Image generation with diffusion models

Réka Boros and Tamás Szűcs

Dec 2024

1 Introduction

Diffusion models are based on a Markov chain that gradually adds noise to the data and then restores the original data by removing the noise. These models are particularly popular in fields like image generation and segmentation because the samples they produce often rival other generative models, such as GANs (Generative Adversarial Networks).

The operation of diffusion models can be divided into two main steps: first, the input data is gradually distorted through a step-by-step process, and then the distorted data is transformed back into its original form. Below, we will present one of the most popular approaches, the U-Net-based architecture (which we also used in our project), and the application of attention mechanisms that contribute to the success of these models.

In this project work we implemented and trained a U-Net architecture with time embedding. We tried to improve our model's performance by integrating channel attention and spatial attention into the architecture. We trained the model on the Flower102 and CelebA datasets, for which we created a module to handle these datasets. We used the image generation method from another notebook and modified it to fit our project. We could not replicate the results seen in the articles cited however, we did get meaningful results.

2 Literature, Previous works

In the first article [1] titled "Denoising Diffusion Probabilistic Models", the authors introduce diffusion models for image generation. Denoising Diffusion Probabilistic Models (DDPMs) revolutionize generative modeling by producing high-quality samples without adversarial training.

These models rely on a two-step process: a forward diffusion process progressively adds Gaussian noise to data, turning it into random noise, while a reverse process, trained as a parameterized Markov chain, removes the noise to reconstruct the original data.

The training leverages a variational inference objective, refined for simplicity and efficiency, ensuring robust and stable learning. DDPMs achieve state-of-the-art results on datasets like CIFAR-10 (Inception Score of 9.46, FID of 3.17) and LSUN (256×256 resolution), rivaling advanced methods like GANs.

Beyond generating high-quality images, DDPMs enable innovative applications such as progressive lossy compression and smooth interpolation in latent space. However, the iterative nature of the sampling process, requiring thousands of steps, limits computational efficiency compared to GANs.

In the second article [2] the authors introduce several improvements to the original method. Denoising Diffusion Implicit Models (DDIMs) build upon Denoising Diffusion Probabilistic Models (DDPMs), addressing their primary limitation of slow sampling.

By replacing the stochastic reverse process with a deterministic trajectory, DDIMs significantly enhance efficiency while preserving the same training framework and sample quality. Unlike DDPMs, which require thousands of Markov chain steps, DDIMs use a non-Markovian forward process, enabling sample generation $10\times$ to $50\times$ faster. This approach also ensures consistency in high-level features across different sampling trajectories, allowing meaningful image interpolations and precise reconstructions from latent representations.

Experiments on datasets like CIFAR-10 and CelebA show that DDIMs achieve comparable or superior Frechet Inception Distance (FID) scores while reducing computational cost. Additionally, DDIMs' deterministic nature provides a robust mechanism for latent space manipulation, supporting applications like image interpolation and encoding.

3 Our model, and method for image generation

Data preparation

When we are preparing the data, we are essentially doing the forward step of the image generation process. We implemented a module that creates noisy image and timestep pairs from the Flower102 and CelebA datasets. We use 500 timesteps and cosine noise schedule, because with linear noise schedule the information vanishes very quickly, and we have pure noise in about half of the timesteps.

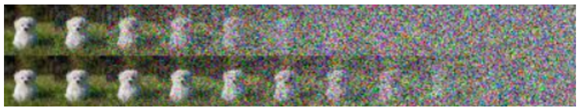

Figure 1: Linear noise schedule (top) vs. cosine noise schedule (bottom)

Model architecture

We used the U-Net architecture. This architecture consists of an encoder part with 5 downlevels and a decoder part with 5 uplevels. In each downlevel we double the number of channels and reduce the height and width of the latent space by a factor of 2. In the uplevels we reverse this process by reducing the number of channels by a factor of 2, and double the height and width of the latent space in each uplevel. This way the output will have the same dimensionality as the input.

We also add skip-connections connecting the latent spaces with matching dimensionality from the first (encoder) part of the network with the second (decoder) part of the network. This helps preserve low-level features while also obtaining high-level features, and therefore improves the model's performance.

We add sinusoidal time embedding in each level. This is necessary for the network to be able to learn how much noise to expect on an image given the timestep. We use 32 dimensions for the time embedding then, with a fully connected layer we map this vector to match the number of channels at the given level.

$$\mathrm{PE}(t, 2i) = \sin\left(\frac{t}{10000^{\frac{2i}{d}}}\right), \quad \mathrm{PE}(t, 2i+1) = \cos\left(\frac{t}{10000^{\frac{2i}{d}}}\right)$$

We also tried to add attention modules to the network [3]. The Attention U-Net enhances the basic U-Net architecture by integrating attention mechanisms at each level, consisting of two main components: Channel Attention and Spatial Attention modules.

The Channel Attention module learns the importance of individual channels by performing global average pooling across them, followed by scaling their weights through two linear layers, enabling the model to selectively emphasize channels carrying more significant information.

Meanwhile, the Spatial Attention module focuses on spatial dimensions, determining the importance of individual pixel locations using global average pooling and max pooling, followed by a convolutional layer to generate spatial attention weights.

