Image Generation using Diffusion Models

Aditya Kumar
Electrical Engineering
IIT Bombay
Roll 210070003

Azeem Motiwala
Electrical Engineering
IIT Bombay
Roll 210070018

Jay Chaudhary
Electrical Engineering
IIT Bombay
Roll 210070022

***Abstract*—This report delves into the exploration and analysis of diffusion models, a prominent framework within the field of mathematical modeling and dynamic systems. The concept of diffusion models revolves around the understanding of how entities, be they information, innovations, or behaviors, spread through a population over time. The report investigates three distinct variants of diffusion models: unconditional diffusion, conditional diffusion, and stable diffusion. Unconditional Diffusion models generate samples in an unconditional setting without any supervision, it is a type of unsupervised learning and is the most basic setting for image generation using diffusion models. Conditional Diffusion models go a step further and generate images based on a condition like a textual prompt or any other inference. Lastly, we explored Stable Diffusion. Stable Diffusion is a type of hidden diffusion process that simplifies images into a hidden area called the latent zone. This zone is much smaller than the original image's size. It aids in enhancing performance through the addition of an autoencoder, which supports extensive training.**

I. Introduction

Diffusion Models are generative models, meaning that they are used to generate data similar to the data on which they are trained on. Fundamentally, Diffusion Models work by destroying training data through the successive addition of Gaussian noise through numerous time steps. Typically, messing with data in Diffusion Models focused on changing the amount of noise added. Still, this report brings up a new way, stepping away from the normal routine. In this fresh process, we change not only the noise level during the noisy stage but also carefully tweak the variance of the noise in each time step.

By changing both mean and variance of noise at the same time, this suggested idea aims to give Diffusion Models a more flexible and rich structure.

II. Noising Process

The noised image after each time step is given by:

$$x(t + \Delta t) = x(t) + \sigma(t)\sqrt{\Delta t} r \tag{1}$$

Here

- Δ t : is the time step
- x(t) : is the image after t time steps
- x(t+Δ t) : is the noised image after time steps t+Δt
- $\sigma(t) > 0$ is the noise strength
- r $\sim \mathcal{N}(0, 1)$ is a standard normal random variable

After noising the image for a given number of time steps, until the image is almost completely noise. After this we try to recover the image through a denoising process. In the reverse process, the neural network predicts the mean given the image. The neural network will look at the image and try to determine the distribution of the images where that image came from in the forward process.

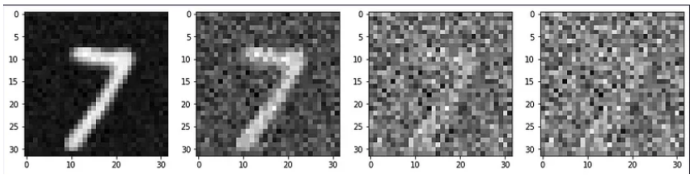

Fig. 1. Noising process

III. Denoising

The update equation in the denoising process is given by:

$$x(t+\Delta t) = x(t)+\sigma(T-t)^2 \frac{d}{dx} \left[\log p(x, T-t)\right] \Delta t+\sigma(T-t)\sqrt{\Delta t}\, r \tag{2}$$

Here

- Δ t : is the time step
- x(t) : is the image after t time steps
- x(t+Δ t) : is the denoised image after time steps t+Δt
- $\sigma(t) > 0$ is the noise strength
- s(x, t) is the score function which is give as := $\frac{d}{dx} \log p(x, t)$
- r $\sim \mathcal{N}(0, 1)$ is a standard normal random variable

