Report on Denoising Diffusion Probabilistic Models by Jonathan Ho (2020)

Members :

- **Ajinkya Phanse (amp660)**
- **Jayant Chaudhari (jc3303)**

Problem Description :

What are Generative Models?

Generative models are a class of statistical and machine learning techniques that aim to model the probability distribution of a dataset. Unlike discriminative models, which learn to predict a specific output from given inputs, generative models are capable of generating new data instances that are similar to the training data. They do this by learning the underlying probability distribution of the input data. Generative models are widely used for tasks such as image and video generation, text generation, and more, where the goal is not just to classify input data but to create entirely new samples that could plausibly belong to the original dataset.

Functioning of Generative Models

The basic function of a generative model involves two main steps:

- Learning the Distribution: The model learns the intricate patterns and structures of the input data, capturing the underlying statistical distribution. For images, this might involve understanding textures, colors, and object relationships, while for text, it involves syntax, semantics, and contextual usage.
- Data Generation: Once trained, the model can sample from this learned distribution to produce new data instances that mimic the original data. This is typically achieved through various sampling techniques depending on the model's architecture.

Types of Generative Models and Their Challenges

Among the most popular types of generative models are Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs):

- GANs work through a competitive process where a generator model creates samples and a discriminator model evaluates them. This adversarial process can lead to high-quality generation but is often prone to training difficulties such as mode collapse, where the generator starts producing a limited variety of outputs.
- VAEs use a probabilistic approach to map input data to a latent distribution and then sample from this distribution to generate outputs. While generally easier to train than GANs, VAEs often result in outputs that are blurrier or less precise compared to the training data.

Denoising Diffusion Probabilistic Models (DDPMs)

DDPMs provide a robust mechanism for generating high-quality data, handling the shortcomings of earlier generative models by using a principled approach rooted in stochastic processes and variational inference. Their ability to model the gradual transition from noise to structured data via a learned reverse process opens up new possibilities in generative modeling, making them suitable for a wide range of applications, from image generation to speech synthesis:

- They start with a distribution of pure noise and gradually transform this into structured data through a series of learned reverse diffusion steps.
- This approach avoids the adversarial setup of GANs and provides a clearer training objective compared to VAEs, resulting in stable training dynamics and the generation of high-quality samples.

Mathematical Framework of DDPM's:

DDPMs are structured around a forward diffusion process that gradually adds noise to the data and a reverse process that learns to reconstruct the original data from the noise. Here's a detailed breakdown of the mathematical components:

1. **Forward Diffusion Process:**

   The forward process is defined as a Markov chain where each step incrementally adds Gaussian noise to the data:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

   - $\mathbf{x}_{t-1}$ is the data from the previous step, and $\mathbf{x}_t$ is the data at the current step.
   - β_t is a small variance term that dictates how much noise to add.
   - This Gaussian formula ensures that the data gradually becomes noisier, approaching a Gaussian distribution as t increases.

2. **Transition Distribution (Forward Process):**

The distribution of $\mathbf{x}_t$ given $\mathbf{x}_0$ under this noise-adding process is:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1 - \bar{\alpha}_t)\mathbf{I})$$

- $\bar{\alpha}_t$ represents the cumulative retention of the original data amidst the noise added over time.
- This expression explains how much of the original data $\mathbf{x}_0$ is preserved at each timestep, and how much is replaced by noise.

3. **Reverse Diffusion Process:**

The reverse process aims to reconstruct the original data from the noise:

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu(\mathbf{x}_t, t), \Sigma(\mathbf{x}_t, t))$$

- This reverse process is modeled as learning the reverse of the forward Gaussian noise process.
- Learning to predict earlier, less noisy states from more noisy states.

4. **Parameterization of the Reverse Process Mean:**

$$\mu(\mathbf{x}_t, t) = \frac{1}{\sqrt{1 - \beta_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right)$$

- ϵ_θ is the neural network predicting the noise that was added at step t.
- This formula is crucial for effectively 'denoising' the data at each step, gradually restoring the original data by reversing the noise added.

