CS 726: Final Project Report

Tackling the Generative Learning Trilemma

May 2023

Contents

1 Task Description 1

2 Challenges Addressed 1

3 Denoising Diffusion Probabilistic Models 1
3.1 MODEL ARCHITECTURE 1
3.2 Results and Observations 2

4 Variational Diffusion Models 3
4.1 Background : 3
4.2 Outline of the method : 4
4.3 Experiment details and main results: 5
4.4 An observation 7
4.5 Loss comparison 7

5 DDPMs + GANs 8
5.1 Outline of the method : 8
5.2 Outline of the training, sampling methods 8
5.3 Experiment details and Main results : 8

6 Related Work : Inference of diffusion models via shortcut MCMC sampling 9
6.1 Outline 9
6.2 Results 10

7 Conclusion : 10

8 Work split up among team members : 10

1 Task Description

All Generative models struggle with simultaneously addressing three key requirements including: high sample quality, mode coverage, and fast sampling (generative learning trilemma) as they often trade some of them for others. Particularly, DDPM qualiy is good, but sampling is expensive. As part of this project, we tried to address this problem by combining the strengths of various generative models, compare them and also tried to combine the goods of different models.

2 Challenges Addressed

Generally, Diffusion process takes hundreds to thousands of steps. One area where we can improvise is the nature of the denoising distribution. Using conditional GANs, we attempt to model complex denoising distributions in the image domain.
Variational Diffusion models belong to a class of likelihood based generative modeling. Using Variational Diffusion Models, we can model complex distributions and effectively use conditioning information for generation tasks. Also, unlike other diffusion models, there is a scope for joint optimization of the noise schedule with the rest of the model. As VDMs are trained to perform Image Density Estimation, they can generate high quality samples.

3 Denoising Diffusion Probabilistic Models

We reimplemented Simple Diffusion for the CIFAR10 dataset. The results are summarized as follows:

- We used the **UNET** architecture as used in the DDPM paper.
- We started from a low value of T and continuously increased it to see how many steps we need for good samples.
- We used a (32,32) $\longrightarrow$ (16,16) resized version of the CIFAR10 dataset, and also took only the Airplane class images for experimenting with Normal DDPM.

3.1 MODEL ARCHITECTURE

The unet model takes a (3,S,S) image, (S = 16 in our case), and it continuously convolutes the image into more and more channels in the expanding path / down path.

And the upsampling path just does the reverse of upsampling, but including some connections from the downsampling path. The connections, known as **skip connections** bypass one or more levels in the expanding path and link them to the corresponding layers in the contracting path. They enable high-level and low-level information from the input image to be incorporated into the model output.

We explored with the channel sequence in the downsampling path. Like (64,128,256) etc.

Time Embedding dimension = 32 was used in all our experiments, we didnt change this.

3.2 Results and Observations

· When the number of steps is small like (50, 100 , 20) etc. The samples obtained are just like noise, with slightly better lokking images if we used a heavier model(more number of levels in Unet model).

· We got smoother samples only for T = 500 , 1000 . But time increases ofcouse, and to sample 10 images in T = 1000 case, it took nearly 1.5 minutes.

· we tried with T = 2000. But now the sampling process becomes tooo expensive, uses too much memory so we did not try this.