Fig. 2. Denoising process

After training, we can use the Diffusion Model to generate data by simply passing randomly sampled noise through the learned denoising process. If our initial sample is always just one point at $x_0 = 0$, and the noise strength is constant, then the score function is exactly equal to

$$s(x, t) = -\frac{(x - x_0)}{\sigma^2 t} = -\frac{x}{\sigma^2 t} \ . \tag{3}$$

In practice, we don't already know the score function; instead, we have to learn it. One way to learn it is to train

a neural network to 'denoise' samples via the denoising objective

$$J := \mathbb{E}_{t\in(0,T),x_0\sim p_0(x_0)}\left[\ \|s(x_{noised},t)\sigma^2(t)+(x_{noised}-x_0)\|_2^2\ \right] \tag{4}$$

where $p_0(x_0)$ is our target distribution (e.g. pictures of cats and dogs), and where x_{noised} is the target distribution sample x_0 after one forward diffusion step, i.e. $x_{noised} - x_0$ is just a normally-distributed random variable. We finally obtain the objective function as :

$$J := \mathbb{E}_{t\in(0,T),x_0\sim p_0(x_0),\epsilon\sim\mathcal{N}(0,I)}\left[\ \|s(x_0+\sigma(t)\epsilon,t)\sigma(t)+\epsilon\|_2^2\ \right] \tag{5}$$

For Stable Diffusion we use an autoencoder to compress the input images to a latent dimension which is much smaller than the original input dimension and then obtain an image from the compressed sample.This has a few advantages. An obvious one is speed: compressing images before doing forward/reverse diffusion on them makes both generation and training faster. Another advantage is that the latent space, if carefully chosen, may be a more natural or interpretable space for working with images. For example, given a set of pictures of heads, perhaps some latent direction corresponds to head direction.

If we do not have any a priori bias towards one latent space or another, we can just throw an autoencoder at the problem and hope it comes up with something appropriate.

IV. Our Work and Results

The basic architecture of our Denoising Diffusion Probabilistic Model comprises of a basic input noising block which adds noise to an input image based on equation (1) after noising the noised image is passed through a U-Net architecture which uses a Gaussian Fourier Projection block, the primary objective is to embed time-related information into the network's representation, facilitating the modeling of dynamic phenomena. After the U-Net we have used a sampler to sample or generate images from random noise as input.
We have performed training and generation of images on 3 datasets :

- MNIST
- Fashion MNIST
- CIFAR10

We have performed unconditional and conditional genertaion on all the three datasets the basic architecture for all the three models remains the same except for the complexity of the U-Net used, for the FashionMNIST and CIFAR10 models the complexity of the respective U-Nets is much higher as compared to that of the MNIST U-Net because the images in the latter two datasets contain many more features as compared to MNIST.

V. Unconditional DDPM

We obtain the following results for unconditional DDPM model, in all our models we have set the number of time steps as 500 and the time step as .We have set the LR scheduler in all the unconditional models and used Adam as optimizer, we have set the learning rate after extensive hyper parameter tuning based on the output of the model using different learning rates.
Below are the generated images for the model trained on Fashion MNIST.

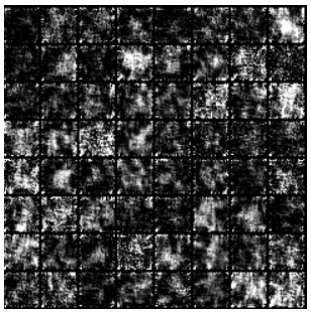

Fig. 3. 1st Epoch

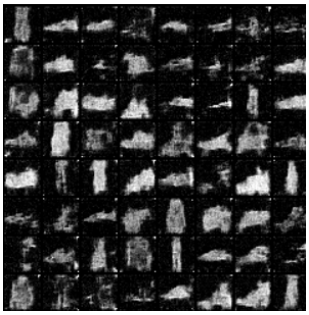

Fig. 4. 10th Epoch

Fig. 5. 25th Epoch

Fig. 6. Final Epoch

Fig. 7. Output Images

The loss per epoch is plotted below for MNIST and CIFAR10 dataset, the bottom two images shows the images generated after training on CIFAR10 :