5. **Variance of the Reverse Process:**

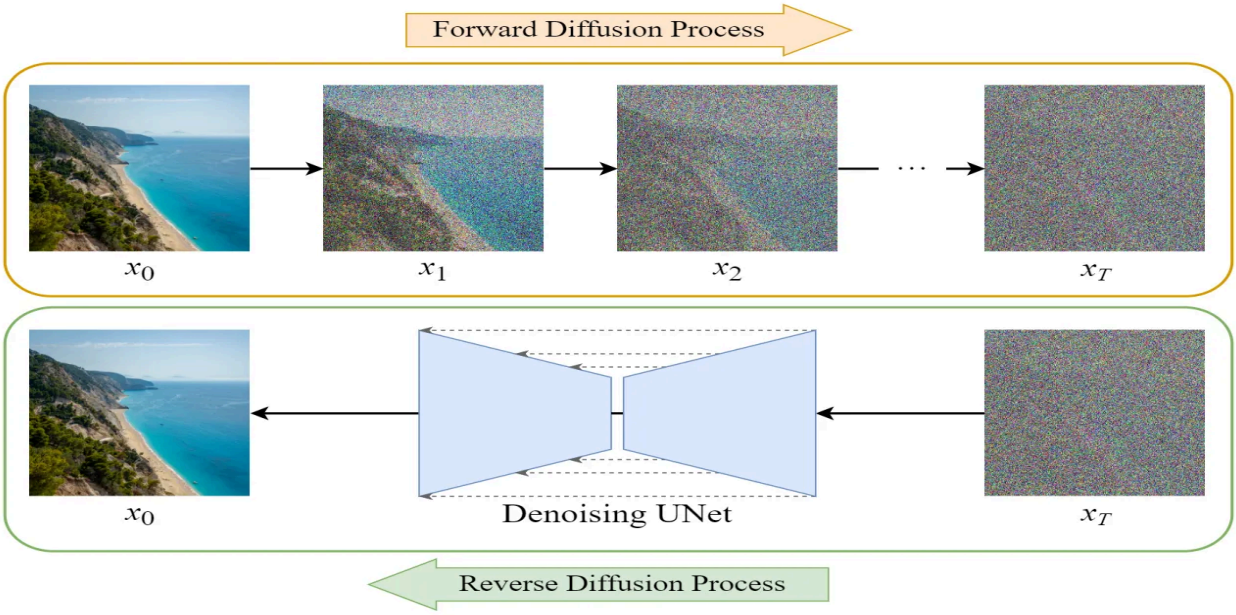
$$\Sigma(\mathbf{x}_t, t) = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t \mathbf{I}$$

- This component controls the uncertainty in the reverse process, essential for the model to explore various plausible denoising paths.

6. **Objective Function - Evidence Lower Bound (ELBO):**

$$\mathcal{L} = \mathbb{E}_q \left[\log \frac{p(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right]$$

- The ELBO maximizes the likelihood of reconstructing the original data from the noise.
- It encourages the model to accurately learn the reverse of the noise process, optimizing both parameterization of the noise prediction and the denoising sequence.



Models & Algorithms :

1. Model Formulation

Forward Diffusion Process:

- **Mathematical Model:** The forward diffusion is mathematically defined by a sequence of Gaussian transitions:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

  Each transition t incrementally adds noise via a variance parameter β_t, chosen to ensure gradual transition from data to noise.

- **Computational Steps:** In practice, the forward process can be computed in a single step using:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

  where $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ accumulates the effect of the noise across all previous steps.

Reverse Diffusion Process:

- **Parameterization:** The reverse process involves estimating the distribution of earlier states given later noisy states:

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu(\mathbf{x}_t, t), \Sigma(\mathbf{x}_t, t))$$

- **Mean and Variance Computations:**
  - Mean: $\mu(\mathbf{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right)$
  - Variance: $\Sigma(\mathbf{x}_t, t) = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t\mathbf{I}$

2. Training the Model

Training a model in the context of Denoising Diffusion Probabilistic Models (DDPMs) involves several distinct steps and concepts. The goal of training in DDPMs is to teach a model to reverse a noisy diffusion process by learning to progressively denoise data.

The Diffusion Process

Training a DDPM starts by defining the forward diffusion process, which is essentially a Markov chain that adds Gaussian noise to data over a series of time steps. This process transforms the original data x_0 into a sequence of increasingly noisy versions $x_1, x_2, \ldots, x_T$ until the data becomes indistinguishable from Gaussian noise. The transition from x_{t-1} to x_t is typically modeled as:

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_t,$$

where $\epsilon_t \sim \mathcal{N}(0, I)$ is Gaussian noise and β_t is a small positive value dictating the noise level at step t.

Parameterization of the Noise Schedule

The sequence $\beta_1, \beta_2, \ldots, \beta_T$ (known as the noise schedule) is crucial as it defines how quickly the data is corrupted by noise. This schedule is typically predefined and can be constant, linear, or learned based on experimental tuning to optimize performance.

