Denoising Diffusion Probabilistic Models: Generative AI

Samer Bujana (s240033), Rasmus Clausen (s203279), Alba Gonzalo (s243343), Leonardo Rodovero (s240095)

DTU

Introduction

Denoising Diffusion Probabilistic Models (DDPM) are a class of generative models that have recently gained significant attention for their ability to generate high-quality data, particularly in image generation tasks. In this project, we studied the principles and techniques behind these models, focusing on their application to image generation and noise removal using the MNIST dataset. Additionally, we explored the influence of various model components, including different schedulers, time steps, and conditional training, to assess how these factors impact performance.

Background

- **Forward (Diffusion) Process:** Noise is gradually added to an original sample x_0 over time. We define a variance schedule $\{\beta_t\}$, which controls the amount of noise added at each time step. This schedule can be rewritten in terms of $\{\overline{\alpha}_t\}$, allowing us to directly obtain a noised image at time step t from the original, bypassing the intermediate steps. The relationship is given by:

$$x_t = \sqrt{\overline{\alpha}_t}\, x_0 + \sqrt{1-\overline{\alpha}_t}\,\epsilon, \quad \epsilon \sim \mathcal{N}(0, I),$$

where $\alpha_t = 1 - \beta_t$ and $\overline{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. As $t \to T$, x_t approaches isotropic Gaussian noise.

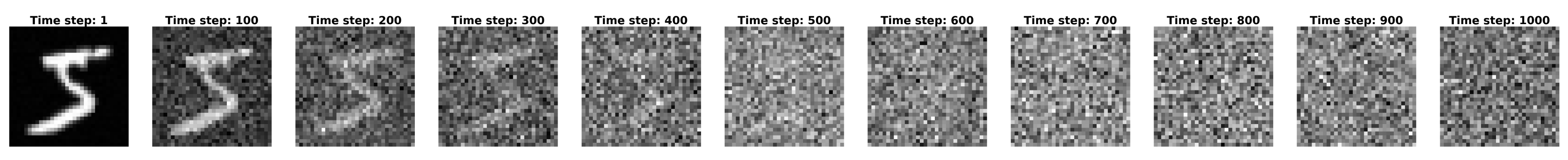


Figure 1: Visualization of forward process with linear schedule.

- **Reverse (Denoising) Process:** This process reverse the forward process by progressively removing the noise, starting from a noisy sample. Unlike the forward process, it is not feasible to directly recover the original image from the noisy one, as this is intractable and unstable. Therefore, we utilize a model (e.g., U-Net) to predict the noise added (ϵ_θ) and subtract it from the noisy image. The equation for recovering the image at each reverse time step is:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon_\theta(x_t, t)\right) + \sqrt{\beta_t} z$$

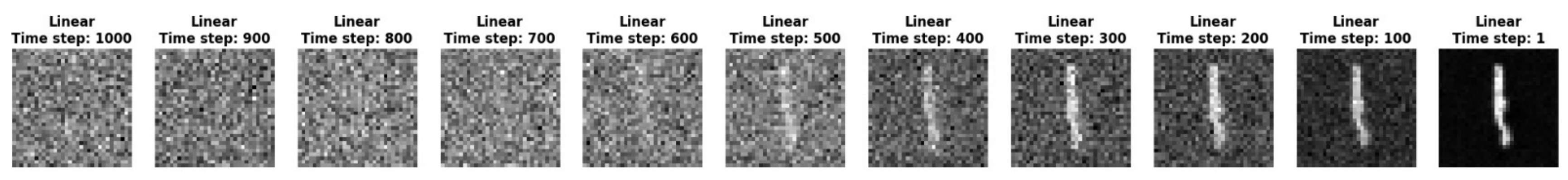


Figure 2: Visualization of reverse process with linear schedule.

- **Training Objective:** We aim to optimize the variational upper bound of negative log-likelihood of $p_\theta(x_0)$, which has been shown to be equivalent to optimizing the Mean Squared Error (MSE) between the true noise ϵ and its predicted counterpart ϵ_θ.