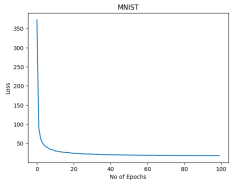

Fig. 8. Loss per epoch for MNIST

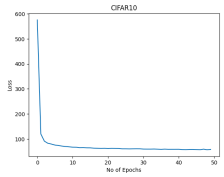

Fig. 9. Loss per epoch for CIFAR10

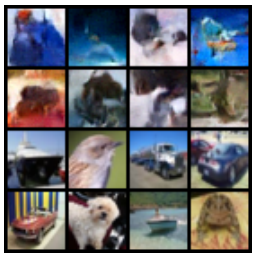

Fig. 10. DDPM output for CIFAR after 43rd epoch

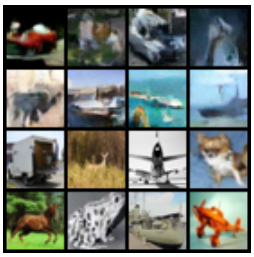

Fig. 11. DDPM output for CIFAR after 100th epoch

Fig. 12. Unconditional DDPM Results

In the last 2 images the bottom 8 are the ground truth images whereas the top 8 are generated images.

VI. Conditional DDPM

The results for the conditional DDPM are illustrated below. For conditional DDPM we made changes in the U-Net architecture by introducing attention in the conv layers. We change the scheduler and optimizer for training on CIFAR and FashionMNIST after trying out many different combinations of optimimzers we decided to go with AdamW as optimimzer as it gave the minimum loss and the best outputs after trainig.

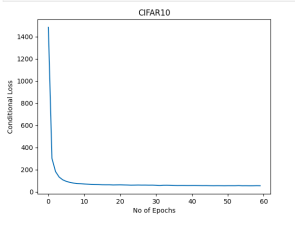

Fig. 13. Conditional Loss per epoch for CIFAR10

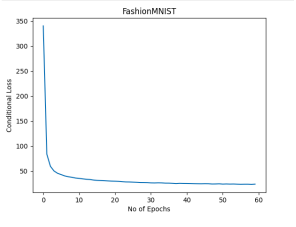

Fig. 14. Conditional Loss per epoch for FashionMNIST

Fig. 15. Number 9 generated using Conditional DDPM

Fig. 16. Number 2 generated using Conditional DDPM

Fig. 17. Conditional DDPM Results

We tried out various schedulers for conditional training on CIFAR dataset but the loss exploded repeatedly after a certain number of epochs(approximately 40) on each one of the schedulers. So we used cosine annealing initially till the loss decreased monotonically and then we used constant learning rate for rest of the remaining epochs.

VII. Stable Diffusion

For stable diffusion we compress the input images using an autoencoder which was initially trained on image degradation and restoration task based on a bottlneck neural network architecture, on the FashionMNIST dataset. After training the autoencoder we use it to compress the input images and then perform noising and denoising on the compressed images.

Now, instead of dealing with the whole detailed image, we let this compressor crunch it down into a much simpler form, like creating a summary. This simpler version is about 10 times smaller than the original picture. Then, we mess around with this compressed summary for stable diffusion.

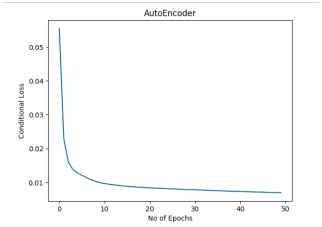

Fig. 18. Loss per Epoch for AutoEncoder

Fig. 19. Original Image passed into the Autoencoder

Fig. 20. Encoded Image

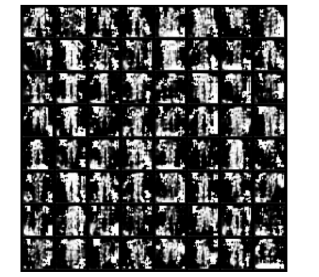

Fig. 21. Final Output using Stable Diffusion to generate Trouser Images

Fig. 22. Stable Diffusion Results

This helps our model work super fast because it's dealing with a much smaller chunk of info. It's like telling a story with fewer words but still capturing the important stuff. So, not only does it speed up training, but it also makes our model more efficient and zippy. It's like making things run smoother with less work