These mechanisms help the model highlight relevant regions in the image while ignoring less important areas, improving feature representation and overall performance.

In the case of the spatial attention, applying sigmoid to the attention scores then multiplying it with the actual features was proven to be too strong. When we tried to generate images, it erased almost all pixels leaving only a small portion that it found important, and even in this area it only generated noise. Scaling the attention values after applying sigmoid to the $[\frac{1}{2}, 1]$ interval solved this issue and improved performance.

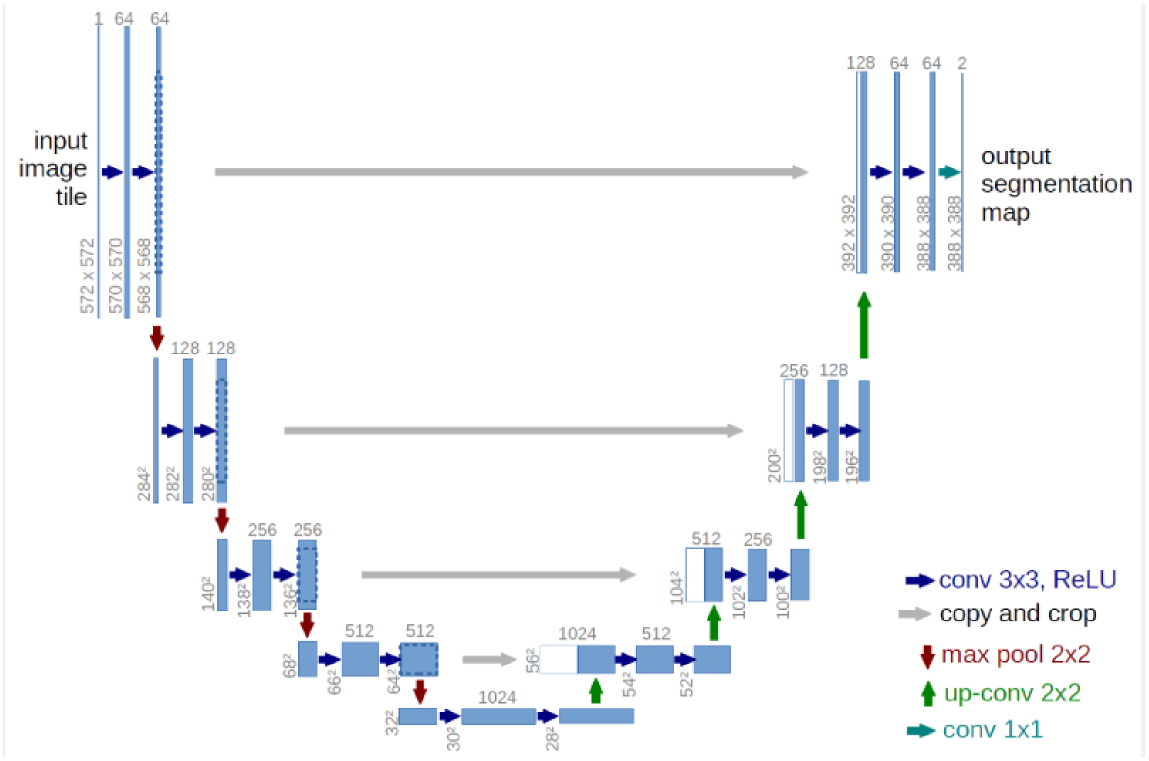

Figure 2: U-Net architecture

Image generation

We can see the formulas we used to generate images [1]. Here α_t values are gained by dividing the $[0, 1]$ interval into T equal parts. In the final version, we use $\sigma_t = 1 - \alpha_t$, as this yields meaningful results for both CelebA and Flower102 datasets. However it is worth mentioning, that if we use $\sigma_t = \sqrt{1 - \alpha_t}$, on the Flower102 dataset we get essentially noise, but on the CelebA dataset it improves the performance significantly.

$$z = \begin{cases} 0 & \text{if } t = 0 \\ \mathcal{N}(0, I) & \text{otherwise} \end{cases}$$

$$x_{t-1} = \sqrt{\frac{1}{\alpha_t}} x_t - \sqrt{\frac{1-\alpha_t}{1-\bar{\alpha}_t}}\, \theta(x_t, t) + \sigma_t z$$

$$\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$$

4 Training and results

Contrary to common practice, we chose not to use a validation set. The reasoning behind this decision is that we did not notice signs of overfitting in the model in either of the datasets, it was rather underfitting, and we wanted to keep as much data for training as possible. The other purpose of the validation set would have been early stopping during training. However in the case of Flower102 dataset, we had around 1000 images to train the U-Net which is not a lot, and we wanted to use all of it, in the case of CelebA we had a lot of images, but then we did not need that many epochs, so we had a rough idea how long we want to train the model.

On the Flower102 dataset we trained the model for 100 epochs, on the CelebA dataset we trained it for 20 epochs. In both cases we set the image size to 64×64, we used Adam optimizer with learning rate $1e-4$ and weight decay $1e-5$ and cosine learning rate schedule with learning rate ratio $1e-3$. We used MSE loss as suggested in [1], and we used batchsize of 24.

