+ \sum D_{KL}(q(x_{t-1}|x_t, x_0)\|p_\theta(x_{t-1}|x_t))$$
$$- \log p_\theta(x_0|x_1)]$$

Therefore, we minimize with respect to 3 terms.

The term $L_T = \mathbb{E}_q\left[D_{KL}(q(x_T|x_0)\|p(x_T))\right]$ expresses the distance of the distribution of our noisy data from the white noise where we wanted to end up. Since we consider the parameters β_t as being constant, this term does not affect training.

The term $L_0 = -\mathbb{E}_q\left[\log p_\theta(x_0|x_1)\right]$ is a reconstruction term, which can be trained separately.

The term $L_{t-1} = \mathbb{E}_q\left[D_{KL}(q(x_{t-1}|x_t, x_0)\|p_\theta(x_{t-1}|x_t))\right]$ expresses the distance between the distribution p_θ we use for reconstruction and the actual reverse transition probability q. It is proven that

$$q(x_{t-1}|x_t, x_0) =$$
$$\mathcal{N}(x_{t-1}; \frac{\sqrt{a_{t-1}}\beta_t}{1-a_t}x_0 + \frac{\sqrt{\alpha_t}(1-a_{t-1})}{1-a_t}x_t, \frac{1-a_{t-1}}{1-a_t}\beta_t I)$$

Thus, we can set the variance equal to a constant value and use a neural network to compute the mean value. Instead of trying to predict the value of x_0 from x_t to substitute in the above formula, we first use the relationship:

$$x_t = \sqrt{a_t}x_0 + (1-a_t)\varepsilon \implies x_0 = \frac{x_t + (1-a_t)\varepsilon}{\sqrt{a_t}}$$

Using the known formula for the KL divergence between two Gaussians, we end up minimizing the quantity:

$$L_{t-1} = \mathbb{E}_q\left[\frac{\left\|(\frac{\sqrt{a_{t-1}}\beta_t}{1-a_t}x_0 + \frac{\sqrt{\alpha_t}(1-a_{t-1})}{1-a_t}x_t) - \mu_\theta(x_t, t)\right\|^2}{2\sigma_t^2}\right] + C$$
$$= \mathbb{E}_{t,x_0,\varepsilon}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-a_t)}\left\|\varepsilon - \varepsilon_\theta(\sqrt{a_t}x_0 + \sqrt{1-a_t}\varepsilon, t)\right\|^2\right] + C$$

Therefore, we want to predict the added noise given the point where we are and the corresponding step. A simplified form of the above equation for optimization is also suggested.

C. Score Based Diffusion Models

The Noise Conditional Score Networks, the first Score Based Diffusion models, are introduced by Song et al in [3]. Initially, the more general Score Based Generative Models are based on the idea of estimating the score function $\nabla_x \log p(x)$ with a function $s_\theta(x)$. Two of the main techniques used for this purpose are Denoising Score Matching, which introduces noise $q_\sigma(\chi|x)$ into the data and optimizes the function:

$$\frac{1}{2}\mathbb{E}_{q_\sigma(\chi|x)p(x)}\left[\left\|s_\theta(x) - \nabla_x \log q_\sigma(\chi|x)\right\|^2\right]$$

and Sliced Score Matching, where the function optimized is:

$$\mathbb{E}_{p_{data}}\mathbb{E}_{p_v}\left[v^T\nabla_x s_\theta(x)v + \frac{1}{2}\left\|s_\theta(x)\right\|^2\right]$$

Next, we sample from the distribution p using Langevin Dynamics: We randomly choose an initial point and then apply iteratively the equation:

$$x_t = x_{t-1} + \frac{\varepsilon}{2}\nabla_x \log p(x_{t-1}) + \sqrt{\varepsilon}z_t$$
$$z_t \sim \mathcal{N}(0, I)$$

The above method faces three main problems:

- As our data usually lies on a lower-dimensional manifold, the $\nabla_x \log p(x)$ is not well-defined, and score matching computation techniques do not converge. This problem is solved by adding a very small noise to the data.

- Due to areas in space with low data concentration, score matching will produce poor estimates, since Monte Carlo methods for computing the mean value will not sample from these areas.
- Additionally, areas with low concentration significantly delay the convergence of Langevin Dynamics. That may result to difficulties in the calculation of weights of each sub-distribution and may affect convergence to the distribution.

