EXPLORING NOVEL SAMPLING TECHNIQUES IN SDE-BASED DENOISING GENERATIVE MODELS *

Ninad Gandhi
MS by Research, C-MInDS
IIT Bombay
22m2151@iitb.ac.in

Prabhat Reddy
MS by Research, C-MInDS
IIT Bombay
22m2156@iitb.ac.in

Sachin Giroh
MS by Research, C-MInDS
IIT Bombay
22m2159@iitb.ac.in

Jimut Bahan Pal
PhD, C-MInDS
IIT Bombay
22d1594@iitb.ac.in

1 Task Description

1.1 Learn about efficient sampling from diffusion models

Diffusion models are generative models that can generate high-quality data samples from the training distribution. But, diffusion models are notoriously inefficient to sample from. In our case, we focus on an image-based dataset. Sampling from a diffusion model starts with a completely random noisy image, and the model slowly transforms this noise into an image that imitates our actual distribution. Sampling takes a lot of time because of the denoising procedure that takes an order of thousand steps to go from noise $\mathbf{x}_T$ to complete image $\mathbf{x}_0$[1]. For each time step $t \in [N]$, we have to compute a transformation from $\mathbf{x}_t$ to $\mathbf{x}_{t-1}$ (because $\mathbf{x}_0$ is the target). This is called the reverse diffusion process, which removes noise instead of the forward process that adds it.

Since diffusion models were first published by in the paper Denoising Diffusion Probabilistic Models (DDPM) [1] by Ho et al., 2020, there has been a lot of research in making the sampling procedure efficient. One of the approaches is Score-Based Generative Modeling Through Stochastic Differential Equation[2], where they evaluate a score function $\nabla_x \log p_t(\mathbf{x})$ using a neural network $s_\theta(\mathbf{x}, t)$. This score function is a measure of the gradient in the probability space, and we can do gradient ascent to a higher probability region and consequently generate an image that imitates the actual distribution. Another important contribution was made by Qinsheng Zhang and Yongxin Chen in the paper Fast Sampling of Diffusion Models With Exponential Integrator[3], where they proposed a diffusion exponential integrator sampler (DEIS) which can generate very high-quality images using as less than 10 steps or score function evaluations. Their technique, DEIS, can be used with any pre-trained model.

Hence, through our course project, we aim to explore various efficient sampling techniques for diffusion models without trading off the quality of generated images.

1.2 Implement a sampling algorithm using DEIS

We attempt to implement a sampling algorithm using the DIES sampler provided in the GitHub repository code. Specifically, we tried to implement fifth-order Dormand-Prince ODE solver. But because of resource constraint we are not able to conduct the experiment.

*Course project report for CS726 - Advanced Machine Learning taught by Prof. Sunita Sarawagi. Code is available at https://github.com/pace577/score_sde_pytorch.

2 Challenges Addressed

2.1 Conceptually challenging topics with mathematical derivations

Diffusion models are relatively new, and the literature around them is constantly evolving. A lot of research on diffusion models is published annually, and researchers look at it from different perspectives. It is very cumbersome to parse all the research before getting lost in the other notational conventions used by different researchers.

Another essential aspect involved with understanding Diffusion models is the number of concepts involved. It also demands an understanding of complex concepts in probability theory, probabilistic graphical models, and even knowledge of solving differential equations.

2.2 Getting the code to run

The code provided by the authors of the paper has used the Jax library by Google for their implementation. Though Jax is more efficient than Python in terms of speed, it has a much larger memory footprint. This made us port the code from Jax to Pytorch, which was a difficult task. Another issue with the code was setting up the environment. The environment.yml file provided in the GitHub repository had a lot of outdated versions of libraries like TensorFlow==2.4.0, which has been discontinued and is not available anywhere. The code does not run with any other version of TensorFlow. Hence, the environment setup took a lot of time. Our porting of the code will ensure reproducibility for further scientific research in the field.