The above four images illustrate the loss per epoch for autoencoder and the encoded image given an input image, at first it seems there is no difference in between the two images but if we zoom in further we see there is a difference in the pixel levels i.e. the encoded images are more pixelated and have lower quality as compared to the input images. We obtain the final outptut of stable diffusion for trousers, the generated images are of a lower quality than of the previous model as expected because of the use of compressed images as input into the model

VIII. Conclusion

In summary, our exploration of Diffusion Models represents a departure from the conventional noise manipulation techniques. Traditionally, these models generate data by tweaking the amount of Gaussian noise added. Our report introduces a new method, altering both the mean and variance of the noise simultaneously at each time step.

This unconventional approach aims to provide Diffusion Models with increased flexibility and a more refined structure. By embracing this idea, we seek to enhance the model's adaptability and its capacity to capture intricate patterns in the data.

The core architecture of our Denoising Diffusion Probabilistic Model comprises a basic input noising block, a U-Net structure incorporating Gaussian Fourier Projection for temporal information, and a sampler for image generation from random noise. We conducted training and generation experiments on three datasets: MNIST, Fashion MNIST, and CIFAR10. Both unconditional and conditional generation tasks were performed, demonstrating the versatility of our model.

While maintaining a consistent foundational architecture, we adjusted the complexity of the U-Net based on the richness of features in each dataset. The Fashion MNIST and CIFAR10 models feature more intricate U-Nets to accommodate the complexity of images within these datasets.

In essence, our report introduces a novel approach to Diffusion Models and highlights the adaptability of our model across diverse datasets. The simultaneous manipulation of mean and variance during the diffusion process opens up avenues for future research and advancements in generative modeling. Looking forward, this fusion of innovation and adaptability promises to refine our understanding and application of Diffusion Models in generative artificial intelligence.

IX. Future Work

We plan to expand this model to generate high quality images by training on large datasets such as CelebA, we tried to perform training on CelebA using the models with same complexity as of CIFAR10 but as expected the model didnot generate anything meaningful. The major reason behind this could be because of the large size of the dataset and many high level features of the images the same model couldnot capture the complexity of the dataset and the images. For training on a large dataset such as CelebA we need to increase the depth of the U-Net and increase the embedding dimension. Following is the output we obtained while trainig on CelebA:

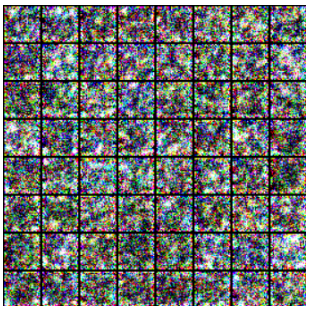

Fig. 23. Image 1

Fig. 24. Image 2

Fig. 25. Combined Results

References

- Denoising Diffusion Probabilistic Models: https://arxiv.org/abs/2006.11239
- Stable Diffusion: https://browse.arxiv.org/pdf/2112.10752.pdf
- A friendly Introduction to Denoising Diffusion Probabilistic Models: https://medium.com/@gitau$_a m/a - friendly - introduction - to - denoising - diffusion - probabilistic - models - cc76b8abef25$
- High-Resolution Image Synthesis with Latent Diffusion Models: https://arxiv.org/abs/2112.10752
- Annonated diffusion U-Net: https://nn.labml.ai/diffusion/ddpm/unet.html
- Weight initialization: https://365datascience.com/tutorials/machine-learningtutorials/what-is-xavier-initialization/: :text=For
- LearningRateScheduler: https://towardsdatascience.com/a-visual-guide-tolearning-rate-schedulers-in-pytorch-24bbb262c863