Project Machine Learning
— Milestone 2 —

Seif Daknou, Patrick Lindemann, Nguyen Pham

January 8, 2024

1 Introduction

Building on our foundational work from the first milestone, we focus on implementing the reverse process and the image sampling procedure, which are both integral components of the Denoising Diffusion Probabilistic Models (DDPM) framework proposed by Ho et al. (2020), in this phase. Furthermore, we expand our U-Net which was presented in the first report with temporal embeddings and train it on two new datasets, namely *MNIST*[1] and the *FGVC-Aircraft Benchmark*[2]. Finally, we use our implementation to generate new images based on aforementioned datasets and evaluate our results using the Inception Score (IS), a metric introduced by Salimans et al. (2016) and applied in the DDPM paper for assessment purposes.

2 Reverse Diffusion Process

In this section, we begin by examining the mathematical foundations of the reverse diffusion process, building upon the definitions introduced in the first milestone. Following this, we present the revised U-Net architecture for noise prediction, discuss the training phase, and finally, delve into the sampling process.

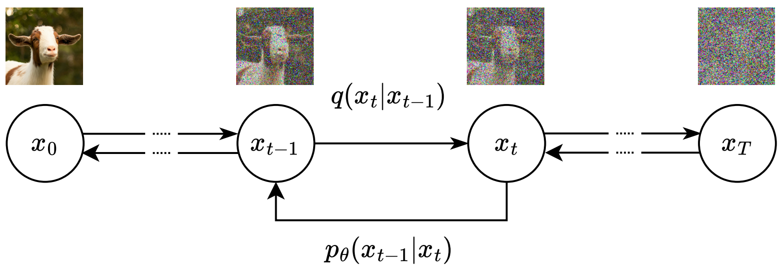

Figure 1: The *forward diffusion kernel* $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ applies gaussian noise gradually to an image $\mathbf{x}_0$, while the (learned) *reverse diffusion kernel* $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ denoises the image between two consecutive time steps t and $t-1$.

2.1 Background

For the mathematical derivation, we use the equations of Nichol and Dhariwal (2021), Singh (2023) and Hanjin (2021), which build the work of the original DDPM paper by Ho et al. (2020).

Let $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ be an image sampled from the original data distribution, which is corrupted gradually in T time steps by the *forward noising process*. This is done through the *forward diffusion kernel* $q(\mathbf{x}_t|\mathbf{x}_{t-1})$, which adds a small amount of Gaussian noise to the image $\mathbf{x}_t$ at time step t. We model the joint distribution of the process as a Markov chain with

[1] http://yann.lecun.com/exdb/mnist/
[2] https://www.robots.ox.ac.uk/~vgg/data/fgvc-aircraft/

$$q(\mathbf{x}_1, \ldots, \mathbf{x}_T|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{1}$$

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_t - 1, \beta_t\mathbf{I}) \tag{2}$$

where $\beta_t \in [0, 1]$ is the variance generated according to a *schedule*. We reparameterize the variances with $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ to enable sampling from the distribution $q(\mathbf{x}_t|\mathbf{x}_0)$ at an arbitrary time step t (refer to the report of milestone 1 for more details).

Given that the forward diffusion kernel is represented as a Gaussian distribution, the *reverse diffusion kernel* $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ has the same functional form (Feller (1966)) and thus can be modeled by another Gaussian. Ho et al. (2020) parameterize the mean $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$ and variance $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)$ such that can be learned by a neural network and provide the following formulas:

$$p_\theta(\mathbf{x}_0, \ldots, \mathbf{x}_T) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \tag{3}$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)) \tag{4}$$

Starting at a sample of pure noise $\mathbf{x}_T$, the probabilifty density function of the reverse diffusion process is the integral across all conceivable pathways leading to a data sample $\mathbf{x}_0$ (Singh (2023)). Thus, we formulate the marginal $p_\theta(\mathbf{x}_0)$ and the resulting loss function L for the model as

$$p_\theta(\mathbf{x}_0) := \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} \tag{5}$$

$$L := -\log(p_\theta(\mathbf{x}_0)) \tag{6}$$

Consequently, the training objective amounts to maximizing the log-likelihood of the sample generated by the reverse process belonging to the original data distribution. However, L is intractable since we need to compute the integral over a high-dimensional image space for continuous values over T time steps. Ho et al. (2020) solve this by reformulating the objective using a variational lower bound (VLB) to formulate the tractable loss function

$$L_{\text{VLB}} := \mathbb{E}_q \left[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \| p(\mathbf{x}_T))}_{L_T} + \sum_{t=2}^{T-1} \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \| p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \right] \tag{7}$$