3 Methodology

Diffusion models [1] have recently shown great generative capabilities compared to models like GANs, with great scalability, stable training and low hyper-parameter sensivity. Though, this comes at the cost of slow sampling. The main idea in our project is to explore better and faster sampling methods and make the sampling procedure of DMs efficient. To this end, there are two classes of methods. First class of methods optimize forward noising process which in turn makes the backward denoising process efficient. The other class of methods make the process of solving SDE or ODE's faster.

3.1 Score-based generative modelling

Score Function is defined as the gradient of the log probability density function. It helps to remove the intractable normalizing constant term. Score-based models have achieved state-of-the-art performance on many downstream tasks and applications.[4] The Training of score-based models is done by minimizing the Fisher divergence.

$$\mathbb{E}_{p(x)}[\|\nabla_x \log p(x) - s_\theta(x)\|_2^2] \tag{1}$$

However as we don't know the data distribution, we can not calculate the ground truth score directly. To overcome this problem, the technique of Score/Sliced Score Matching comes handy. Score/Sliced Score Matching eliminates the data score using integration by parts and the final term contains only the network gradients and constants.[5] Score matching objective can be optimized with stochastic gradient descent, analogous to the log-likelihood objective.

After Training of score based model, the sampling is done through the Langevin Dynamics. Langevin dynamics accesses p(x) only through the score function. The Limitation of this approach is that the estimated score functions are inaccurate in low density regions, where few data points are available for computing the score matching objective.[2] Also, Realistic Data is often sparsely distributed, hence our initial sample is highly likely to be in the low density regions, hence, Inaccurate score-based model will derail Langevin dynamics from the very beginning. Now the question arises as how to accurately estimate the score function in regions of low data density?

One of the possible Solution for the above problem is to perturb the data points with noise and train the score-based models on the noisy data points.[2] But then the question arises as how to choose the appropriate noise scale?As Larger noise will over-corrupt the data and smaller noise does not cover the low density region for better score estimation. The possible Way forward is to use Multiple scales of noise perturbations simultaneously. One of the model based on the above approach is Noise Conditional Score-based model. As it is observed that the addition of multiple noise scales is critical to the generative models. The other possible solution is to perturb the data with stochastic differential equation. Many stochastic processes (diffusion processes in particular) are solutions of stochastic differential equations (SDEs). Process of noise addition is a continuous stochastic process. Solution of a stochastic differential equation is a continuous collection of random variables.

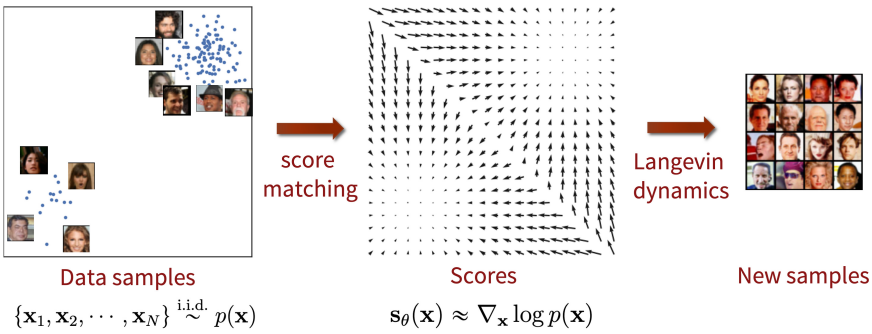


Figure 1: Score Matching with Langevin Dynamics (SMLD)