OBTAINED SAMPLES

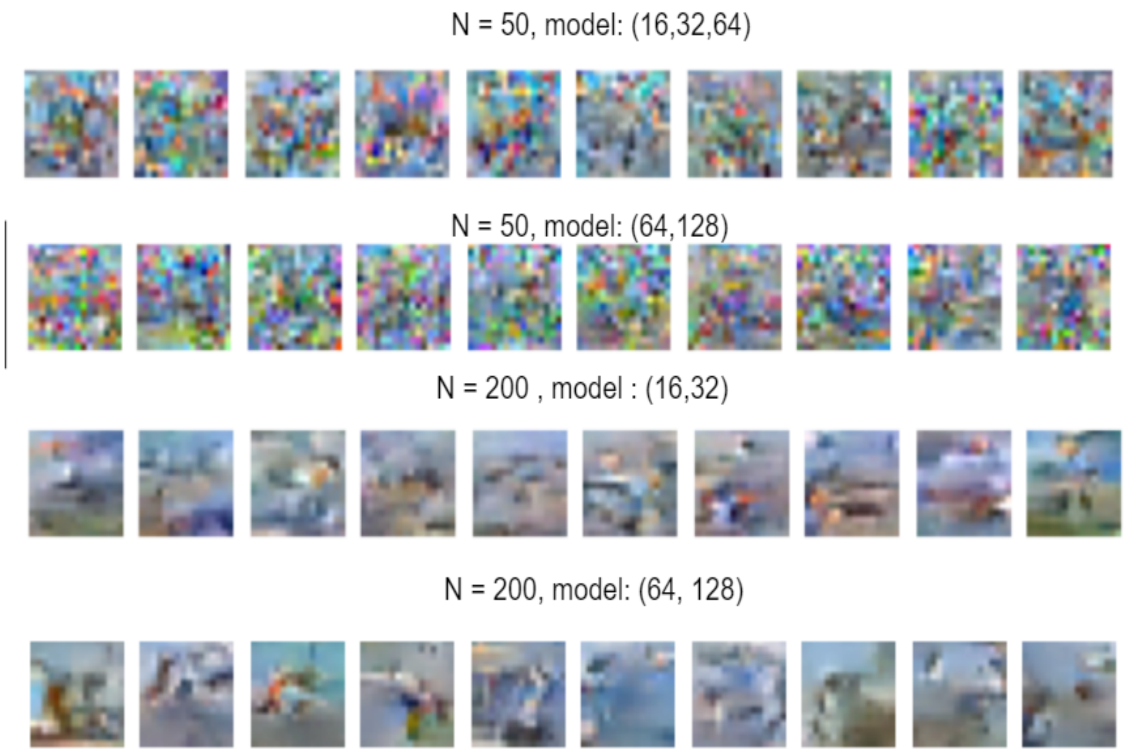
N = 50, model: (16,32,64)
N = 50, model: (64,128)
N = 200 , model : (16,32)
N = 200, model: (64, 128)