To show our results, we include below a handful of pictures generated by our Attention-models.

5 Conclusion

We can see that our pictures resemble faces and flowers, however the clarity and authenticity are lacking. Further improvements could be made by first reviewing some of the formulas and adjusting noise levels both during training and inference as the model is sensitive to small changes in these parameters. Also the resolution of the images could be improved.

We did not manage to replicate the results seen in the papers, however we can see the potential in these techniques, there are also some quite successful images, and with some further improvements and adjustments we could have a model that generates authentic images, that are hard to judge whether it is generated or real.

6 LLM usage

We used Chatgpt to write the train function, and then we modified it a bit. The Attention module was implemented using Chatgpt, and it did a great job with that, only a slight adjustment had to be made. We used Chatgpt to write the plot function and we also used Chatgpt in the introduction and literature parts of the documentation, then corrected it. Also we used it to correct grammer in the document.

References

[1] P. Abbee J. Ho, A. Jain. Denoising diffusion probabilistic models. *arXiv:2006.11239v2*, 2020.

[2] S. Ermon J. Song, C. Meng. Denoising diffusion implicit models. *arXiv:2010.02502v4*, 2022.

[3] A. Nichol P. Dhariwal. Diffusion models beat gans on image synthesis. *arXiv:2105.05233v4*, 2021.

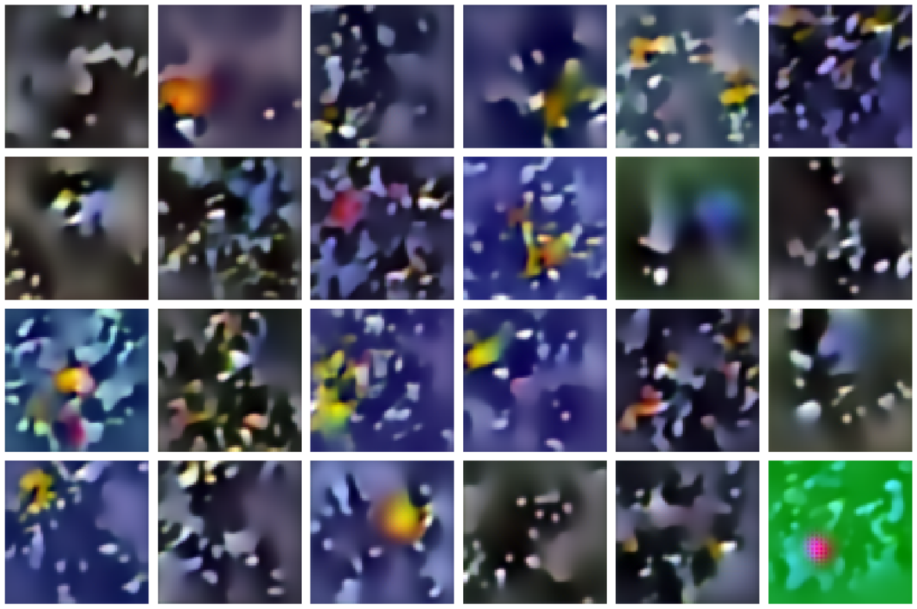

Figure 3: Flower images

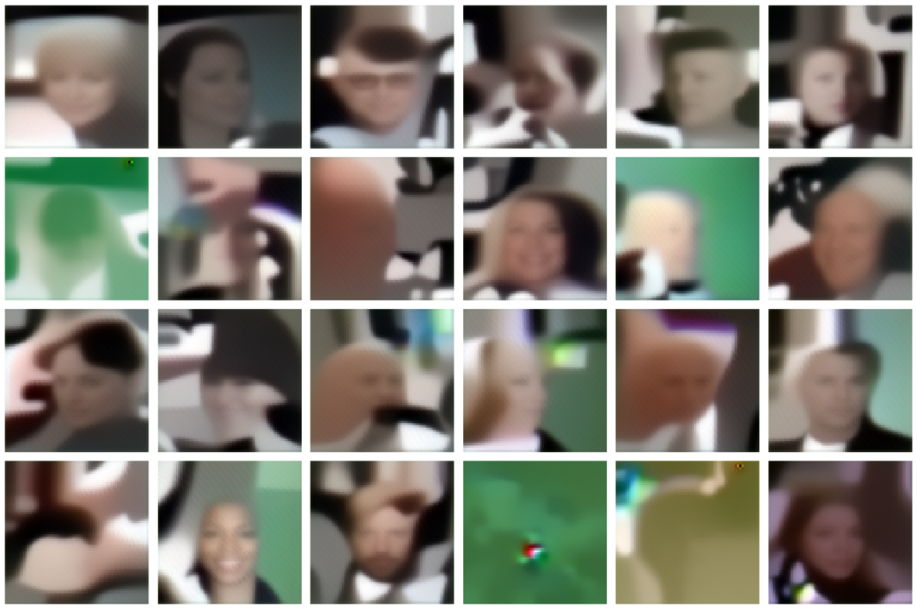

Figure 4: Celeb images