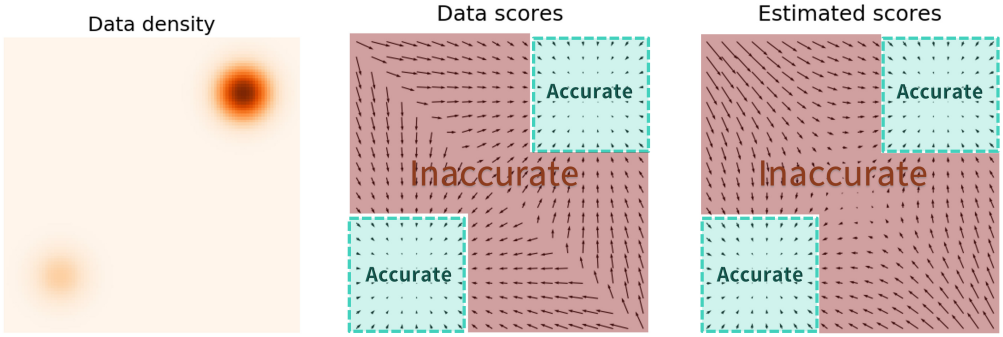


Figure 2: Pitfalls of Langevin MCMC: Score function

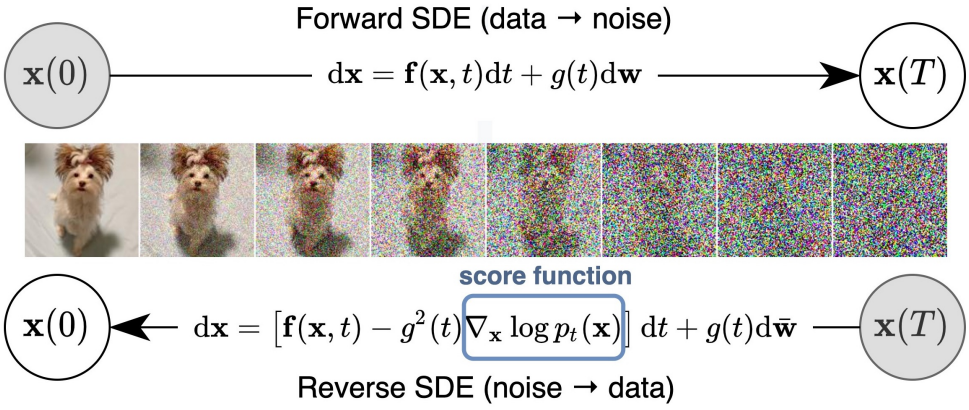
If the input data is perturbed in a continuous-time stochastic process, we obtain - Higher quality samples, Exact log-likelihood computation and Controllable generation for inverse problem solving. [2]$P(x_0)$ denotes the original distribution and $P(x_T)$ denotes the Prior distribution. Forward Process of SDE is defined as -

$$dx = f(x,t)dt + g(t)dw \tag{2}$$

Importantly, any SDE has a corresponding reverse SDE[6] -

$$dx = [f(x,t) - g(t)^2 \nabla_x \log p_t(x)]dt + g(t)d\bar{w} \tag{3}$$

For sample generation, we need to solve the reverse SDE. Training of a Time-Dependent Score-Based Model is to be done for the score based network. Subsequently the reverse SDE is solved through numerical solver methods such as Euler-Maruyama and Runge Kutta method[7]. SDE solvers are time consuming and this results in slowing down of the sampling speed.



Bottleneck for diffusion models is the sampling process. Sampling is slower in comparison with other generative models. Sampling efficiency metric is the number of network function evaluations (NFE) required to generate an image of desired quality.[3] Eg: If a Denoising Diffusion Prbabilistic Model (DDPM) takes 1000 steps to generate 1 sample, and each step requires evaluating the learnt neural network once, then NFE=1000.[1] Sample quality metrics are FID, inception score

One of the possible approach to increase the sampling speed is through probability flow ODE, it converts SDE to ODE, without changing the marginal distribution.[2]

$$dx = [f(x,t) - \frac{1}{2}g(t)^2 \nabla_x \log p_t(x)]dt \tag{4}$$

Thus by solving this ODE, we can sample from the same distributions as the reverse SDE. The probability flow ODE becomes a special case of a neural ODE. In particular, PF is an example of continuous normalizing flows since the probability flow ODE converts a data distribution to a prior noise distribution (since it shares the same marginal distributions as the SDE) and is fully invertible.

Denoising Diffusion Prbabilistic Model (DDPM) uses ancestral sampling, but doesn't use a langevin-based step. Score Matching Langevin Dynamics (SMLD) samples according to langevin dynamics only. What if we use both sampling techniques? Will it improve sample quality? Yes as we use the best of bth the world. Predictor-Corrector samplers first make a prediction step (solve the Differential Equation) followed by a corrector step (gradient ascent). Predictor is any numerical SDE solver that predicts $x(t + \Delta t)$ Corrector is any MCMC approach that uses score function. This makes sampling more efficient and improves sampling quality, when compared to using predictor-only or corrector-only samplers.