Figure 1: A few samples from the basic DDPM

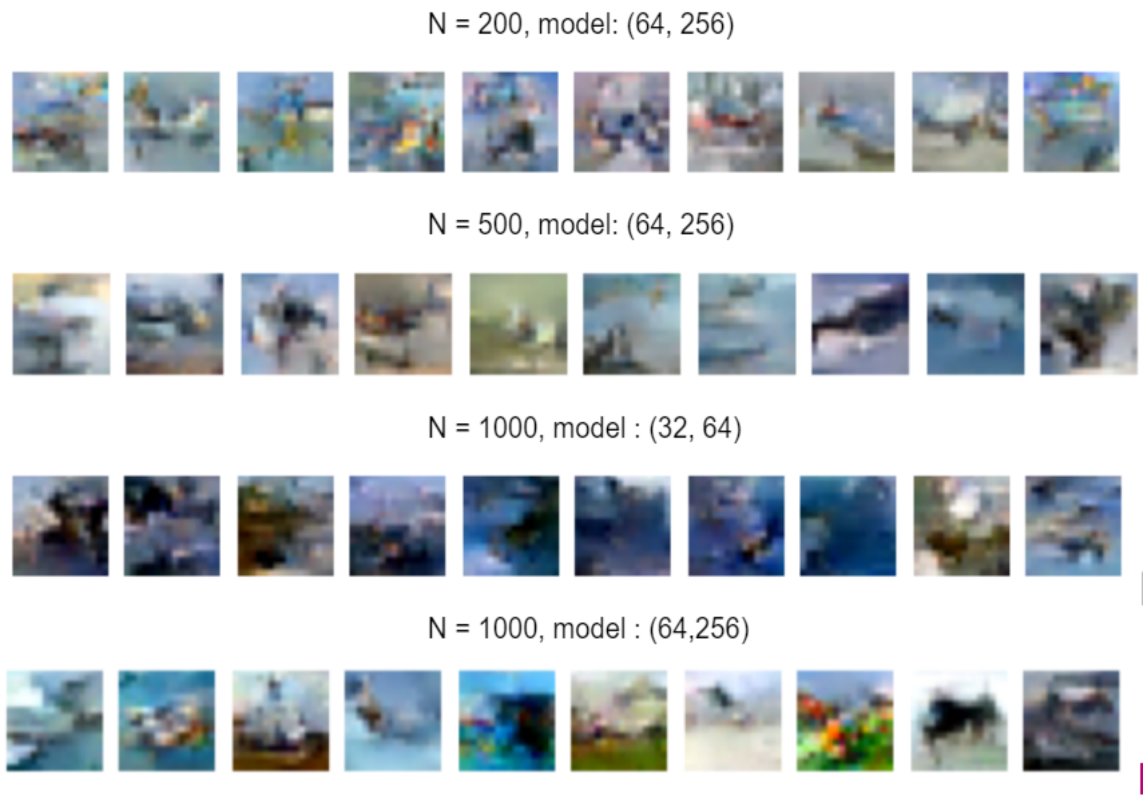
N = 200, model: (64, 256)
N = 500, model: (64, 256)
N = 1000, model : (32, 64)
N = 1000, model : (64,256)

Figure 2: A few samples from the basic DDPM

4 Variational Diffusion Models

4.1 Background :

In a variational autoencoder, we try to learn underlying latent variables that describe our observed data. We try to directly maximise the ELBO in the default formulation of a VAE.

A Hierarchical VAE is a generalization of a VAE that extends to multiple hierarchies over latent variables, i.e, there are multiple levels of latent variables in this formulation. A Markovian HVAE is a HVAE where each transition down the hierarchy is Markovian.

A Variational Diffusion Model [1] is simply a Markovian Hierarchical Variational Autoencoder with three key restrictions :

- The latent dimension is exactly equal to the data dimension
- The structure of the latent encoder at each timestep is not learned; it is pre-defined as a linear Gaussian model. In other words, it is a Gaussian distribution centered around the output of the previous timestep.
- The Gaussian parameters of the latent encoders vary over time in such a way that the distribution of the latent at final timestep T is a standard Gaussian

We also investigate Variational diffusion models as conditional generative models. They can learn a probabilistic mapping from the conditioning input to the latent space, which allows them to capture complex dependencies between the input and the output distribution. A few important equations :

$$\mathbf{x_t} \sim \mathcal{N}(\mathbf{x_t}; \sqrt{\bar{\alpha}_t}\mathbf{x_0}, (1-\bar{\alpha}_t\mathbf{I}))$$

$$q(\mathbf{x_{t-1}}|\mathbf{x_t}, \mathbf{x_0}) \propto \mathcal{N}(\mathbf{x_{t-1}}; \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\mathbf{x_t} + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\mathbf{x_0}}{1-\bar{\alpha}_t}, \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I})$$

For a VDM, we can derive the following relation between the Evidence and the ELBO :

$$\log p(x) \geq \mathbb{E}_{q(x_1|x_0}[\log p_\theta(x_0|x_1)] - D_{KL}(q(x_T|x_0)||p(x_T)) - \sum_{t=2}^{T}[D_{KL}(q(x_{t-1}|x_t, x_0))||p_\theta(x_{t-1}|x_t)] \quad (1)$$