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

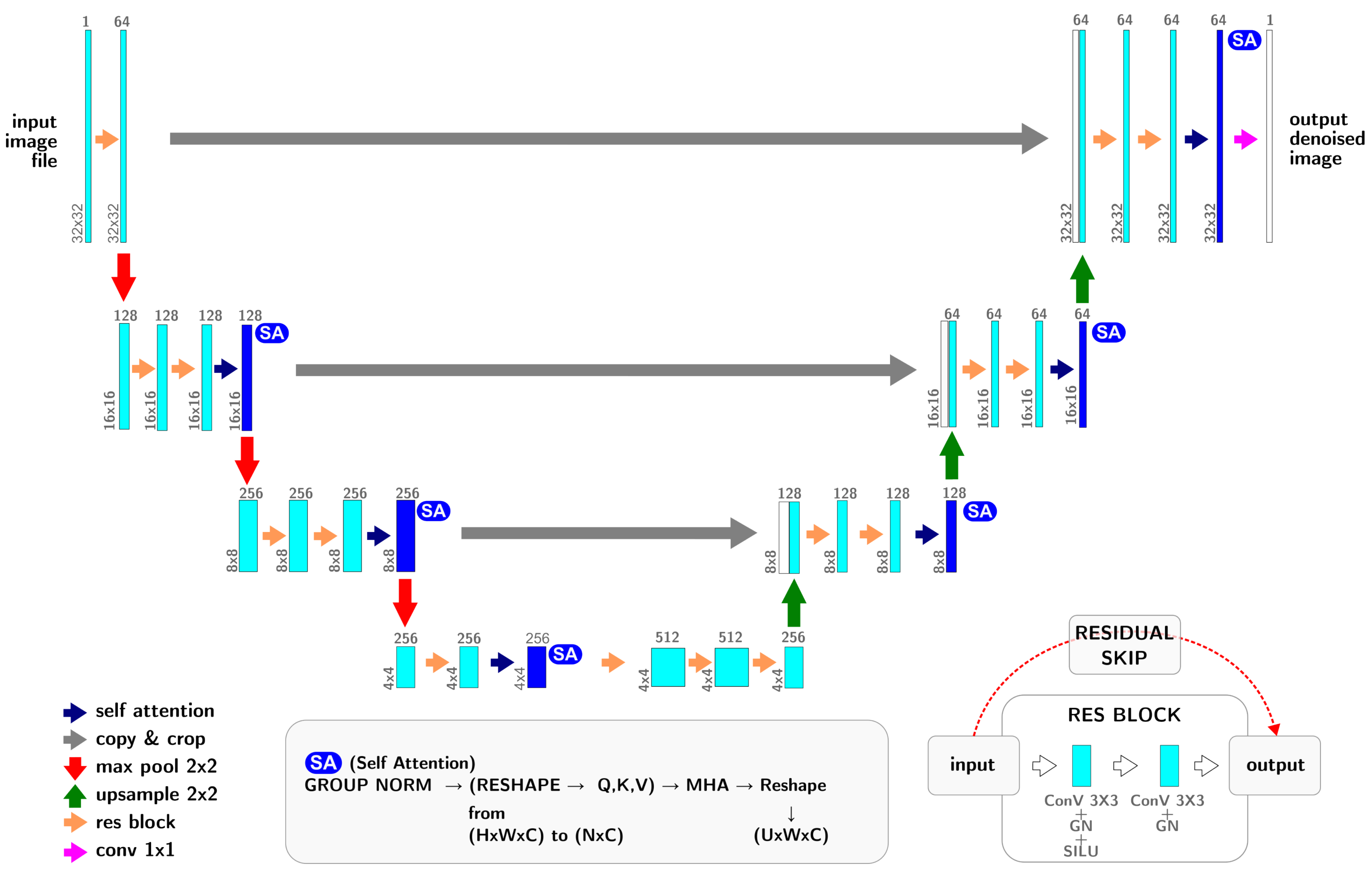
- **FID (Frechet Inception Distance):** To evaluate the effectiveness of image generation, the FID score is used, which is based on calculating the Frechet distance between the feature distributions of generated and real images.

$$\text{FID}(P, Q) = \|\mu_P - \mu_Q\|_2^2 + \text{Tr}(C_P + C_Q - 2(C_P C_Q)^{1/2})$$

Experimental Setup and Key Points

- **Dataset:** MNIST (60,000 training images, 10,000 test images) resized to 32×32, normalized.
- **Training/Validation Split:** **80%** Training, **20%** validation from the 60,000 training images.
- **Training Configuration:**
  - ▷ **Epochs:** 20
  - ▷ **Loss:** MSE
  - ▷ **Optimizer:** Adam (Learning rate $= 0.0001$)
  - ▷ **Batch Size:** 64