3.2 Fast sampling with Diffusion Exponential Integrator Sampler (DEIS)

As discussed at the start of this section, there are two approaches to make the sampling process of diffusion models efficient. The first approach optimizes the forward process, while the second approach finds efficient ways to solve the backward denoising process SDE defined by the score function $\nabla_x \log p_t(\mathbf{x})$. The authors in this paper try to use the second approach by implementing exponential integrator sampler that can efficiently solve the SDE (Network score Function Evaluations (NFE) are very less) associated with the score function, without trading off image quality, assuming $s_\theta(\mathbf{x}_t, t) \approx \nabla_x \log p_t(\mathbf{x})$.

Let us first mention the forward noising and reverse denoising processes in 1 and 2, respectively.

$$d\boldsymbol{x} = \boldsymbol{F}_t \boldsymbol{x} dt + \boldsymbol{G}_t d\boldsymbol{w}, \tag{1}$$

Here, $\mathbf{F}_t$ is the linear drift coefficient and $\mathbf{G}_t$ is the diffusion coefficient. $\mathbf{w}$ is the standard wiener process that injects stochasticity in both forward and reverse process. Given the forward process, Song et al.[2] show that the corresponding reverse-time diffusion process is the following

$$d\boldsymbol{x} = \left[\boldsymbol{F}_t \boldsymbol{x} dt - \boldsymbol{G}_t \boldsymbol{G}_t^T \nabla \log p_t(\boldsymbol{x})\right] dt + \boldsymbol{G}_t d\boldsymbol{w}, \tag{2}$$

where, again w is the standard wiener process. $\nabla \log p_t(\mathbf{x})$ is the score function that we would like the neural network s_θ to approximate. We define $p_t(\mathbf{x}_t)$ as the marginal distribution of $\mathbf{x}_t$. Also, we denote $p_{0t}(\mathbf{x}_t|\mathbf{x}_0)$ as the conditional of $\mathbf{x}_t$ given $\mathbf{x}_0$. We define the training objective

$$\mathcal{L}(\theta) = \mathbb{E}_{t\sim \mathrm{Unif}[0,T]} \mathbb{E}_{p(\boldsymbol{x}_0)p_{0t}(\boldsymbol{x}_t|\boldsymbol{x}_0)} \left[\|\nabla \log p_{0t}(\boldsymbol{x}_t|\boldsymbol{x}_0) - \boldsymbol{s}_\theta(\boldsymbol{x}_t, t)\|^2_{\Lambda_t} \right]. \tag{3}$$

where $\nabla \log p_{0t}(\mathbf{x}_t|\mathbf{x}_0)$ can be solved exactly since $p_{0t}(\mathbf{x}_t|\mathbf{x}_0)$ is a simple Gaussian. Her, we are trying to minimize the score matching loss.

3.2.1 Using learned scored models

If we have a score network trained with sufficient accuracy i.e. $s_\theta(\mathbf{x}_t, t) \approx \nabla_x \log p_t(\mathbf{x})$, we can use it to generate new samples by using 2 and solving the SDE. The authors propose another family of SDEs defined as

$$d\hat{\boldsymbol{x}} = \left[\boldsymbol{F}_t \hat{\boldsymbol{x}} - \frac{1+\lambda^2}{2} \boldsymbol{G}_t \boldsymbol{G}_t^T \boldsymbol{s}_\theta(\hat{\boldsymbol{x}}, t) \right] dt + \lambda \boldsymbol{G}_t d\boldsymbol{w}, \tag{4}$$

The SDEs are parameterized by λ. When $\lambda = 1$, we exactly have the **SDE in 2**. Alternately, if $\lambda = 0$, the stochasticity is removed from the differential equation and becomes an ODE, called **probability flow ODE**. In general, we have $\lambda \geqslant 0$.

