PROBABILISTIC GENERATIVE MODELING

Achintya Gupta, *210101005*, *IIT Guwahati*

Kannan Rustagi, *210101054*, *IIT Guwahati*

Kshitij Maurya, *210101059*, *IIT Guwahati*

Parth Kasture, *210101074*, *IIT Guwahati*

Shashank Kumar, *210101097*, *IIT Guwahati*

Abstract—**Diffusion models and GANs each excel in generative tasks but face limitations: diffusion models achieve high quality and diversity through costly iterative denoising, while GANs produce details quickly but suffer from mode collapse. This paper presents a hybrid model that combines these strengths, using diffusion models for initial coarse generation and GANs for fine detailing. By reducing diffusion steps and relying on GANs for high-frequency details, this approach improves computational efficiency while preserving quality and diversity. We explore separate and joint training methods to harmonize both models, aiming to enable fast, high-resolution generation for real-time applications.**

I. Introduction

Probabilistic generative models are a type of machine learning model that learn to model the underlying process of how data is generated. Instead of directly mapping inputs to outputs, they learn a joint probability distribution over both inputs and outputs. This allows them to understand the relationships between variables, quantify uncertainty, and even generate new data. In the realm of image generation, these models learn a probabilistic distribution over the space of possible images, grasping the patterns, textures, and styles that define images. By sampling from this distribution, the model can generate new, realistic images that share the same characteristics as the training data. This process involves training on a large dataset, learning a distribution, sampling from it, and outputting a new image.

Probabilistic generative models are built on assumptions about the data structure. They work by explicitly or implicitly estimating the probability density of the data. Explicit models define and optimize a probability function directly over data, as in Variational Autoencoders (VAEs) and Gaussian Mixture Models. Implicit models like Generative Adversarial Networks (GANs), on the other hand, do not directly define a probability function but rather learn to generate data samples by training a generator network to mimic the real data distribution in a way that can fool a discriminator network. Regardless of the approach, probabilistic generative models rely on latent variables or noise as a starting point to model the underlying distribution of complex datasets and capture various patterns or structures within the data.

A key advantage of probabilistic generative models is their flexibility in handling different data types and complex distributions, which is invaluable for unsupervised and semi-supervised learning. These models enable unsupervised learning tasks by finding patterns or hidden structures in unlabelled data, which can also improve performance in semi-supervised scenarios where labeled data is limited. For example, VAEs and GANs can create entirely new images based on real samples by discovering the latent factors of variation in the data, such as lighting in images or speaker identity in audio samples, and use these factors to generate realistic variations. This makes probabilistic generative models highly versatile tools in machine learning and artificial intelligence, powering applications from creating art and editing images to generating synthetic data for model training.

II. Literature Review

In [1] Ian et al. introduces Generative Adversarial Networks, a groundbreaking framework for generative models that uses a game-theoretic approach. GANs consist of two neural networks: a generator, which creates realistic data samples, and a discriminator, which evaluates the authenticity of the generated samples compared to real data. This adversarial process, where the generator tries to fool the discriminator, leads to increasingly realistic data generation. GANs avoid the need for Markov Chains, simplifying the training process compared to older methods like Restricted Boltzmann Machines and autoencoders. The success of GANs sparked widespread interest, especially due to their flexibility and capability to generate realistic images, audio, and text. However, GANs are known for their instability during training and issues like mode collapse, where the generator produces limited diversity. Despite these challenges, GANs laid the foundation for innovative generative models that balance quality and diversity in synthetic data.

In [2] Jonathan et al. present a new type of model for generating images called a diffusion probabilistic model. Unlike popular models like GANs, which can be difficult to train and sometimes produce unrealistic images, diffusion models work by adding small amounts of noise to images over many steps and then training a reverse process to remove this noise and recreate the images. This method has links to other techniques like score matching and Langevin dynamics, making it theoretically sound and practically effective. The model performs well on image quality metrics and even beats some GANs on datasets like CIFAR10. An interesting feature of diffusion models is that they can decode images gradually, making them useful for compressing images and creating smooth transformations between images. This paper shows that diffusion models could be a powerful alternative to GANs, with both high-quality results and stable training.