The first term in the above expression is the reconstruction term. The second term represents how close the distribution of the final modified input is to the standard Gaussian prior. The third term is a denoising matching term. We try to learn the denoising step as an approximation to a ground-truth denoising transition step by minimising the KL divergence between them.
The last term in the above equation is the diffusion loss. It can be simplified to

$$\mathcal{L}_T(\mathbf{x}) = \frac{T}{2}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbf{I}), t\sim U(1,T)}[(\text{SNR}(t-1) - \text{SNR}(t))||\mathbf{x} - \hat{\mathbf{x}}_\theta||_2^2]$$

Here, $\mathbf{z_t} = \alpha_t\mathbf{x} + \sigma_t\epsilon$, and the Signal to Noise Ratio is given by

$$\text{SNR}(t) = \frac{\bar{\alpha}_t}{1-\bar{\alpha}_t}$$

4.2 Outline of the method :

For this part, we mainly followed the reference implementation. We used the CIFAR10 dataset which consists of 10 classes and 5,000 images from each class.
Preparing the dataset : Used CIFAR-10 from tensorflow datasets. Preprocessing included transposing the image, converting the image pixels to float, and moving some images into the unknown category. So effectively, we have 11 classes. This is useful while conditioning.
The forward process : We used a variance preserving map in the forward diffusion process. We could use a neural network to efficiently learn γ_t, the signal to noise ratio, but, in this implementation, we have constrained the signal to noise ratio to follow a linear behaviour.
Use of time embeddings : Diffusion process can be slow, especially when dealing with high-dimensional data. To speed up the diffusion process, time embeddings can be used to introduce a time-varying feature that encodes the number of diffusion steps that have been performed. Fourier feature inspired embeddings have been used in the implementation.
Architecture : Latent diffusion has been performed in the implementation. Hence, an Encoder to encode the image into its latent representation, and a decoder to get the original image back from the latent representation have been used. Both of them use a ResNet Block. We used a ScoreNet to predict the noise in a noisy image. This is a very important component when calculating the loss as the diffusion loss depends upon this predicted noise.
Loss calculation : There are three loss terms involved in this implementation, the Reconstruction loss, the latent loss, and finally, the diffusion loss. These have the same meaning as in the first equation mentioned above

Conditioning : When conditioning is not set to None, we pass a conditioning vector at each stage of encoding and decoding. In this case, for the CIFAR10 dataset, we used the labels of images as part of the conditioning. When generating images, we can pass a vector of labels to condition the generation.

4.3 Experiment details and main results:

We experimented with different gamma schedules and different number of time steps. Apparently, the number of timesteps in the diffusion process doesn't have much effect on the results. However, it is important to note that the authors in the original paper have mentioned that the number of parameter updates for their state-of-the art generation is close to 2 million. Supposing that the number of updates is the same as the number of steps, though we couldn't perform as many updates, we tried upto 50,000 steps. Some of the generated samples are as follows :