Training Objective

The training objective in DDPMs is to learn the reverse of the diffusion process, essentially predicting either the noise that was added at each step or directly estimating the cleaner data from the noisier data. The common training objective is to minimize the mean squared error between the predicted noise and the actual noise used during the forward process:

$$L = \mathbb{E}_{t,x_0,\epsilon}\left[\|\hat{\epsilon}_t(x_t, t) - \epsilon_t\|^2\right],$$

where $\hat{\epsilon}_t(x_t, t)$ is the model's prediction for the noise, given the noisy data x_t and the timestep t.

Model Architecture

The architecture typically used for DDPMs is a U-Net, a type of convolutional neural network known for its efficacy in tasks that require understanding and manipulating spatial hierarchies, such as image segmentation and restoration. The U-Net architecture features:

- **Encoder-Decoder Structure**: With skip connections that help retain spatial hierarchies at multiple scales.
- **Conditional Inputs**: The model inputs not only include the noisy data x_t but also a representation of the timestep t, often encoded using positional encodings or learned embeddings to inform the model of the stage of diffusion.

Training Procedure

Training involves the following practical steps:

- **Batch Preparation**: In each training batch, select a random set of data points x_0 from the training dataset.
- **Noise Simulation**: For each x_0, simulate the forward diffusion process up to a randomly chosen timestep t, thereby generating x_t and recording the noise ϵ_t.
- **Model Prediction**: Feed x_t and t into the model to predict $\hat{\epsilon}_t$.
- **Loss Calculation and Update**: Compute the loss between $\hat{\epsilon}_t$ and ϵ_t and update the model parameters using backpropagation and an optimization algorithm like Adam.

3. Sampling :

Sampling in Denoising Diffusion Probabilistic Models (DDPMs) refers to the process of generating new data samples from the learned model. After training the model to reverse the diffusion process, sampling involves iteratively refining noise into a structured sample that resembles the training data.

Initial Setup

Sampling begins with initializing the first sample as pure noise:

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

Here, x_T represents the state of the data after the maximum number of diffusion steps, and is essentially random Gaussian noise.

Reverse Diffusion Steps

The core of the sampling process is to apply the learned reverse process iteratively from step T back to step 1:

$$x_{t-1} = \frac{1}{\sqrt{1-\beta_t}}(x_t - \sqrt{\beta_t}\hat{\epsilon}_t(x_t, t))$$

In this formula:

- x_t is the current noisy sample.
- $\hat{\epsilon}_t(x_t, t)$ is the noise predicted by the model, based on the current sample and timestep.
- β_t is the predefined noise level at timestep t, as used in the training phase.

Each iteration uses the model to estimate the noise that was added at timestep t, and then subtracts this estimated noise, appropriately scaled, from x_t. This gradually denoises the image, step by step, reconstructing the data from the noise.

Sampling Details

Step-by-Step Sampling

- **Starting from Noise**: The process begins at the last timestep T where the data is entirely noise.
- **Iterative Refinement**: For each timestep from T to 1, the model predicts the noise component $\hat{\epsilon}_t$ that was likely added at that step during the forward process. This noise is then removed to estimate the less noisy version of the data.
- **End Result**: At $t = 0$, the output x_0 is a generated data sample that mimics the distribution of the training data.

Model Predictions

- **Conditional on x_t and t**: The noise prediction by the model depends not only on the current state of the sample but also explicitly on the timestep, incorporating how far along in the reverse process the sample is.

Experimentation/Simulation :

Overview of MNIST Dataset in CSV Format

Dataset Files

Testing Data: mnist_test.csv

- Contains 10,000 test examples and labels.

Training Data: mnist_train.csv

- Contains 60,000 training examples and labels.

Data Structure

Each row represents one image.

Total columns per row: 785

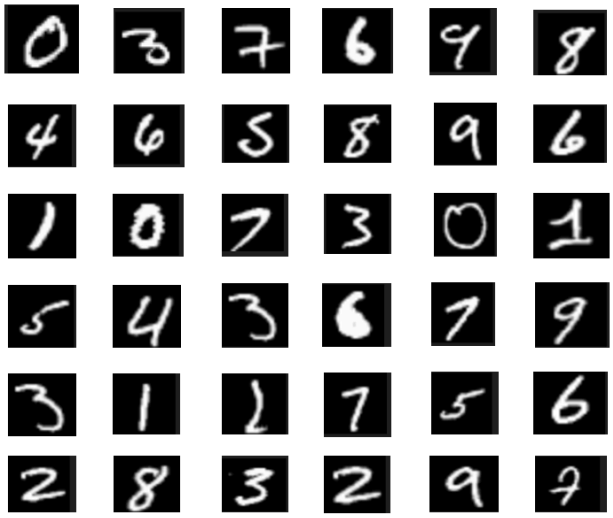
First Column: Label (digit from 0 to 9).
Remaining 784 Columns: Pixel values (ranging from 0 to 255).