In [3] Jiaming et al. presents Denoising Diffusion Implicit Models as an improvement on Denoising Diffusion Probabilistic Models (DDPMs), focusing on reducing computational complexity. DDIMs leverage a deterministic, non-Markovian diffusion process, where data is generated by reversing a noising process applied to samples. By estimating the data distribution with fewer steps than in traditional diffusion models, DDIMs make sampling more efficient while maintaining high sample quality. Unlike GANs, which rely on adversarial training, DDIMs generate realistic data without this need, leading to more stable training. They are particularly advantageous in applications that prioritize sample diversity and quality, as well as in domains where training efficiency is crucial. The design of DDIMs thus addresses some limitations in both GANs and DDPMs by providing an approach that combines quality and computational efficiency, making them competitive for complex data generation tasks, such as high-resolution image synthesis.

In [4] Yand et al. advances generative modeling further by introducing Noise Conditional Score Networks, which generate samples by approximating gradients of the data distribution using score matching. NCSNs are designed to overcome two main challenges in score-based modeling: the instability caused by data residing on low-dimensional manifolds and difficulties in estimating scores in low-density regions of the data distribution. NCSNs apply Gaussian noise at various levels to the data to ensure the distribution spans the entire space, then train a score network to estimate gradients for all noise levels simultaneously. To generate samples, NCSNs use annealed Langevin dynamics, which gradually reduces noise levels, guiding the samples toward high-density areas of the data distribution. This process circumvents some of GANs' challenges, such as adversarial instability, and avoids the need for computationally expensive steps used in diffusion models. NCSNs are experimentally validated on standard datasets and achieve competitive scores compared to GANs, especially in image inpainting and high-resolution image generation.

In [5] Karsten et al. discusses a generative learning model that addresses the challenges of high sample quality, mode coverage, and fast sampling, known as the "generative learning trilemma." Traditional models like Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and Diffusion Models often struggle to achieve all three goals simultaneously. Diffusion models, known for their sample quality and diversity, suffer from slow sampling times. To address this, the authors introduce Denoising Diffusion GANs, a model that replaces the Gaussian assumptions in diffusion processes with a multimodal distribution, which allows for faster, high-quality sampling. This approach, which uses conditional GANs in each denoising step, results in competitive sample quality and diversity, while achieving significant speed improvements over existing diffusion models, making them more suitable for real-world applications.

III. Proposed Work

In the field of probabilistic generative modeling, diffusion models and Generative Adversarial Networks (GANs) have independently demonstrated remarkable success but are limited by specific trade-offs. Diffusion models excel in generating high-quality and diverse samples by sequentially refining noisy data into realistic outputs, albeit at a high computational cost due to the large number of iterative sampling steps required. GANs, on the other hand, can produce high-frequency details quickly in a single forward pass, yet they often suffer from issues like mode collapse and challenges in faithfully capturing data diversity. These limitations highlight the need for a novel approach that combines the strengths of diffusion models and GANs to overcome their individual weaknesses, enabling efficient, high-quality generation of realistic data samples.

This project addresses a fundamental limitation in existing generative modeling approaches and aims to establish a new framework that combines diffusion models and GANs to achieve rapid, high-quality sample generation. The proposed hybrid model has the potential to transform applications that rely on real-time, high-resolution image generation by offering a solution that balances quality, diversity, and computational efficiency.

IV. Experimental details

A. Dataset

The dataset used in this study consists of grayscale images of handwritten digits (0–9), organized in subdirectories corresponding to each digit label. The custom MnistDataset class loads these images from a specified root directory, where each image is stored in PNG format. The class preprocesses the images by converting them to PyTorch tensors and normalizing the pixel values to the range [-1, 1] for improved model training. The dataset is split into training and testing subsets, typically containing 60,000 training images and 10,000 test images, each with a resolution of 28x28 pixels. This dataset is designed for image classification tasks and is compatible with PyTorch's data-loading utilities for training deep learning models.

