Reproduction Of The Denoising Diffusion Probabilistic Model AI6103 Project Report

Chen Zhitong (G2204728D),[1] **Fu Ziming (G2205212F),**[2]
Luo Yiyang (G2204807B),[3] **Wang Xinyue (G2204921G),**[4]

s220166@e.ntu.edu.sg,[1] zfu009@e.ntu.edu.sg,[2] luoy0043@e.ntu.edu.sg,[3] wang1913@e.ntu.edu.sg,[4]

Abstract

In this Project, we reproduce the model from the paper *Denoising Diffusion Probabilistic Models*, which presents an innovative image denoising approach employing diffusion processes. The method iteratively denoises images, leveraging a probabilistic model to estimate noise distribution and enable effective noise reduction. We implement the DDPM and train the model for the task of human face image synthesis. The project code is available at https://github.com/MRTater/AI6103-Project.

Introduction

THE *Denoising Diffusion Probabilistic Models* (DDPM) represents a significant advancement in generative modeling, introducing an approach based on diffusion processes for iterative denoising. This method has demonstrated state-of-the-art (SOTA) performance in various image processing tasks, including image synthesis, super-resolution, and inpainting, making it a versatile tool for diverse applications. Additionally, DDPM has inspired the development of other diffusion-based models, such as Stable Diffusion, further expanding its impact on the field.

In this Project, we focus on reproducing the DDPM based on our understanding of the original paper. We adapt the network architecture to suit our limited computational resources without compromising performance. Furthermore, we incorporate strategies such as learning rate scheduling, skip connections, better activations, etc. to enhance model performance. Our implementation targets the task of human face image synthesis, utilizing the *Synthetic Faces Dataset* to train the model.

Background

Over the past few years, deep generative models have gained popularity due to their ability to learn complex data distributions, generate high-quality samples, and perform a wide range of tasks, such as image synthesis, inpainting, and translation. Examples of deep generative models include Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), and autoregressive models like PixelCNN.

Despite their success, these models face challenges. GANs struggle with generating high-quality samples and avoiding mode collapse, while autoregressive models are computationally expensive and time-consuming. Consequently, researchers have sought alternative approaches to address these limitations.

In this context, denoising score matching has garnered attention, leading to the development of DDPM. This alternative generative model addresses some of the shortcomings of VAEs and GANs by using a diffusion process to learn a noise model within a continuous-time framework.

DDPMs have emerged as a promising alternative to traditional generative models due to their high-quality training samples and stability. The DDPM approach simulates a reverse diffusion process, transforming a simple noise distribution into a complex data distribution through a series of denoising steps. A neural network describes the denoising process, learning to predict the noise added at each step of the diffusion process.

Additionally, the diffusion process used in DDPM has applications in other domains, such as optimization and machine learning, leading to new algorithms and techniques. This highlights the significant impact DDPM has had on the machine learning field and its potential to drive further advancements in the future.

Dataset

We chose the Synthetic Faces High Quality (SFHQ) Part 1 dataset, as shown in Figure 1, for training our model. This dataset comprises 89,785 high-quality 1024×1024 curated face images created by "bringing to life" various artworks (paintings, drawings, 3D models) using a process similar to that described in a short Twitter thread. This process involves encoding the images into StyleGAN2 latent space and performing a minor manipulation that converts each image into a photo-realistic representation.

Data Preprocessing

Data preprocessing involves transforming raw data into a suitable range or scale that facilitates model learning. In our experiment, we performed several preprocessing steps on the data. Initially, we randomly selected 20,000 images from the dataset and resized them to 256×256 for our training

Figure 1: A preview of the forward diffusion process by time steps T

set. To further accelerate training, given limited computational resources, we resized the images to 64 × 64 and applied a random horizontal crop as data augmentaion. The image data, which initially consisted of integers ranging from 0 to 255, was linearly scaled to the range of [-1, 1] to ensure compatibility with the model.

Diffusion Foward Process

In the context of denoising diffusion probabilistic models (DDPM), the forward diffusion process, as described by Equation (1), is responsible for incrementally adding noise to the original image x_0 following a precise strategy at each timestep. This procedure aims to mimic the diffusion of the image across T timesteps, generating a sequence of increasingly noisy images that ultimately converge to Gaussian noise.

$$q\left(x_{1:T} \mid x_0\right)=\prod_{t=1}^{T} q\left(x_t \mid x_{t-1}\right) \tag{1}$$

The method of sampling noises is described by Equation (2). It is the conditional Gaussian noise with a mean that depends on the previous image and fixed variance, i.e., $(\beta_t I)$ in our implementation. The sequence of betas is called the noise schedule, which describes how much noise is added at each timestep.

$$q\left(x_t \mid x_{t-1}\right)=N\left(x_t ; \sqrt{1-\beta_t} x_{t-1}, \beta_t I\right) \tag{2}$$

Since the sum of Gaussian distributions is still a Gaussian distribution, as proven by Equation (3), given the set of betas, we can precompute various terms that will be employed in the forward diffusion process, such as the cumulative product of alphas ($\alpha_t^{cumprod}$), and 1 minus the cumulative product of alphas ($1-\alpha_t^{cumprod}$). These terms are utilized to control the variance and the noise introduced to the image at each timestep.