If we have $s_\theta(\mathbf{x}_t, t) = \nabla_x \log p_t(\mathbf{x}) \quad \forall \mathbf{x}, t$, then it has been shown by [8] that the estimated $\hat{p}^*_T = \pi$ exactly matches the actual $p_t \quad \forall 0 \leqslant t \leqslant T$. The problem is that this assumption is not true for most choices of $\mathbf{x}, t$. Consequently, we have to discretize 4 and get an approximate solution. This gives rise to two types of errors in the approximation.

Fitting error: This error arises due to the mismatch between ground truth and the learned score network.

Discretization error: This error is introduced due to the numerical approximations involved in the discretization process.

The authors recommend finding a good discretization scheme to mitigate the effects of fitting errors. It is intuitive because reducing the discretization error involves choosing the discretization steps carefully, which can imply that we step through the regions of $\nabla \log p(\mathbf{x})$ that correspond to a good fit of the score function.

3.2.2 Discretization error and designing DEIS

The authors go on to propose the estimation of discretization error for probability flow model and provide the DEIS algorithm for mitigating it. First they define the following dynamics which correspond to $\lambda = 0$ ODE

$$\frac{d\hat{\boldsymbol{x}}}{dt} = \boldsymbol{F}_t \hat{\boldsymbol{x}} - \frac{1}{2} \boldsymbol{G}_t \boldsymbol{G}_t^T \boldsymbol{s}_\theta(\hat{\boldsymbol{x}}, t). \tag{5}$$

which has the following exact solution equation

$$\hat{\boldsymbol{x}}_t = \Psi(t, s)\hat{\boldsymbol{x}}_s + \int_s^t \Psi(t, \tau) \left[-\frac{1}{2} \boldsymbol{G}_\tau \boldsymbol{G}_\tau^T \boldsymbol{s}_\theta(\hat{\boldsymbol{x}}_\tau, \tau) \right] d\tau, \tag{6}$$

where the function $\Psi(t, s)$ satisfies the following two properties

$$\frac{\partial}{\partial t} \Psi(t, s) = \mathbf{F}_t \Psi(t, s)$$

$$\Psi(s, s) = \mathbf{I}$$

Euler vs Exponential Integrator method: The first thing is choosing a good ODE solver so that the discretization error is minimized. Even though Euler method is widely used as a numerical solver, it is a first order method. The equation for solving the ODE using Euler's method is given as follows

$$\hat{\boldsymbol{x}}_{t-\Delta t} = \hat{\boldsymbol{x}}_t - \left[\boldsymbol{F}_t \hat{\boldsymbol{x}}_t - \frac{1}{2} \boldsymbol{G}_t \boldsymbol{G}_t^T \boldsymbol{s}_\theta(\hat{\boldsymbol{x}}_t, t) \right] \Delta t. \tag{7}$$

The EI method described by 8 uses higher order ODE solvers and is expected to give better results. Though this does not happen straightaway, which is not trivial and is surprising even to the authors. Following is the EI integrator solution equation

$$\hat{\boldsymbol{x}}_{t-\Delta t} = \Psi(t - \Delta t, t)\hat{\boldsymbol{x}}_t + \left[\int_t^{t-\Delta t} -\frac{1}{2} \Psi(t - \Delta t, \tau) \boldsymbol{G}_\tau \boldsymbol{G}_\tau^T d\tau \right] \boldsymbol{s}_\theta(\hat{\boldsymbol{x}}_t, t). \tag{8}$$

Note that the bad performance of EI can be attributed to the fact that the above system has an approximation where $\boldsymbol{s}_\theta(\hat{\boldsymbol{x}}_t, t)$ is outside the integral and we assume that for a small step $t + \Delta t$, the score function does not change much. This does not hold in general. The first image in 3 shows this result empirically. We define score approximation error as $\Delta_s(\tau) = \|s_\theta(x_\tau, \tau) - s_\theta(x_t, t)\| \quad \forall \tau \in [t - \Delta t, t]$, then we have the plot of Δ_s against t as in the figure. This problem can be solved by a reparameterization introduced next.