B. Training and Sampling

1) Using pretrained GAN : UNet Architecture is used for the diffusion model, which is a common choice for generative models like DDPM. The model is designed to reverse the noise process by predicting the noise added at each timestep of the diffusion process. A linear noise scheduler is used to gradually add noise to the images over several timesteps. The scheduler helps the model learn how to denoise images step-by-step.

The optimizer used is Adam with a learning rate of 0.0001.The model is trained for a specified number of 10 epochs and batch size is set to 64. The diffusion process runs for 100 timesteps, with gradually increasing beta values defining the noise schedule.

The model starts with random noise (from a standard Gaussian distribution). Using the trained UNet model, it iteratively denoises this random noise across the timesteps (from the last to the first), effectively generating an image step-by-step. At each timestep, the model predicts the noise at that step, and the image is progressively ”cleaned” until it becomes a realistic generated image (at timestep 0).

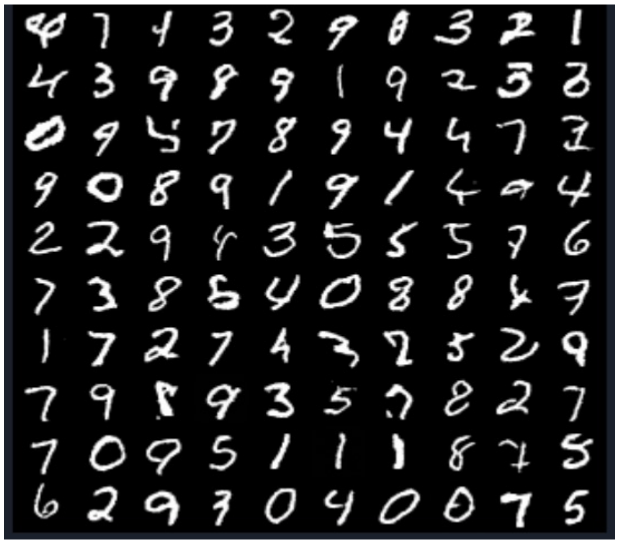
2) By training GAN: The discriminator is trained first by feeding it both real and generated images. It tries to predict whether the image is real or fake. For real images, the target label is 1 (real), and for fake images, the label is 0 (fake). Real Image Training: A batch of real images from the MNIST dataset is passed to the discriminator, and it learns to classify them as real (target label = 1). Fake Image Training: A batch of generated fake images (produced by the generator) is passed to the discriminator. These images are labeled as fake (target label = 0). The generator aims to trick the discriminator by improving the quality of the generated images over time. The loss for the discriminator is computed using binary cross-entropy loss (BCELoss), and the gradient is backpropagated to update the discriminator's parameters.

The generator is trained next. It produces a batch of fake images from random noise, and the discriminator evaluates them. The generator aims to fool the discriminator into classifying fake images as real (target label = 1). The generator's loss is computed based on how successful it is in this task, and the gradient is backpropagated to update the generator's parameters. The networks (generator and discriminator) use Adam optimizers with a learning rate of 0.0002. Adam helps in faster convergence by adapting the learning rate during training.

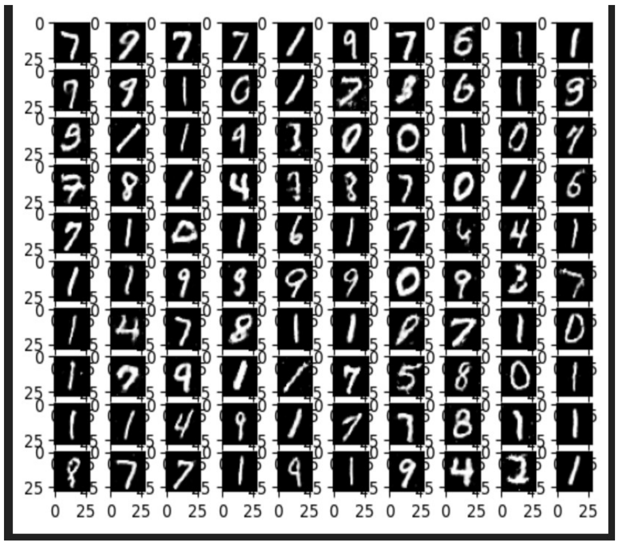
The model is trained for 100 epochs, with loss values being printed for both the discriminator and the generator at each epoch. Over time, the generator learns to produce images that are increasingly harder for the discriminator to distinguish from real images.