The above problems are addressed with the idea of Noise Conditional Score Networks. Specifically, consider a decreasing sequence of standard deviations with $\frac{\sigma_1}{\sigma_2} = \ldots = \frac{\sigma_{L-1}}{\sigma_L} > 1$. We construct L sets of data, where in each one, we have added Gaussian noise with variance σ_t^2. Using Denoising Score Matching, the optimization function is:

$$l(\theta, \sigma) = \frac{1}{2}\mathbb{E}_{p_{data}}\mathbb{E}_{\chi\sim\mathcal{N}(x,\sigma^2 I)}\left[\left\| s_\theta(\chi,\sigma) + \frac{(\chi - x)}{\sigma^2}\right\|^2\right]$$

Then, we take a weighted average of the above, so

$$L(\theta, \{\sigma_i\}) = \frac{1}{L}\sum_{i=1}^{L}\lambda(\sigma_i) l(\theta, \sigma_i)$$

Next, for sampling, we use Annealed Langevin Dynamics, a process inspired by Simulated Annealing. We start from a random point and iterate the process of classical Langevin Dynamics for a specific number of repetitions, using the score function $s_\theta(\chi, \sigma_1)$. Then, we repeat the process with $s_\theta(x, \sigma_2)$ until we reach $s_\theta(x, \sigma_L)$. This way, we avoid low probability points in the early iterations, and later, as the sample variance decreases, we take smaller steps near high probability areas. This process closely resembles the noise injection and denoising process of samples in classical diffusion models.

D. Stochastic Differential Equations

The connection between Diffusion Models and Stochastic Differential Equations was illustrated by Song et al in [4]. Instead of using a discrete diffusion schedule where noise is added at each step, it was proposed to model the diffusion process as the solution to an Ito SDE:

$$dx = f(x,t)dt + g(t)dw$$

This equation can be viewed as a process of adding noise w to the data and destroying their structure as time flows. Let $p_t(x)$ be the probability density of x at timestep t. It can be shown that under certain conditions, the above process can be reversed, resulting in the reverse-time SDE:

$$dx = \left[f(x,t) - g(t)^2\nabla_x \log p_t(x)\right]dt + g(t)dw$$

To solve the above equation, we need to know the score function, which can be calculated using Denoising or Sliced Score Matching as in SMLDs, and $p(x_t|x_0)$. Then, we can solve the equation using one of the following techniques:

- Numerical SDE Solvers
- Predictor-Corrector Samplers, which use a Numerical SDE Solver as the predictor and score-based methods as correctors that correct the sample's marginal distribution
- ODE Solvers, where we take the deterministic process corresponding to the diffusion process: $dx = \left[f(x,t) - g(t)^2\nabla_x \log p_t(x)\right]dt$ and solve it numerically.

It can be shown that both DDPMs and SMLDs are discretizations of an underlying SDE. Specifically, the update rule $x_i = x_{i-1} + \sqrt{\sigma_i^2 - s_{i-1}^2}z_{i-1}$ of SMLDs can by modeled by the SDE

$$dx = \sqrt{\frac{d[\sigma^2(t)]}{dt}}dw$$

and the update rule $x_i = \sqrt{1-\beta_i}x_{i-1} + \sqrt{\beta_i}z_{i-1}$ of DDPMs can be modeled by the SDE

$$dx = -\frac{1}{2}\beta(t)xdt + \sqrt{\beta(t)}dw.$$

E. Denoising Diffusion Implicit Models

Despite generating high quality data, DDPMs suffer from very long sampling times. DDIMs were proposed by Song et al in [5] as models that are very similar to DDPMs but significantly faster in sampling. Instead of using a Markovian Forward Process, as in DDPM, the distribution proposed is:

$$q_\sigma(x_{t-1}|x_t, x_0) = \mathcal{N}(\sqrt{\alpha_{t-1}}x_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\frac{x_t - \sqrt{\alpha_t}x_0}{\sqrt{1-\alpha_t}}, \sigma_t^2 I)$$

$$q_\sigma(x_T|x_0) = \mathcal{N}(\sqrt{\alpha_T}x_0, (1-\alpha_T)I)$$

Thus, we can sample from the equation below (where we have substituted the expression for x_0 as in DDPM):

$$x_{t-1} = \sqrt{\alpha_{t-1}}\frac{x_t - \sqrt{1-\alpha_t}\cdot\epsilon_\theta^{(t)}(x_t)}{\sqrt{\alpha_t}} + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\epsilon_\theta^{(t)}(x_t) + \sigma_t\epsilon_t$$

$$\epsilon_t \sim \mathcal{N}(0, I)$$

The training objective $\epsilon_\theta^{(t)}(x)$ is the same as in DDPMs, so we can train the parameters in DDPM and then generate data in this more general setting. By assigning a specific value to σ_t, we get the DDPM generative process, while setting σ_t equal to 0, the process becomes non stochastic.