UNet Architecture



Time Steps & Scheduler Impact

We employed different number of time steps and schedulers in the DDPM to examine their impact on image generation. Initially, we employed the linear scheduler, as described in [3]. Later, we tested the cosine scheduler, based on the advancements presented in [5], which demonstrate that the linear scheduler is less effective for generating lower-resolution images. Finally, we also explored the sigmoid scheduler, introduced in [1] and [4]. The results obtained are summarized in Table 1 and Figure 3.

Linear: $\beta_t = \text{linear interp}(\beta_{\text{start}}, \beta_{\text{end}}, t)$, where we set up $\beta_{\text{start}} = \mathbf{0.0001}$, $\beta_{\text{end}} = \mathbf{0.02}$.

Cosine: $\overline{\alpha}_t = \frac{f(t)}{f(0)}, \quad f(t) = \cos\left(\frac{t/T+s}{1+s}\cdot\frac{\pi}{2}\right)^2$, where we set up $s = \mathbf{0.008}$.

Sigmoid: $\overline{\alpha}_t = \left(\frac{1-\sigma(t)-\min(\overline{\alpha}_t)}{\max(\overline{\alpha}_t)-\min(\overline{\alpha}_t)}\right)\cdot \mathbf{0.9999} + \mathbf{0.0001}$, where σ is the sigmoid function.

Timesteps	100	500	1000
Linear Scheduler	28.247	9.239	**7.823**
Cosine Scheduler	9.207	7.023	**5.735**
Sigmoid Scheduler	7.734	5.773	**5.039**

Table 1: FID scores for different schedulers and time steps, training for 20 epochs.

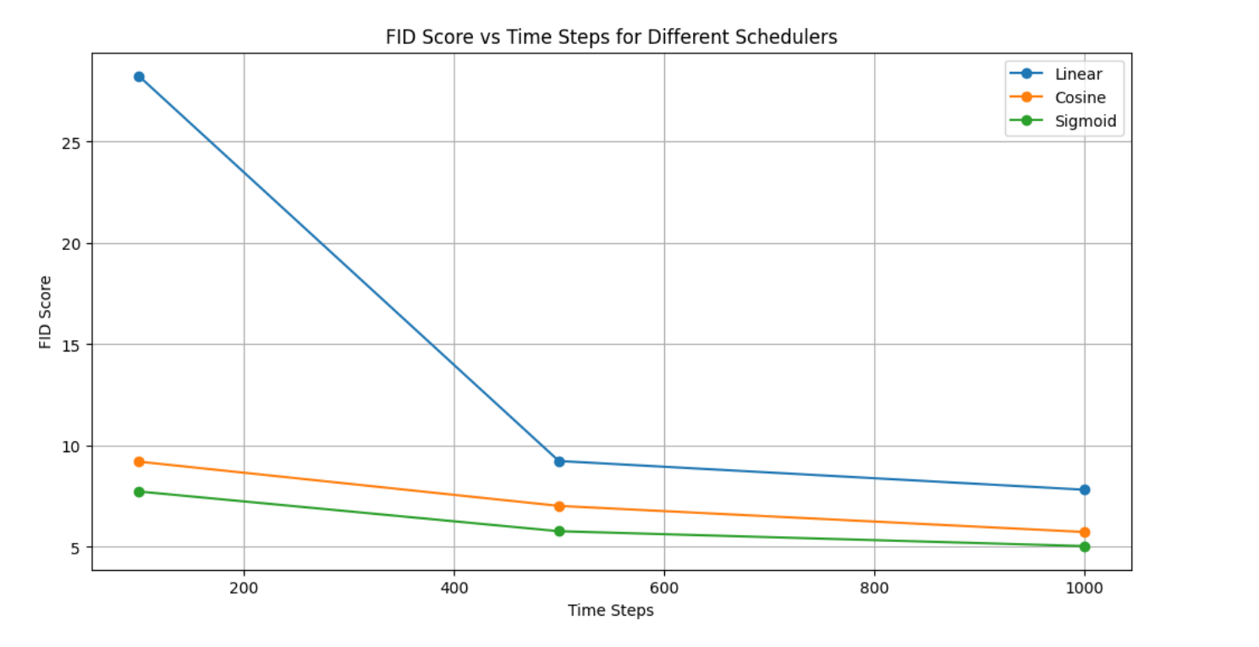


Figure 3: FID scores over timesteps.