Proposing an ϵ_θ parameterization for s_θ: The score function s_θ is parameterized by ϵ_θ with the relation $s_\theta = -\mathbf{L}_t^{-T} \epsilon_\theta$, where $\mathbf{L}_t$ is any matrix that satisfies $\mathbf{L}_t \mathbf{L}_t^T = \Sigma_t$. Consequently, we are trying to learn a parameterized score function $\nabla \log p_t(\mathbf{x}) \approx -\mathbf{L}_t^{-T} \epsilon_\theta(\mathbf{x}, t)$. We observe from figure 3 second plot that Δ_s greatly reduces when the ϵ_θ reparameterization is applied. When we use ϵ_θ approximation in the EI equation, we get the following

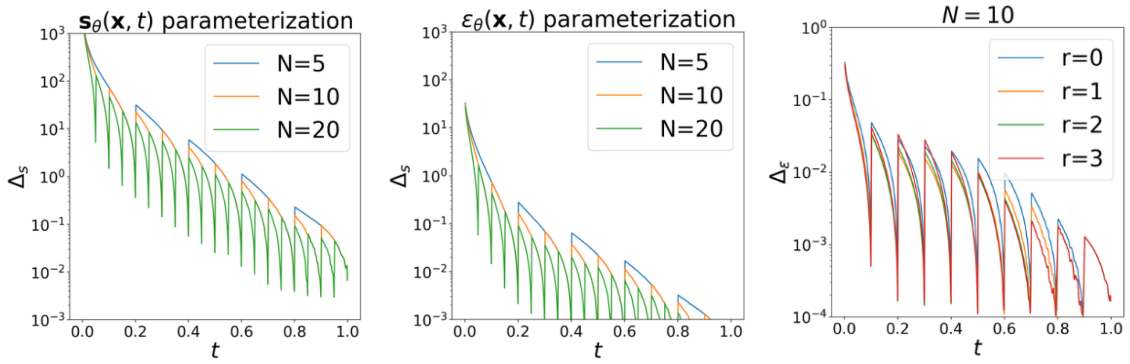


Figure 3: Improving upon score approximation.[3]

$$\hat{\boldsymbol{x}}_{t-\Delta t} = \Psi(t-\Delta t, t)\hat{\boldsymbol{x}}_t + \left[\int_t^{t-\Delta t} \frac{1}{2}\Psi(t-\Delta t, \tau)\boldsymbol{G}_\tau \boldsymbol{G}_\tau^T \boldsymbol{L}_\tau^{-T} d\tau\right] \epsilon_\theta(\hat{\boldsymbol{x}}_t, t). \tag{9}$$

The above equation uses a zeroth order approximation of ϵ_θ because $\epsilon_\theta(\hat{\boldsymbol{x}}_t, t)$ is a constant for a give t. Here comes the last piece of the DEIS sampler.

Higher order approximation for ϵ_θ: For a time discretization $\{t_i\}_{i=0}^N$, with $t_0 = 0$ and $t_N = T$, we fit a polynomial $\mathbf{P}_r(t)$, a degree-r polynomial, for each i. For an interpolation point $(t_{i+j}, \epsilon_\theta(\hat{\mathbf{x}}_{t_{i+j}}, t_{i+j})), \quad 0 \leqslant j \leqslant r$, $\mathbf{P}_r(t)$ is given as

$$\boldsymbol{P}_r(t) = \sum_{j=0}^{r} \left[\prod_{k\neq j} \frac{t - t_{i+k}}{t_{i+j} - t_{i+k}}\right] \epsilon_\theta(\hat{\boldsymbol{x}}_{t_{i+j}}, t_{i+j}). \tag{10}$$

Indeed, the higher order approximation $\mathbf{P}_r(t)$ yields much better estimates of ϵ_θ. This is also apparent when a very less NFE yields high-quality images. Observe in the third plot of figure 3 that as the value of r increases, the approximation of ϵ_θ becomes better and consequently, Δ_ϵ reduces. Using higher order approximations, when number of discretization steps is small $(N \approx 10)$, leads to better quality samples.