During sampling, we can make fewer, larger steps instead of the T steps of the forward process, by changing the α_t parameters accordingly, achieving even greater sampling speeds while producing samples of almost the same quality. The deterministic sampling process also ensures that sample generation is consistent, ie a specific noise input will lead to the generation of the same sample output. This property can be useful in tasks such as interpolation.

F. Further Advances

We will now discuss some more recent advances in the field of diffusion models.

In [6], Nichol et al proposed to incorporate the variance into the parameters for training in order to improve the log likelihood. As the values it takes lie between β_t and $b_t = \frac{1-a_{t-1}}{1-a_t}$, the variance is chosen as $\Sigma_\theta(x_{t,t}) = e^{v\log\beta_t + (1-v)\log b_t}$.

Additionally, the schedule of β_t has changed from linear to cosine, as it was observed that a sharp increase in the diffusion rate towards the end resulted in many steps consisting of highly noisy data, effectively not contributing to the quality of the samples. With the new schedule, the number of steps can be reduced. Finally, a hybrid loss function is defined for optimization, consisting of a term with variable variance and a term with a constant: $L_{\text{hybrid}} = L_{\text{simple}} + \lambda L_{\text{vlb}}$. Lastly, instead of giving equal weight to the contribution of each step t in the loss function, a weight schedule p_t is used.

The same authors make some changes to the architecture of the models in [7]. An idea of particular interest is that of Guidance. Specifically, when sampling from a specific class, a term $p_\phi(y|x_t)$, related to the probability of the sample appearing in a specific class is introduced into the product. Taking a Taylor expansion, the mean of the result has an additive term of the form $\Sigma\nabla_{x_t} \log p_\phi(y|x_t)|_{x_t=\mu}$. This term can be viewed as a corrective parameter that "guides" the sample closer to the desired class. We can also use a weighted version of the above, allowing us to value the impact of lying in the correct class. Strong impact leads to better and more relevant samples, at the expense of them being less varied.

Finally, Latent Diffusion Models were introduced by Rombach et al in [8]. One of the main challenges of diffusion models is their high complexity and the substantial computational resources required during sampling. Thus, it is suggested to use an autoencoder which transfers high-dimensional data (e.g., images) to a lower-dimensional latent space before introducing them to the diffusion model. As a result, the model is trained and generates samples in the latent space, with significantly reduced complexity. Subsequently, the samples are brought back to the original space by the decoder.

III. Poisson Flow Generative Models

A. PFGM

The Poisson Flow Generative Model is a physics inspired generative model that was introduced in [9] by Xu et al. The underlying idea is to consider our target data distribution as an electric charge density, and sample by following the electric field lines created.

The foundation of this model is the Poisson Equation:

$$\nabla^2\phi(x) = -\rho(x)$$

where $\rho(x)$ is the source function and $\phi(x)$ is the potential function. Equivalently, using the gradient field $E(x) = -\nabla\phi(x)$, we have:

$$\nabla E(x) = \rho(x)$$

The solution to this equation is:

$$E(x) = -\int \nabla_x G(x,y)\rho(y)dy$$

$$\nabla_x G(x,y) = -\frac{1}{S_{N-1}(1)}\frac{x-y}{\|x-y\|^N}$$

where $S_{N-1}(1)$ is the surface of the unit ball of $N-1$ dimensions. The flow model (noising and denoising) is described by the forward ODE $\frac{dx}{dt} = E(x)$ and the backward ODE $\frac{dx}{dt} = -E(x)$.

When $\|x\| \gg \|y\|$, $E(x) \sim \frac{x}{\|x\|^N}$. Thus, when we move away from our initial distribution, the field behaves as if it was generated by a single unit point source at 0. Using this intuition, we can understand that the forward process maps our initial data distribution to a uniform distribution in an infinite radius half sphere.

The backward ODE introduced above suffers from the problem of mode collapse, as the field often points towards a few points. To overcome this difficulty, we augment our space by one extra dimension z. We place our data in the hyperplane $z = 0$. It can be proven that starting from a uniform distribution in the half sphere and moving along the field lines until $z = 0$, the sampling distribution will be our source (data) distribution.