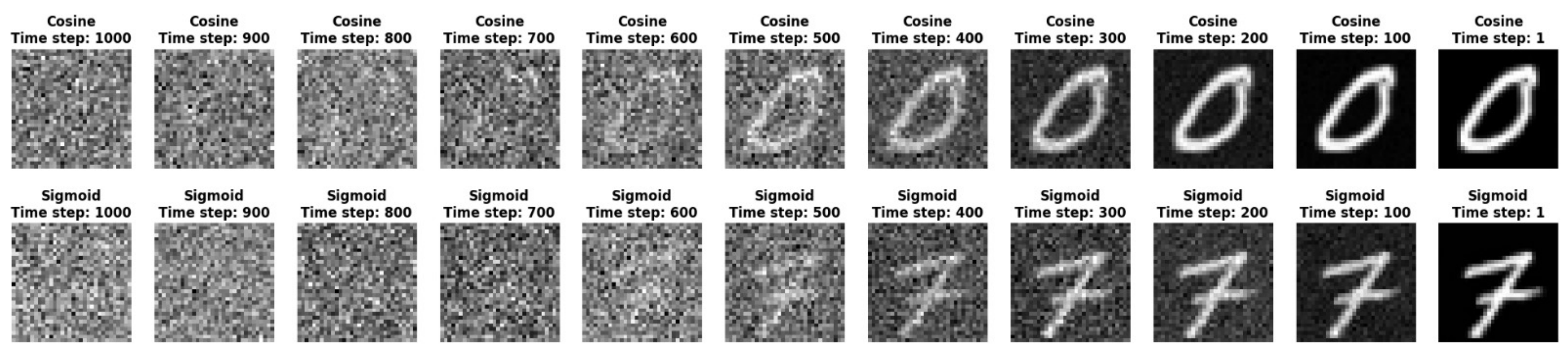


Figure 4: Reverse process for the cosine and sigmoid schedulers.

Generated Images for Different Schedulers

According to Table 1, the best performance for each scheduler was achieved with 1000 time steps. This section presents the generated images, using these time steps to visualize the models performance across different schedulers. All models were run for 20 epochs.

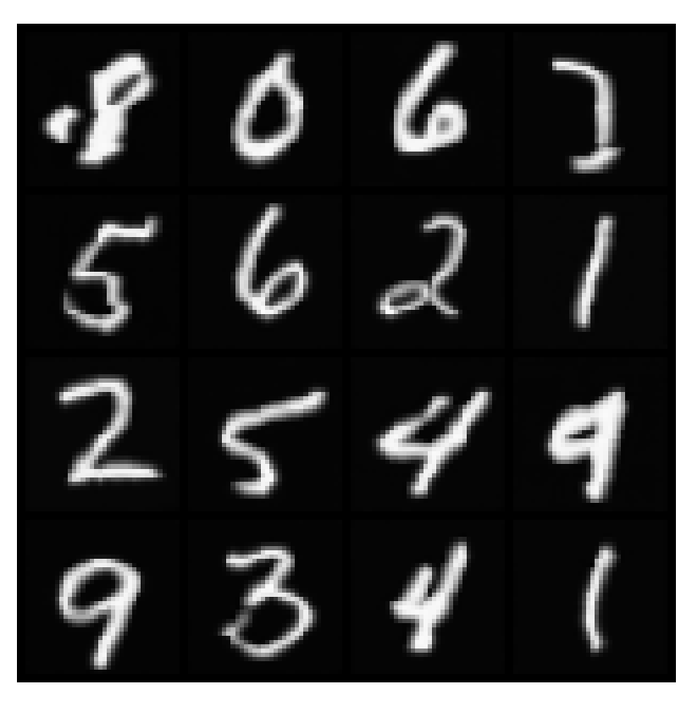


Figure 5: Linear.