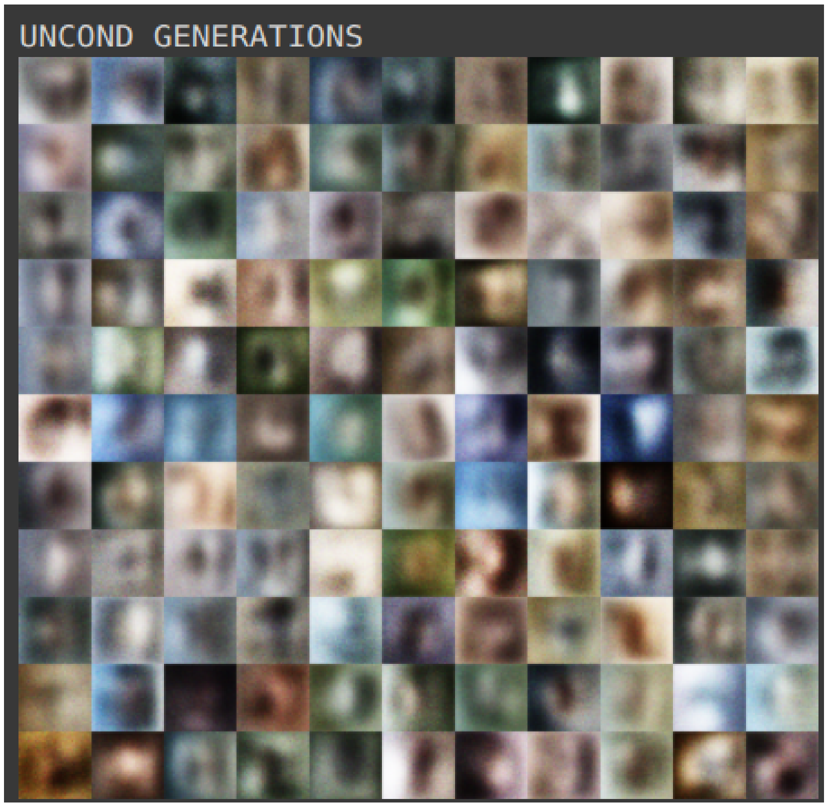

Figure 3: 500 steps

As we can see from this, and the below examples, that, as the number of steps increases, loss decreases and quality tends to increase.