Visualization

label	1x1	1x2	1x3	1x4	1x5	1x6	1x7	1x8	1x9	1x10	1x11	1x12	1x13	1x14	1x15	1x16	1x17	1x18	1x19	1x20	1x21	1x22	1x23	1x24
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Usage : Used for training image processing models to recognize handwritten digits.

Training Data in PNG Format



Noise Calculation in our code :

The loss calculation in the provided DDPM (Denoising Diffusion Probabilistic Model) training script involves a Mean Squared Error (MSE) criterion, which measures the difference between the predicted noise and the actual noise used to perturb the clean images at a specific timestep.

- Noise Sampling: For each image in the batch, random noise is generated that has the same shape as the image. This is done using `torch.randn_like(im)`.
- Timestep Sampling: A random timestep `t` is chosen for each image in the batch. This timestep determines how much noise will be added to the clean images to create the noisy images.
- Noise Addition: The clean images are perturbed with the noise based on the sampled timestep `t`. The specific amount of noise added at each timestep is determined by the noise schedule defined in `scheduler.add_noise(im, noise, t)`. The output here is the noisy version of the original image, referred to as `noisy_im`.
- Noise Prediction: The noisy images are then fed into the U-Net model along with the timestep `t`. The model attempts to predict the noise that was added to the clean image to generate `noisy_im`. The output, `noise_pred`, is the model's prediction of the noise.
- Loss Calculation: The MSE loss (`torch.nn.MSELoss()`) is computed between the predicted noise (`noise_pred`) and the actual noise (`noise`) that was added to the clean images. This loss measures the model's ability to accurately predict the noise, which is crucial for the reverse diffusion process used to reconstruct clean images from noisy ones during inference.
- Optimization: The loss is then backpropagated to update the model's weights via the optimizer (`Adam`), aiming to minimize this loss over training iterations.

Formula

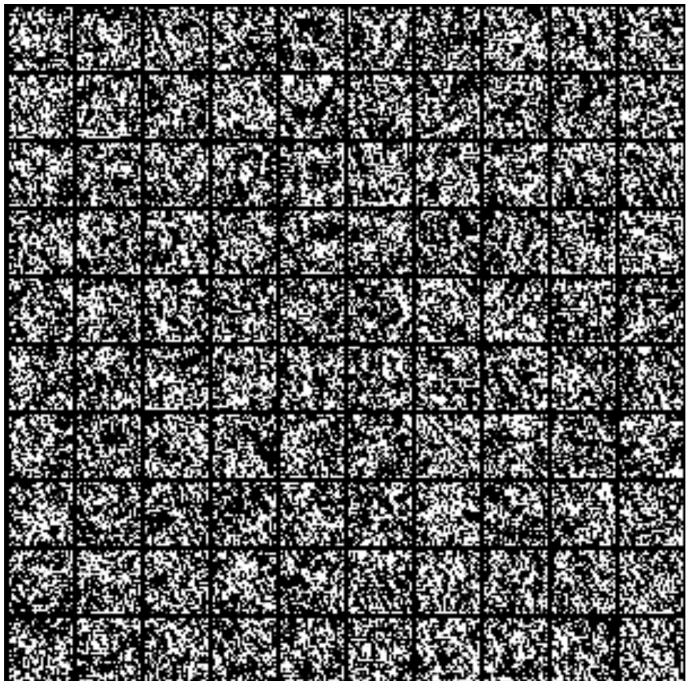
$$\text{MSE Loss} = \frac{1}{N}\sum_{i=1}^{N}(\text{noise\_pred}_i - \text{noise}_i)^2$$

- noise_pred is the noise predicted by the U-Net model (`model(noisy_im, t)` in the script). It is the model's estimation of the noise that was added to each pixel of the input images at the sampled timestep `t`.
- noise is the actual noise that was used to perturb the clean images, generated by `torch.randn_like(im)` in our script.
- N is the total number of pixels across all images in the batch, considering all dimension elements (e.g., width, height, and color channels if applicable).