Finally, replacing the higher order approximations for ϵ_θ is in the ODE equation gives the equation 11, where the coefficient C_{ij} is described in the next equation 12.

$$\hat{\boldsymbol{x}}_{t_{i-1}} = \Psi(t_{i-1}, t_i)\hat{\boldsymbol{x}}_{t_i} + \sum_{j=0}^{r} [\mathrm{C}_{ij}\epsilon_\theta(\hat{\boldsymbol{x}}_{t_{i+j}}, t_{i+j})] \tag{11}$$

$$\mathrm{C}_{ij} = \int_{t_i}^{t_{i-1}} \frac{1}{2}\Psi(t_{i-1}, \tau)\boldsymbol{G}_\tau \boldsymbol{G}_\tau^T \boldsymbol{L}_\tau^{-T} \prod_{k\neq j} [\frac{\tau - t_{i+k}}{t_{i+j} - t_{i+k}}] d\tau. \tag{12}$$

4 Implementation and Results

Attach plots and figures here. Discuss results

5 Related Work

The usage of score function, which is the gradient of the likelihood, in generative modeling was first motivated by Song et al.[7], in which ideas from Langevin dynamics for generative modeling[9] were used in along with score functions. Neural networks with similar input and output dimensions were used to learn the score function through the process of score-matching[10]. By using a annealed noise schedule, Song et al. came up with Annealed Langevin Dynamics sampling[7].

The family of denoising diffusion probabilistic models (DDPM) [11] add noise to datapoints in the training set and attempts to learn the reverse denoising process, so that the model can learn how to generate samples given a sample that comes from a certain prior distribution. While DDPMs don't seem to have to do anything with score-based models, it turns out that the training objective looks quite similar to the score matching objective in [7].

Parallels between DDPM and SMLD are drawn in [2] in which stochastic differential equations that govern the continuous time stochastic processes corresponding to both DDPM and SMLD are obtained. Using the work by Anderson[12], reverse SDEs are obtained for both the equations. Moreover, a novel Predictor-Corrector sampling technique is discussed in [2].

A non-markovian approach to diffusion models is discussed in [4], which improved the sampling efficiency. A fast sampling method using an exponential integrator is discussed in [3], in which the authors closely examine the sources of error in the generation process of diffusion models and develop a sampling method that attempts to minimize these detected errors.

reading the fast sampling process:

6 Conclusion

We have gone through lots of literature in the domain of score-based generative models and familiarized ourselves with various sampling techniques and existing state of the art models.

We have tried to compile all that information in this project presentation and tried to make it easier to digest.

We have tried to understand the implementation of the code provided by the authors. Since the implementation uses a very old environment configuration of python packages, it is very tricky to build the environment. We have tried to do this manually but there are some compatibility issue that we cannot resolve. Despite this, we have had success in generating some image samples from a pre-trained model on CIFAR-10.

Acknowledgements

We would like to thank the creators of the original repository by the authors of the paper[2], available at https://github.com/yang-song/score_sde. The classes from CS726 by Prof. Sunita Sarawagi on Advanced Machine Learning were found to be very helpful in understanding and development of this project.

Work Split

- **Literature Review** - Done by all
- **Score based generative modelling** - Sachin and Prabhat
- **Fast sampling with DEIS** - Ninad and Jimut
- **Working on code** - All have contributed equally to the development of the code. Prabhat has understood the code base most deeply.
- **Presentation** - Prabhat and Sachin
- **Report** - Ninad and Jimut

References

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. CoRR, abs/2006.11239, 2020.

[2] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In International Conference on Learning Representations, 2021.

[3] Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. arXiv preprint arXiv:2204.13902, 2022.

[4] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. arXiv preprint arXiv:2010.02502, 2020.

[5] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In Uncertainty in Artificial Intelligence, pages 574–584. PMLR, 2020.

[6] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. Advances in Neural Information Processing Systems, 34, 2021.

[7] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. Advances in neural information processing systems, 32, 2019.

[8] Qinsheng Zhang and Yongxin Chen. Diffusion normalizing flow. Advances in Neural Information Processing Systems, 34, 2021.

[9] 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. PMLR, 2015.

[10] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. Journal of Machine Learning Research, 6(4), 2005.

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. Advances in Neural Information Processing Systems, 33:6840–6851, 2020.

[12] Brian DO Anderson. Reverse-time diffusion equation models. Stochastic Processes and their Applications, 12(3):313–326, 1982.