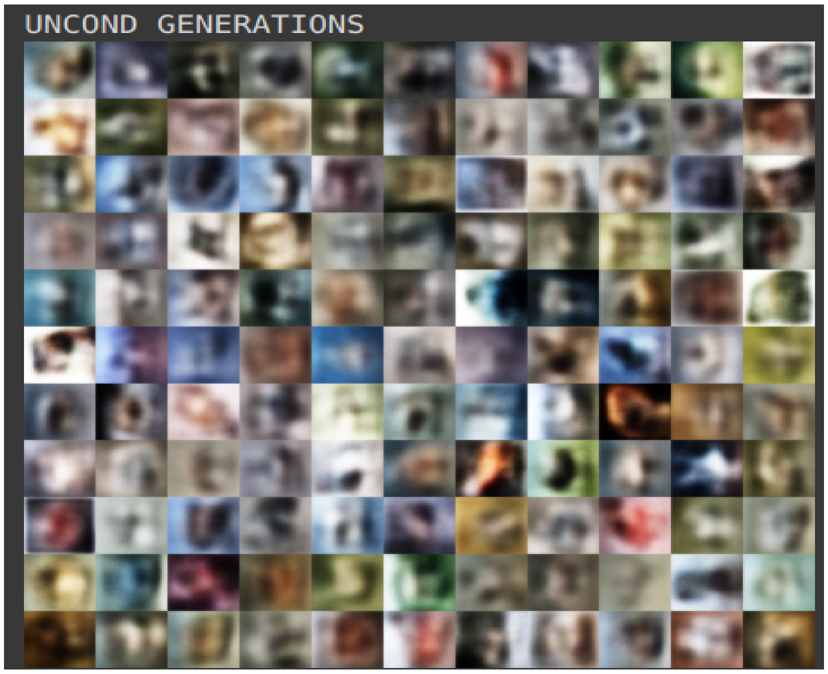

Figure 4: 2000 steps

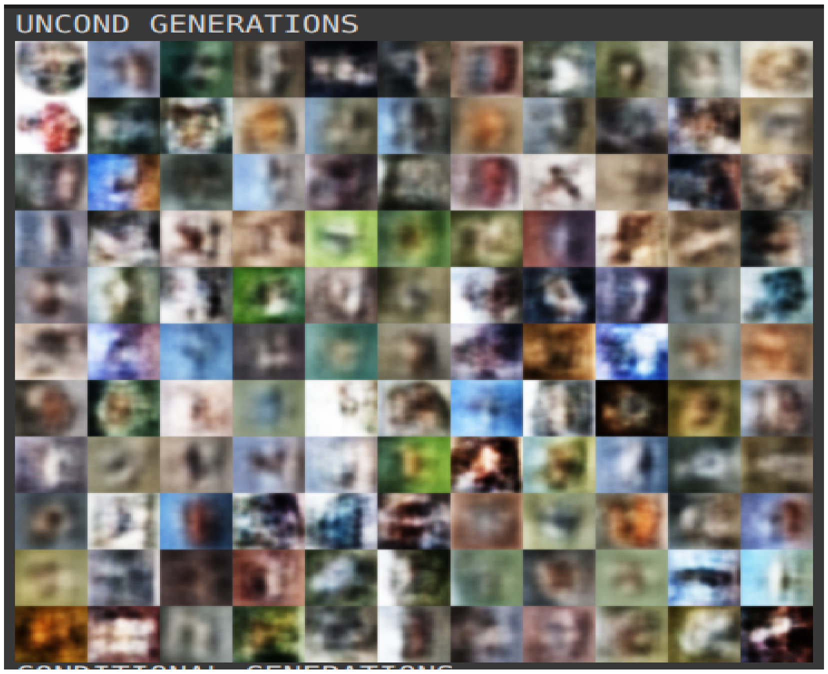

Figure 5: 50000 steps

4.4 An observation

: When we didn't turn 'off' any pixels from the data before training in the preprocessing step, we observed that the loss became very low (almost 20 times lower) but the samples generated were of the form shown below. Upon inspecting the pixel values, we found that most of the pixels had very low values(of the order 10^{-7}).

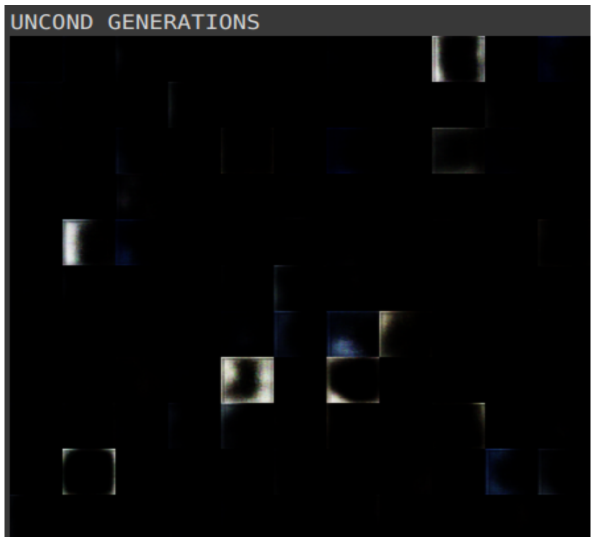

Figure 6: Samples obtained using a different preprocessing technique

4.5 Loss comparison

:

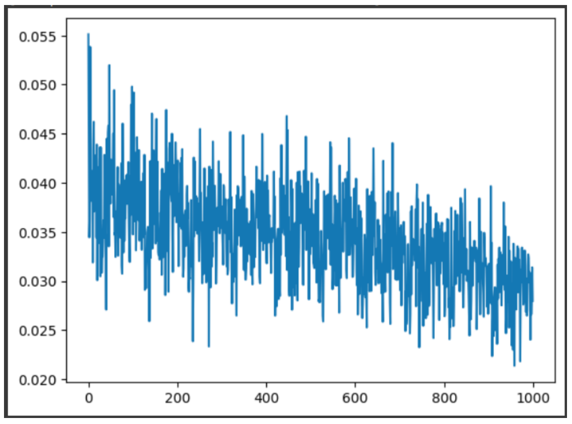

Figure 7: Loss for the above technique

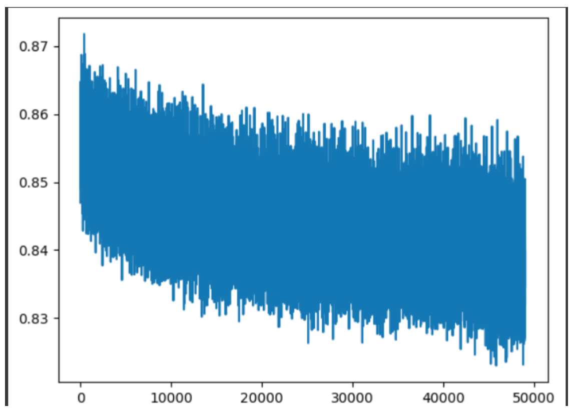

Figure 8: Loss for standard preprocessing

5 DDPMs + GANs

5.1 Outline of the method :

Normal Diffusion Process, the number of denoising steps T is often assumed to be in the order of hundreds to thousands of steps

The true denoising distribution $q(x_{t-1}|x_t)$ can be written as $q(x_{t-1}|x_t) \propto q(x_t|x_{t-1})q(x_{t-1})$ using Bayes' rule where $q(x_t|x_{t-1})$ is the forward Gaussian diffusion shown and $q(x_{t-1})$ is the marginal data distribution at step t. It can be shown that in two situations the true denoising distribution takes a Gaussian form. First, in the limit of infinitesimal step size β_t, the product in the Bayes' rule is dominated by $q(x_t|x_{t-1})$ and the reversal of the diffusion process takes an identical functional form as the forward process . Thus, when β_t is sufficiently small, since $q(x_t|x_{t-1})$ is a Gaussian, the denoising distribution $q(x_{t-1}|x_t)$ is also Gaussian, and the approximation used by current diffusion models can be accurate.

In this method, Our goal is to reduce the number of denoising diffusion steps T required in the reverse process of diffusion models. We propose to model the denoising distribution with an expressive complex distribution. Since conditional **GANs** have been shown to model complex conditional distributions in the image domain we use them to approximate the true denoising distribution $q(x_{t1}|x_t)$. [2]

5.2 Outline of the training, sampling methods

Similiar to GANS we have a **Generator** to generate fake samples, and a **Discriminator** to give the probability that its fake/real.

TRAINING PROCESS

We sample some time **t** from **Uniform[1,2,3,.....,T]** . And we take a true image from $q(x_0)$. We apply the **forward** diffusion process on this by conditioning on this image just like the usual diffusion Model, and obtain x_{t-1} and x_t. No gradient copmputations are needed in this part as the output is just a deterministic function of x_0 , once we fix the noise schedules.

Then we train the Generator to predict x_0 given x_t. So the obtained x_t is used to get an x'_0 from Genrerator $\mathbf{G(x_t, t, z)}$, where z is some latent input, randomly choosen from N(0,I).

Then from the obtained x'_0, we apply the forward diffusion process to get x'_{t-1}. Now the discriminators job is to tell if x_{t-1} is real (obtained from frwd diffusion) OR fake (x'_{t-1} obtained by Generator + frwd diffusion). $\mathbf{D(x_t, x_{t-1}, t)}$.

We used the usual Binary Cross entropy kind of loss in the trained, but the training needs to be done in 2 steps, first gradient propagating through Discriminator, then generator.

loss = -log(prob_fake) - log(1-prob_real) .... for Discrminator. loss = -log(1-prob_fake) .... for Generator.

5.3 Experiment details and Main results :

Model Architecture Used.

for **Generator**, we used the UNET Architecture same as the one used in DDPM. except that we need to include a latent input as well. We took **latent_dim = 128** throughout our experiments.

for **Discrminator**, we just used tht downsampling path of the Unet architecture, and a AveragePooling layer , then a sigmoid to emit the probability. in [0,1]

The main adavntage of this method is , now we just need a very small number of steps like T = 4 , 8 etc. And thus the sampling is super fast compared to normal DDPM.

We tried to train the model with T = 8, and other parameets choosen similiar to described in the paper.

Our training was a bit unstable, maybe because of the log loss. But Eventually after some tweaking of learning rate etc, we the loss decreased very substantially,A But the samples it generated were just black images.