Training with T=300(see below img)

Loss For each Epoch :

0 : 0.9691
1 : 0.9897
2 : 0.9965
3 : 0.9812
4 : 0.9989
5 : 0.9199
6 : 0.9883
7 : 0.9719
8 : 0.9924
9 : 0.9856



Training with T=10000(see below img)

Loss For each Epoch :
0 : 0.4128
1 : 0.3976
2 : 0.4484
3 : 0.3892
4 : 0.3901
5 : 0.4107
6 : 0.4261
7 : 0.4127
8 : 0.4992
9 : 0.3919

Training with T=30000(see below img)

Loss For each Epoch :

0 : 0.2193
1 : 0.2016
2 : 0.1958
3 : 0.2937
4 : 0.2597
5 : 0.1958
6 : 0.2068
7 : 0.2478
8 : 0.1939
9 : 0.2940

Training with T=50000(see below img)

Loss For each Epoch :

0 : 0.065
1 : 0.1003
2 : 0.0478
3 : 0.1090
4 : 0.1172
5 : 0.0465
6 : 0.060
7 : 0.051
8 : 0.1024
9 : 0.0453

Limitations of Denoising Diffusion Probabilistic Models Implementation :

- **Data Variety and Volume**: DDPMs often require large amounts of diverse data to learn effective representations. In scenarios where data is limited or lacks diversity, these models may struggle to generalize well, leading to poor performance.
- **High-Dimensional Data**: While DDPMs are adept at handling complex data distributions, they might face difficulties when dealing with extremely high-dimensional data or data with complex spatial or temporal dependencies. The model's performance can degrade due to the increased difficulty in accurately estimating the diffusion process across such high dimensions.
- **Noise Sensitivity**: The effectiveness of DDPM can be compromised in scenarios where the data is excessively noisy or the noise is not well-modeled by the assumptions of the diffusion process. The model might not be able to accurately reconstruct the clean data from its noisy version, particularly when the noise characteristics are not aligned with those the model expects based on its training.
- **Computational Resources**: DDPMs are computationally intensive due to the iterative nature of their sampling process. They require significant computational resources and time to generate samples, which might not be feasible in resource-constrained environments or for applications requiring real-time performance.
- **Stability Issues**: Training DDPMs can encounter stability issues, particularly when dealing with vanishing or exploding gradients. This can be exacerbated by inappropriate hyperparameter settings or if the model architecture is not well-suited to the complexity of the data.

Theoretical Guarantees :

- **Convergence to Data Distribution:**
  DDPMs are designed based on a Markov chain that progressively transforms data into Gaussian noise through a forward process and recovers data through a reverse process. Theoretically, if the reverse process accurately learns the noise distribution added at each step of the forward process, the model is guaranteed to converge to the original data distribution. This convergence is underpinned by the Fokker-Planck equation which describes the evolution of probability densities under diffusion processes. Thus, the reverse process in DDPMs theoretically ensures that the generated samples converge in distribution to the training data.

- **Stability of Training:**
  The training process in DDPMs involves minimizing the mean squared error between the actual noise added during the forward process and the noise predicted by the model during the reverse process. Theoretically, this objective ensures that the model learns a stable and accurate

approximation of the inverse of the forward diffusion, contributing to stable training dynamics, provided the noise levels (βt) are chosen appropriately.

- **Quality of Generated Samples:**
  DDPMs provide theoretical guarantees regarding the quality of generated samples based on the error bounds of the noise prediction during the reverse process. The smaller the error in predicting the added noise at each step, the closer the generated sample will be to the true data distribution. These models leverage the error decay across diffusion steps, which is critical for ensuring high-quality sample generation as it implies a high fidelity in reconstructing the original data from noise.

- **Continuous-Time Generalization:**
  In the continuous-time setting, DDPMs are related to score-based generative models through the Schrödinger-Föllmer process, providing a robust theoretical framework. This framework offers a seamless transition between discrete and continuous formulations, ensuring that DDPMs maintain theoretical soundness even when extended to continuous diffusion settings. This generalization helps in understanding and potentially improving various aspects like sampling efficiency and model robustness.

- **Robustness and Generalization:**
  Theoretically, DDPMs are expected to exhibit robustness to variations in input data and noise settings due to their inherent properties of learning the distribution dynamics via diffusion. This aspect theoretically supports the generalization capabilities of DDPMs across different data domains and noise conditions, provided the model has been trained adequately over diverse conditions.