$$\begin{aligned}
&\mathrm{q}\left(\mathrm{x} t \mid x t-1\right)=\mathcal{N}\left(x_t, \sqrt{1-\beta_t} x_{t-1}, \beta_t I\right) \\
&=\sqrt{1-\beta_t} x_{t-1}+\sqrt{\beta_t} \epsilon \\
&=\sqrt{\alpha_t} x_{t-1}+\sqrt{1-\alpha_t} \epsilon \\
&=\sqrt{\alpha_t \alpha_{t-1}} x_{t-2}+\sqrt{1-\alpha_t \alpha_{t-1}} \epsilon \\
&=\sqrt{\alpha_t \alpha_{t-1} \alpha_{t-2}} x_{t-3}+\sqrt{1-\alpha_t \alpha_{t-1} \alpha_{t-2}} \epsilon \\
&=\sqrt{\alpha_t \alpha_{t-1} \cdots \alpha_1 \alpha_0} x_0+\sqrt{1-\alpha_t \alpha_{t-1} \ldots \alpha_1 \alpha_0} \epsilon \\
&q\left(x_t \mid x_0\right)=\mathcal{N}\left(x_t ; \sqrt{\bar{\alpha}_t} x_0,\left(1-\bar{\alpha}_t\right) I\right)=\sqrt{\bar{\alpha}_t} x_0+\sqrt{1-\bar{\alpha}_t} \epsilon
\end{aligned} \tag{3}$$

Beta Scheduling

The choice of the beta schedule has a significant impact on the performance of the DDPM. We propose two beta schedules: linear and cosine.

The linear beta schedule is defined as Equation (4):

$$\beta_t=\text{start}+\frac{t(\text{end}-\text{start})}{T}, \tag{4}$$

where T is the total number of timesteps, and start and end are the initial and final values of the beta schedule.
The cosine beta schedule is defined as Equation(5):

$$\begin{aligned}
&\bar{\alpha}_t=\frac{f(t)}{f(0)}, \quad f(t)=\cos\left(\frac{t/T+s}{1+s} \cdot \frac{\pi}{2}\right)^2, \\
&\beta_t=1-\frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}
\end{aligned} \tag{5}$$

The cosine beta schedule features a linear drop-off of $\bar{\alpha}_t$ in the middle while maintaining stable noise levels near $t=0$ and $t=T$as shown in Figure 2. This provides a more gradual noise addition compared to the linear schedule. A small offset s is introduced to avoid overly small β_t values near $t=0$. The schedule is defined using the cosine function, ensuring smooth noise transitions. By clipping beta values to a maximum limit, it prevents excessive noise and improves denoising performance, especially for lower resolution images.

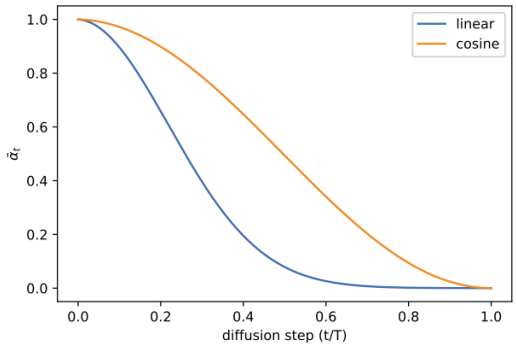

Figure 2: Linear and cosine beta schedules

Summary of the Forward Process

The forward diffusion process for a specific timestep t can be described as follows:

1. Sample Gaussian noise $\epsilon \sim \mathcal{N}(0, 1)$ with the same dimensions as the input image x_0.
2. Compute the noisy image x_t as Equation(4):

$$x_t = \sqrt{\alpha_t^{cumprod}} x_0 + \sqrt{1 - \alpha_t^{cumprod}} \epsilon, \tag{6}$$

where $\alpha_t^{cumprod}$ and $1-\alpha_t^{cumprod}$ govern the mean and variance of the noise introduced to the image, respectively. The noisy image is then clamped to the range $[-1, 1]$ to ensure valid pixel values.

The forward diffusion process is applied iteratively for all T timesteps as shown in Figure 3, producing a sequence of increasingly noisy images. During training, the DDPM model learns to invert this process by predicting the noise added at each timestep, enabling it to synthesize new images by denoising Gaussian noise in the later part.

Figure 3: A preview of forward difuusion process by time steps T

Parametrized Backward Process

The backward process in denoising diffusion probabilistic models (DDPM) is responsible for reconstruct the image x_0 back from the noisy image at the final timestep x_T. This process, parametrized by θ, is described by Equation (9):

$$\begin{aligned} p_\theta\left(x_{0:T}\right) &= p\left(x_T\right) \prod_{t=1}^{T} p_\theta\left(x_{t-1} \mid x_t\right) \\ x_{t-1} &\approx x_t - \text{ noise} \end{aligned} \tag{7}$$

The parametrized backward process is learned by the DDPM using a neural network, in our case, a simplified variant of the U-Net, along with sinusoidal position embeddings for the information of time steps. Notice that we only learn the mean, since the variance is fixes as mentioned.