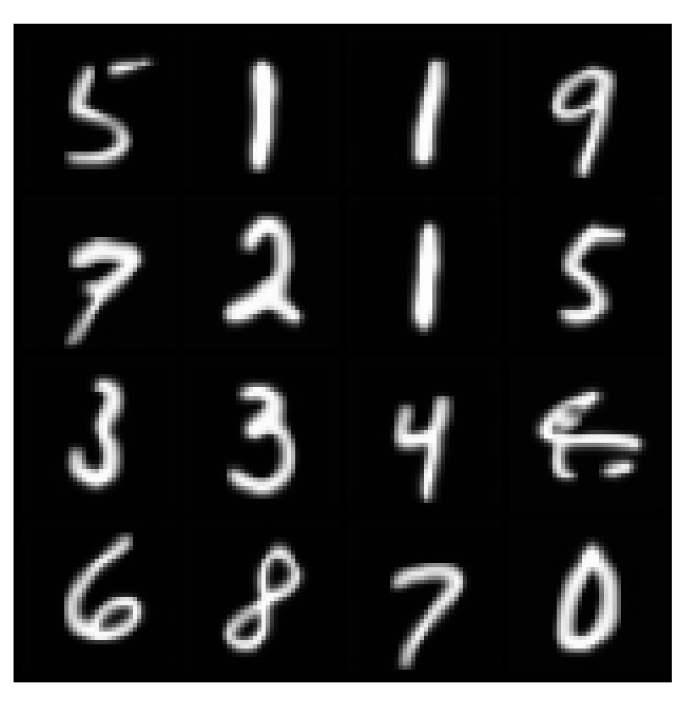


Figure 6: Cosine.

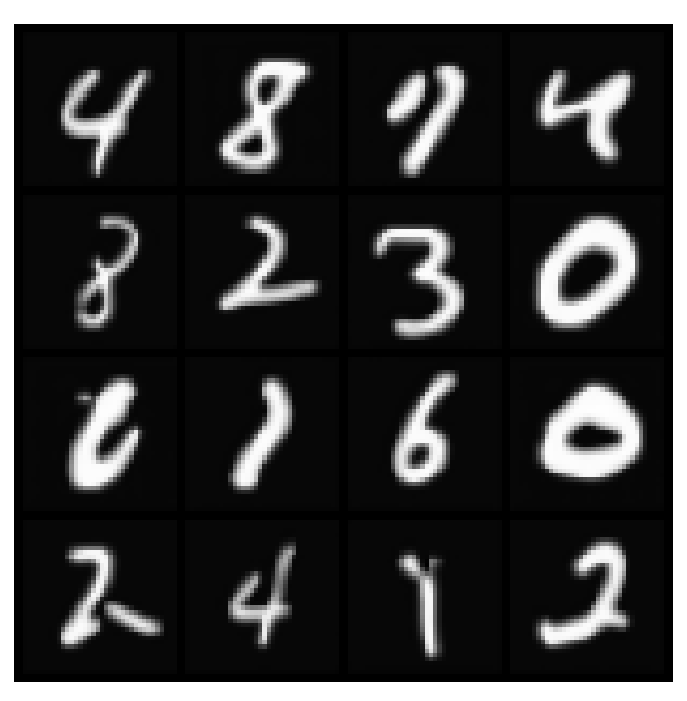


Figure 7: Sigmoid.

Classifier Free Diffusion Guidance

We implemented conditional training using Classifier Free Diffusion Guidance [2]. This was done by modifying our model with a class embedding for each digit. This works by concatenating the digit embedding to the feature maps before each UpBlock, DownBlock, and the Resblocks in the latent space of our UNet.
During training the embedding is added, but with some probability p a null embedding containing only zeros would be added. Thus, the model learns to predict noise both with and without the digit embedding.
During the backward process, the subtracted noise is guided using the formula:

$$\epsilon_{\text{CFG}} = (w+1)\cdot\epsilon_\theta(x_t, t, y) - w\cdot\epsilon_\theta(x_t, t)$$

Where w is a weight > 0, controlling how much the backwards process is guided towards the given label. By increasing w the quality of the sampled digits increases, but they also lose variety. Guidance generally improves the FID score as seen in Table 2 and Figure 8.