During training, we first perturb the training data and then use them to try to predict a normalized version of the gradient field $\hat{E}$. In the backward ODE discussed above, the boundary conditions are unclear, since we don't know the timesteps when we are in the half sphere or when we reach the hyperplane $z = 0$. To remedy this problem, we change the ODE to evolve with the variable z. Instead of starting from the half sphere, we start from its projection to a large z_{max} and move towards $z = 0$.

This model achieves great results on image generation, with FID and Inception scores near or surpassing state of the art. It is also faster and more robust than other flow and SDE based models, without having to rely on corrector techniques. Another nice property of the model is that it is invertible (non stochastic), which is useful for tasks such as latent representation.

B. PFGM++

PFGM++ is a family of generative models that was introduced by Xu et al in [10] as a generalization of PFGM. The new idea is to augment our data dimension by a value D that can be greater than 1. Instead of using the hyperplane $z = z_{max}$ as the starting distribution, we now start from the hypercylinder $r = \sum_{i=1}^{D} z_i^2 = r_{max}$. It is proven that as $r_{max} \to \infty$, the ODE $\frac{dx}{dr} = \frac{E(\tilde{x})_x}{E(\tilde{x})_r}$, which is the analogous to the PFGM ODE for $D > 1$, maps the hypercylinder to the initial data distribution. The training objective is also slightly changed to reduce the large batch size that was previously required.

The interesting property of this D-formulation is that as $D \to \infty$, the PFGM++ reduces to classic diffusion models, while as $D \to 1$, the model gets closer to PFGM. The advantages of having smaller values for D is that the model becomes more robust and is thus less prone to estimation errors. Therefore, with smaller D, we can make use of smaller architectures using more aggressive quantization and larger step sizes. On the other hand, larger values for D lead to more rigid models, that are easier to train. In order to train

TABLE I
FID SCORES ON CIFAR10

D	128	2048	∞
FID	4.02	3.96	4.04

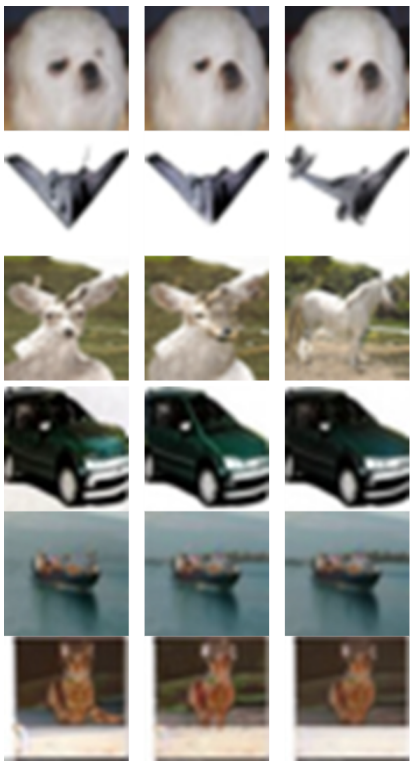

Fig. 1. Examples of image generation for $D = 128$ (left), $D = 2048$ (center) and $D = \infty$ (right)

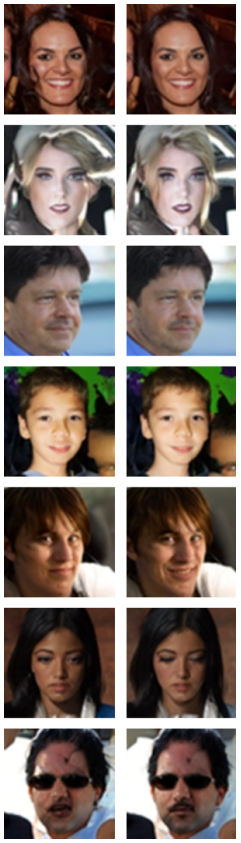

Fig. 2. FFHQ images generated for $D = 128$ (left) and $D = \infty$ (right)

these small-D models such as PFGM, several heuristics need to be applied, that prevent the model from replicating the initial data distribution. By finding a *sweet spot between* $D = 1$ *and* $D \to \infty$*, we can balance robustness and rigidity and achieve the best performance.*