- **Efficiency of Sampling Process:**
  While practical implementations focus on computational efficiency, theoretically, the sampling process in DDPMs is guaranteed to produce samples from the correct distribution as long as the reverse chain is executed correctly. This includes maintaining the integrity of the step-wise denoising, which mathematically aligns with reversing the noise addition characterized by Gaussian distributions.

Advantages & Disadvantages:

Advantages :

High-Quality Generation:

- DDPMs are capable of generating high-quality samples that are competitive with or even surpass the quality of those produced by other state-of-the-art generative models like GANs (Generative Adversarial Networks) and VAEs (Variational Autoencoders). The fidelity and resolution of images, in particular, are notable.

Stability of Training:

- Unlike GANs, DDPMs do not suffer from training instability issues such as mode collapse or non-convergence. The training process of DDPMs is relatively stable because it relies on a straightforward loss function that directly measures the difference between predicted and actual noise, rather than relying on adversarial dynamics.

Theoretical Underpinnings:

- DDPMs are grounded in well-understood stochastic processes. Their theoretical foundation lies in reversing a Markov chain that transforms data into Gaussian noise, which provides a clear and rigorous framework for understanding model behavior and guarantees.

Flexibility and Control:

- DDPMs offer excellent control over the generation process through conditional generation techniques. They can be conditioned on various types of information (like class labels or other data modalities), allowing for targeted data generation that can be useful in diverse applications such as targeted drug design or controlled image synthesis.

Robustness to Hyperparameters:

- These models display a robustness to changes in hyperparameters. The performance of DDPMs does not degrade sharply with slight variations in learning rates or noise schedules, which is an advantage over more sensitive models like GANs.

Disadvantages :

Computational Cost:

- One of the primary drawbacks of DDPMs is their computational intensity. The reverse diffusion process involves many sequential steps, each requiring a forward pass of a potentially large neural network. This makes the generation process slower compared to direct sampling methods used in other generative models.

Complexity of Implementation:

- Implementing DDPMs can be complex due to the need for careful management of the noise schedule and the Markov chain dynamics. The architecture also typically relies on sophisticated neural network models like U-Nets, which can be challenging to optimize and scale.

Lack of Intuitive Control:

- While DDPMs allow for conditional generation, manipulating specific attributes of the generated samples in an intuitive way (e.g., changing a specific feature in an image) is less straightforward than in models like GANs, where latent space manipulations can often achieve this.

Energy Consumption:

- The energy consumption due to high computational requirements can be significant, which may not be ideal for applications requiring energy efficiency or rapid generation times, such as mobile or embedded systems.

Sampling Speed:

- The slow sampling speed of DDPMs, due to the iterative nature of the reverse diffusion process, limits their applicability in scenarios where real-time generation is crucial. This contrasts with GANs, where once trained, generation from the model is almost instantaneous.

Literature Survey:

- **Improved Denoising Diffusion Probabilistic Models:** This work introduces modifications that not only preserve the high sample quality of DDPMs but also achieve competitive log-likelihoods. A key advancement is learning the variances of the reverse diffusion process, which significantly reduces the number of forward passes needed during sampling. This enhancement is crucial for the practical deployment of DDPMs, as it makes them more computationally efficient without compromising sample quality (Proceedings of Machine Learning Research).

- **Structured Denoising Diffusion Models in Discrete State-Spaces (D3PMs):** Expanding the application of DDPMs to discrete data, this paper introduces models that handle different types of data corruption processes beyond uniform transition probabilities. This includes methods like using transition matrices that mimic continuous-space operations, improving results in domains like image and text generation. The introduction of a new loss function that combines the variational lower bound with an auxiliary cross entropy loss further refines performance, particularly noted in text generation (NeurIPS Proceedings).

- **Dynamic Programming Approach to Efficient Sampling:** Another innovative approach focuses on optimizing the inference schedule of DDPMs using a dynamic programming method. This approach decouples the training and inference schedules, allowing for optimized paths through pre-trained models based on the expected lower bound (ELBO). This method enhances efficiency by allowing for an optimal selection of timesteps during sampling, reducing computational overhead while maintaining or improving sample quality (ar5iv).

- **Progressive Distillation for Fast Sampling of Diffusion Models:** This paper introduces a progressive distillation technique that significantly speeds up the sampling process of diffusion models. By iteratively reducing the number of sampling steps required and using a teacher-student model setup, the approach improves sampling efficiency without sacrificing sample quality. This method is particularly valuable for practical applications where fast generation times are crucial.(ar5iv)