We are working on fixing this, by changing model architecture, and also we are trying to add a regulariser to the loss.

6 Related Work : Inference of diffusion models via shortcut MCMC sampling

6.1 Outline

It is time consuming to generate high quality images, which may take thousands of steps using DDPM. We can use MCMC shortcut method [**?**] to speed up the inference of diffusion models. Instead of thousands of steps to produce samples, we constrain the number of inference steps, which can be randomly sampled from these thousand steps (we call shortcut MCMC) and then generate images to match the data. Both denoising diffusion probabilistic models (DDPMs) and variational diffusion models (VDMs) train similar denoising deep nets, which focus on local model characteristics and thus long sampling steps needed to produce high quality images. Compared to VDMs, we used the shortcut MCMC sampling along with the fidelity term in the loss function so that the final synthesized images match the original data. This new fidelity term is more like a global constraint and quality control mechanism while generating images in a shortcut manner. Thus, our method can balance the training and inference stages, and mitigates the inference burden significantly.

On maximizing the $\log(p(x_0))$ term we get the loss function (variational upper bound) involving some KL-distance terms and fidelity term. g. To speed up the inference, we can skip steps to produce data while using MCMC sampling. Specifically, we random sample K time steps $t_1, .., t_K$ from $[1, T]$. Then we use the prediction $\hat{x}_{t_k}$ to get the next sample $\hat{x}_{t_{k-1}}$ according to the equation above. Thus we have the fidelity loss $E_q(\log p(x_0|z)) = ||x_0 - \hat{x}_0||^2$ where $\hat{x}_0$ is predicted from the shortcut MCMC sampling. By minimizing this loss, we add the global constraint to the deep denoise models, and further improve the data approximation quality.

The following steps are incorporated into our algorithm.

```
for k = K to 1 do
    predict $\hat{x}_0 = (x_{t_k} - \sigma_t \hat{k}_\theta(x_{t_k}, t))/\alpha_{t_k}$
    update $x_{t_{k-1}} \alpha_{t_{k-1}} \hat{x}_0 + \sigma_{t_{k-1}}$
end for

take gradient step to minimize $||x_0 - \hat{x}_0||^2$
```

In the inference stage, we just sample $\epsilon \sim N(0, I)$, then we sample K time steps from [1, T] and sample $x_{t_k} \sim \alpha_{t_k}\hat{x}_0 + \sigma_{t_k}$, where $\hat{x}_0$ is predicted from the denoise neural network in the previous t_{k1}. Thus, our method has the potential to speed up inference at least an order of magnitude fast.

6.2 Results

DATA : swirl dataset with 1024 points.

- The paper used $T = 200$ in the inference stage, and $K = 10$ to sample 10 time steps. Then we compared the corresponding generated images between VDMs and our method. Images generated by our method converged faster.
- Then, $T = 200$ was used in the inference stage, and $K = 200$ for the full time steps comparison. We can observe that the method can generate very good samples and converge fast then VDMs.
- These experiments showed us promising results on both data quality and fast inference time.

7 Conclusion :

- Though Variational Diffusion Models are designed to generate high quality samples, the number of steps should be large for achieving the desired quality
- Our training in GANs was a bit unstable maybe because of the log loss. Eventually, after some tweaking, we could decrease the loss significantly, but the sampled images were just black.
- We are working on fixing this, by changing the model architecture and maybe by adding a regulariser to the loss

8 Work split up among team members :

Bhuvan and Suman worked on GANs. Sainath and Praneeth worked on VAEs. Initially, Praneeth and Suman also thought of exploring the MCMC shortcut technique, but couldn't due to limited time available.

References

[1] D. P. Kingma, T. Salimans, B. Poole, and J. Ho. Variational diffusion models, 2023.

[2] Z. Xiao, K. Kreis, and A. Vahdat. Tackling the generative learning trilemma with denoising diffusion gans, 2022.