IV. EXPERIMENTS

Below, we make some experiments for models of varying D *values* $(128, 2048, \infty)$ *trained in the CIFAR10 dataset. We show some images generated in Figure 1 and calculate the corresponding FID scores, which are presented in Table I.*

We also show some images generated from models trained in the FFHQ dataset for models with D *values* $(128$ *and* $\infty)$ *in Figure 2.*

We notice that the samples generated for the same seeds are mostly similar. The FID scores are also quite similar, with the one for $D = 2048$*, a sweet spot between PFGM (low* D *values) and DDPM (high* D *values) being the best.*

We then train the PFGM++ model in the MNIST dataset and present our results. Since our computational resources were limited, we chose a smaller version of the model, with the hidden channel multiplier (a parameter similar to hidden layer size) set to 16. We also used a small batch size of 16, in

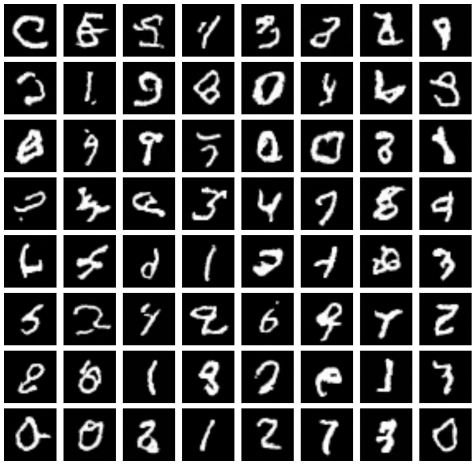

Fig. 3. MNIST images generated by our model for $D = 16$

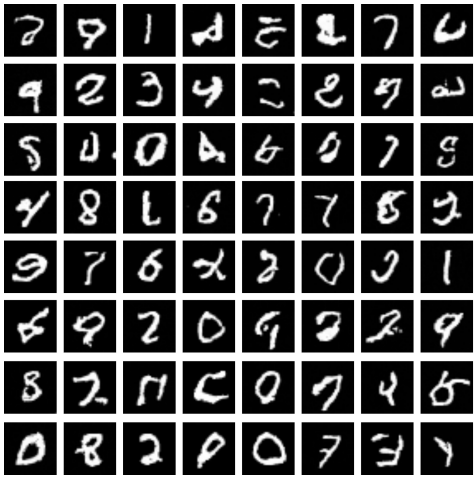

Fig. 4. MNIST images generated by our model for $D = 2048$

order to fit the data into the graphics card memory. We chose the D parameter to be 16, since fewer additional dimensions lead to more robust models that can fit in smaller architectures. After training for 600 kimg (thousand iterations over images), we achieved an FID score of 24.48. Some images generated by our trained model are presented in Figure 3. Despite not entirely capturing the MNIST dataset details, some produced samples are very close to resembling handwritten digits.

When training a model with $D = 2048$ for the same number of iterations, we got a better FID score of 24.04. Some images generated by this model are presented in Figure 4.

V. Conclusion

We present the basic ideas behind Diffusion Models and summarize the most important papers in the field. Focusing on PFGM, we highlight their impressive data generation capabilities and even train a toned-down version of the model for the MNIST dataset. The most important problem that diffusion models face is their heavy computational requirements, particularly during sampling.

References

[1] *Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In International Conference on Machine Learning, pages 2256–2265, 2015.*

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

[3] *Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. arXiv preprint arXiv:1907.05600, July 2019.*

[4] *Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. arXiv preprint arXiv:2011.13456, 2020.*

[5] *Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. ArXiv, abs/2010.02502, 2021.*

[6] *Alexander Quinn Nichol and Prafulla Dhariwal. 2021. Improved denoising diffusion probabilistic models. In International Conference on Machine Learning. 8162–8171.*

[7] *Prafulla Dhariwal and Alexander Nichol. 2021. Diffusion models beat gans on image synthesis. In Advances in Neural Information Processing Systems, Vol. 34. 8780–8794.*

[8] *Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In IEEE Conference on Computer Vision and Pattern Recognition. 10684–10695.*

[9] *Xu, Y., Liu, Z., Tegmark, M., and Jaakkola, T. Poisson flow generative models. ArXiv, abs/2209.11178, 2022.*

[10] *Yilun Xu, Ziming Liu, Yonglong Tian, Shangyuan Tong, Max Tegmark, and T. Jaakkola. Pfgm++: Unlocking the potential of physics-inspired generative models. ArXiv, abs/2302.04265, 2023.*