V. Results

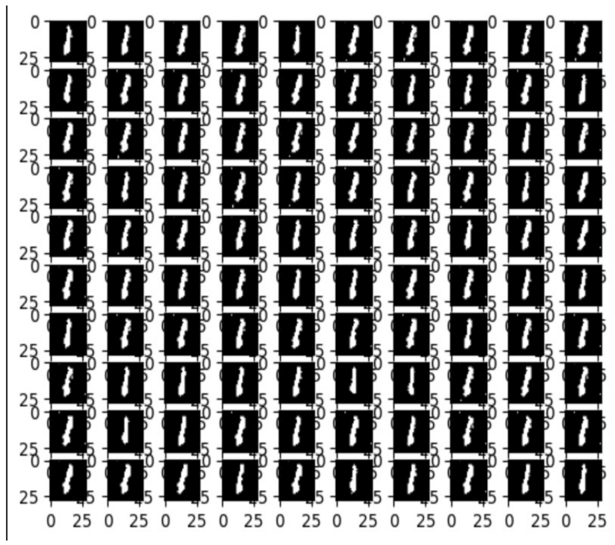
A. Image generated using SOTA implementation of ddpm



B. Image generated using SOTA implementation of GAN with noise as input

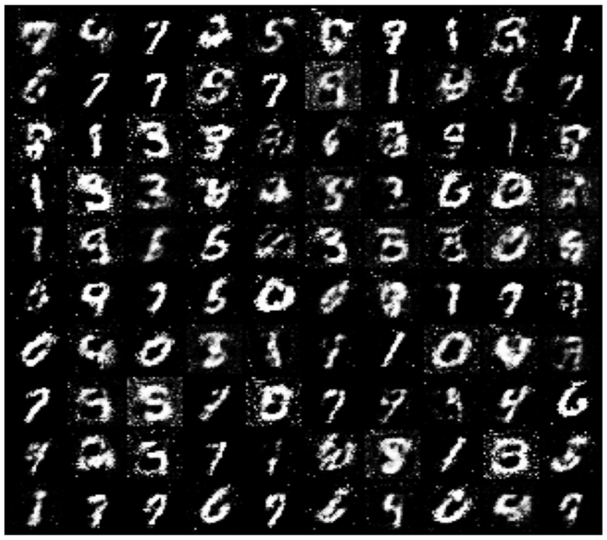


C. *Image generated using pretrained GAN and ddpm*



Since pretrained gan was not performing well as can be seen, we decided to train our own gan which did improve results a little bit.

D. *Image generated by feeding the output of DDPM to the GAN we trained*



VI. Conclusion

The initial implementation of DDPM was trained with the resources available to us for 50 epochs which took 153 minutes to complete and then to generate each image we used 1000 sampling iterations which were processed at the rate of 6 iterations per second, taking the total sampling time to be around 167 seconds per image generated.

Our implementation of the DDPM and GAN hybrid model involved training the DDPM for 10 epochs, which took about 30 minutes and then sampling noisy images for 100 iterations, taking around 17 seconds. This noisy image was then used as the input for GAN which generated a new image almost instantaneously. So we were able to reduce the time for image generation from 167 seconds to 17 seconds, although some amount of quality was compromised.

To improve quality, either we would need to train both the DDPM and GAN for a significantly larger number of epochs which is computationally infeasible for us at this level. Moreover, how this approach would fair on coloured images with multiple channels, as compared to MNIST with just a single channel, is also something that can be explored in the future.

References

[1] Generative Adversarial Networks, https://arxiv.org/abs/1406.2661

[2] Denoising Diffusion Probabilistic Models, https://arxiv.org/abs/2006.11239

[3] Denoising Diffusion Implicit Models, https://arxiv.org/abs/2010.02502

[4] Generative Modeling by Estimating Gradients of the Data Distributions, https://arxiv.org/abs/1907.05600

[5] Tackling the generative learning trilemma with Denoising Diffusion GANS, https://arxiv.org/pdf/2112.07804