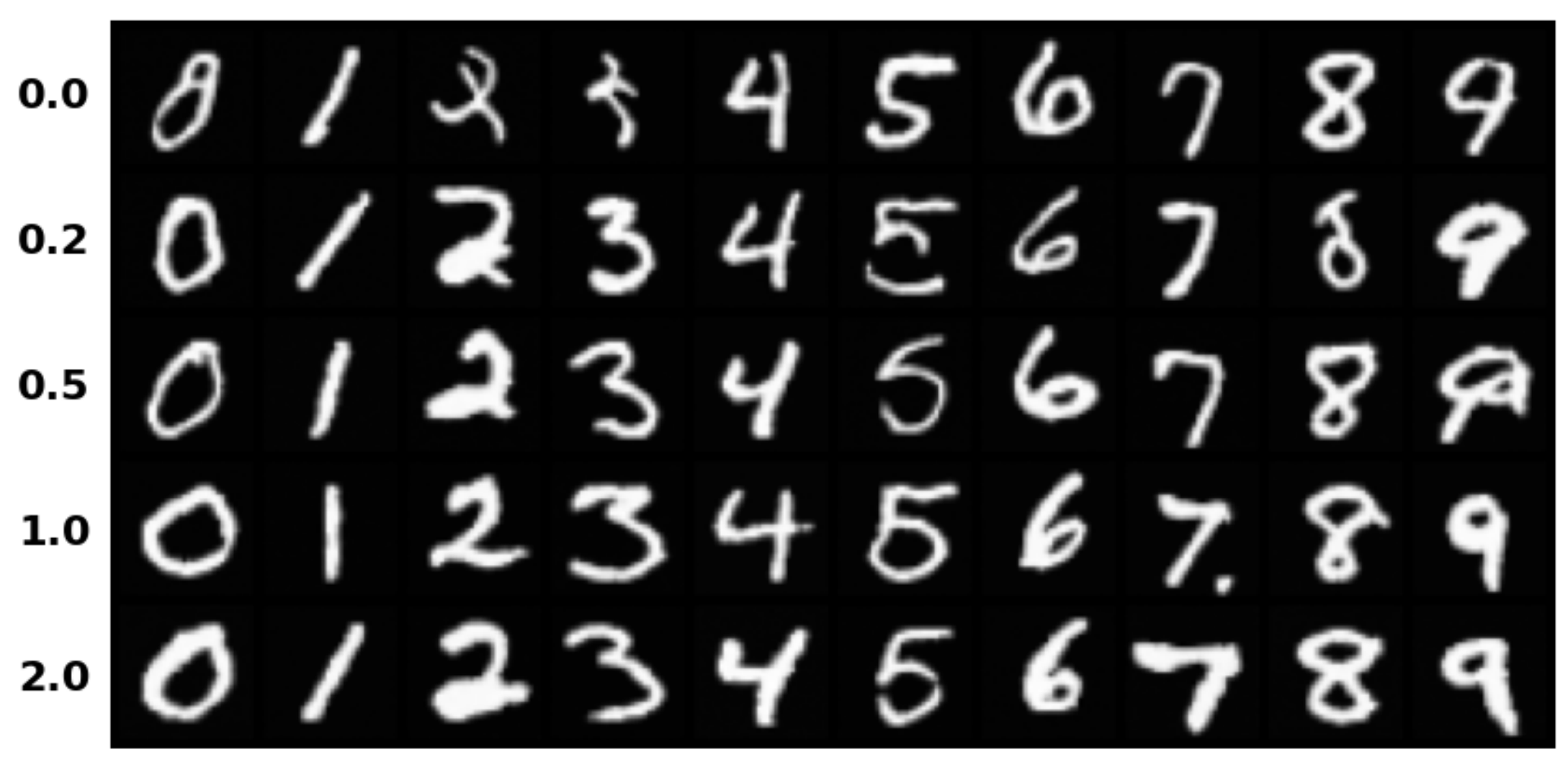


Figure 8: Visualization of digits sampled using CFG with different classification weights w.

weight(w)	FID
0	1.792
0.1	1.113
0.2	1.14
0.5	0.608
1	0.464
2	0.578
10	1.839
50	14.669

Table 2: FIDs for different guidance values w.

References

[1] T. Chen et al. On the importance of noise scheduling for diffusion models. *arXiv preprint arXiv:2301.10972*, 2023.
[2] J. Ho and T. Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.
[3] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
[4] A. Jabri, D. J. Fleet, and T. Chen. Scalable adaptive computation for iterative generation. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.
[5] A. Q. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.