U-Net

The U-Net serves as the core of the parametrized backward process in denoising diffusion probabilistic models. Its ability to capture both local and global information makes it ideal for denoising tasks.

The architecture maintains a large receptive field and preserves spatial information throughout the processing, enabling detailed image generation.

We use a simplified version of U-Net as shown in Figure 4, named SimpleUnet, consisting of an initial convolutional layer, a series of downsampling and upsampling blocks, and an output convolutional layer. The blocks incorporate time embedding and consist of convolutional layers, batch normalization layers, and activation functions.

The SimpleUnet is initialized with customizable settings like activation function (ReLU or SiLU), self-attention, and skip connections. Skip connections link corresponding downsampling and upsampling layers, propagating detailed spatial information to reconstruct denoised images more accurately. Additionally, there are skip connections within the blocks themselves, allowing for better information flow. The architecture has 5 downsampling and upsampling layers, with channels reaching up to 1024 in the middle layers. This structure captures hierarchical input image representations, including both high-level and low-level features.

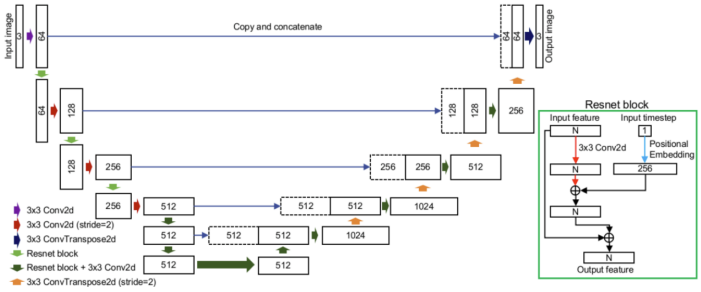

Figure 4: The SimpleUnet

Sinusoidal Position Embeddings

Incorporating temporal information is crucial in denoising diffusion probabilistic models (DDPM). To achieve this, we employ sinusoidal position embeddings, which provide a continuous and differentiable representation of time.

Sinusoidal position embeddings are computed using the following formulae:

$$\begin{aligned} P(k, 2i) &= \sin\left(\frac{k}{n^{2i/d}}\right) \\ P(k, 2i+1) &= \cos\left(\frac{k}{n^{2i/d}}\right) \end{aligned} \tag{8}$$

Where P represents the position embedding matrix, k is the time step, n is a constant (typically 10000), d is the dimensionality of the embeddings, and i is an index that ranges from 0 to $\frac{d}{2} - 1$. These equations generate sine and cosine functions with different frequencies, effectively encoding temporal information within the embeddings.

The resulting sinusoidal embeddings can be used within the DDPM, effectively encoding temporal information for the denoising process. The sinusoidal nature of these embeddings ensures that they can be easily combined with other representations, facilitating the learning process for the model.

Loss Function

In the DDPM, the loss function is used to measure the discrepancy between the actual noise and the predicted noise at each timestep during training. Here, we present two

different loss functions: L1 and L2.

The L1 loss function is defined as follows:

$$\mathcal{L}1(\theta) = \sum_{t=0}^{T-1} \|\epsilon_t - \text{model}(x_t, t)\|_1, \tag{9}$$

The L2 loss function, also known as the Mean Squared Error (MSE) loss, can lead to smoother denoising results as it tends to penalize large deviations more heavily. It is defined similarly to the L1 loss:

$$\mathcal{L}2(\theta) = \sum_{t=0}^{T-1} \|\epsilon_t - \text{model}(x_t, t)\|_2^2, \tag{10}$$

where ϵ_t is the true noise, x_t is the noisy image, t is the timestep, and $\text{model}(x_t, t)$ is the predicted noise at timestep t.

During training, either the L1 or L2 loss can be used to optimize the model, depending on the desired denoising characteristics.

Summary of the Backward Process

The parametrized backward process as shown in Figure 5, in DDPM sythesize the original image x_0 from the noisy image x_T. The process involves a simplified U-Net architecture and sinusoidal position embeddings for encoding temporal information.

1. Perform batch training with noisy images x_T and corresponding time steps.
2. Encode time steps using sinusoidal position embeddings to generate a position embedding matrix P.
3. Process input images using the SimpleUnet and feed in with the sinusoidal position embeddings to incorporate temporal information in the denoising process.
4. Iterate through the training process, adjusting model parameters to minimize the loss function and refining the denoising capability.
5. By following these steps, the parametrized backward process effectively synthesizes the original images from the noisy counterparts, improving the performance of the denoising diffusion probabilistic model.

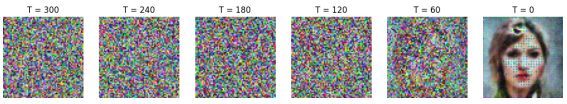

Figure 5: 64*64, T=300, linear, sythesized image via backward process

References