To further simplify the objective, the term L_T is ignored since it doesn't contain any parameters, and the term L_0 is left out since the authors got better results without it. This leaves the sum of the independent terms L_{t-1}, and since we can sample from an arbitrary time step t at the forward process due to the re-parameterization with α_t, we can use the expectation $\mathbb{E}_{t,\mathbf{x}_0,\epsilon}[L_{t-1}]$ to estimate L_{VLB} (Nichol and Dhariwal (2021)). After some transformations, we get

$$L_{t-1} = \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}} \left[\frac{1}{2\sigma_t^2} \| \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t) \|^2 \right] + C \tag{8}$$

Moving forward, this leaves us with multiple approaches of parameterizing $\mu_\theta(\mathbf{x}_t, t)$: Predict $\mathbf{x}_0$ and use it to calculate $\mu_\theta(\mathbf{x}_t, t)$, predict $\mu_\theta(\mathbf{x}_t, t)$ directly, or predict the noise $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ at each time step t and derive $\mu_\theta(\mathbf{x}_t, t)$ with

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) := \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) \tag{9}$$

Ho et al. (2020) found that the best results were achieved by predicting $\boldsymbol{\epsilon}_\theta$. Following the exclusion of certain weight terms, they derived the final form of the loss function

$$L_{\text{simple}} := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}} \left[\left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2 \right] \tag{10}$$

$$= \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}} \left[\left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(x_t, t) \right\|^2 \right] \tag{11}$$

which is used to train the noise prediction model introduced in the next section.

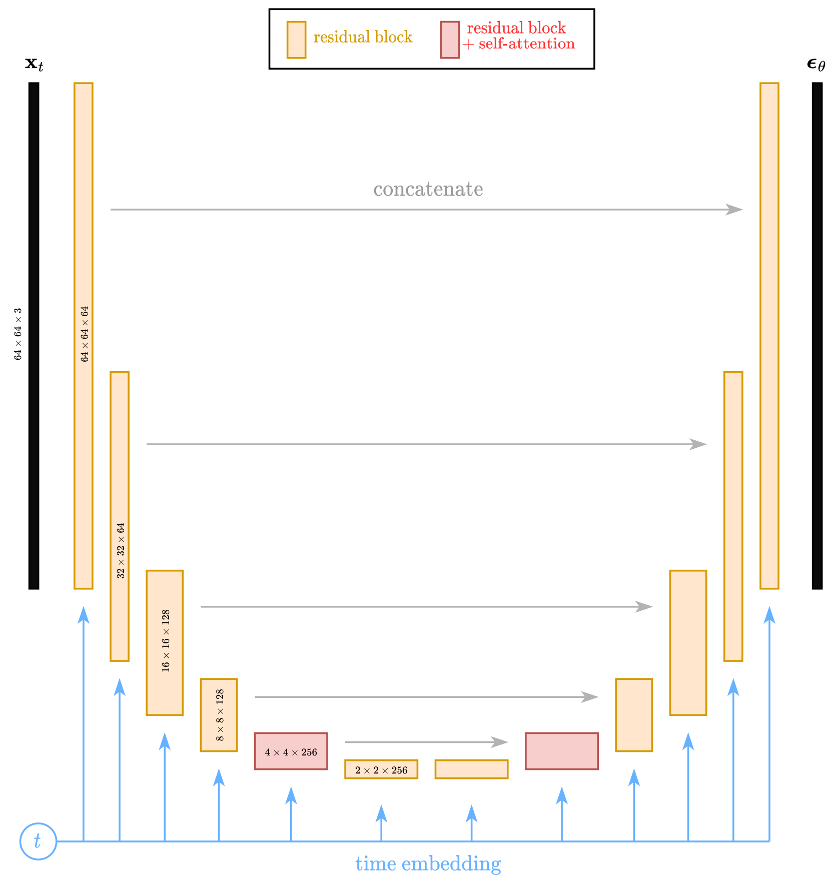

Figure 2: The architecture of the 64×64 noise prediction U-Net model.

2.2 Noise Prediction Model

In the experiments performed by Ho et al. (2020), the authors use a U-Net based on a wide ResNet following the structure of *PixelCNN++*[3]. For their 32×32 models, they employ four feature map resolutions ranging from 32×32 to 4×4, while their 256×256 models utilize six resolutions. All models include two convolutional residual blocks per resolution level, with self-attention modules at the 16×16 resolution positioned between the convolutional blocks. Further, they simplify their implementation by replacing weight normalization with group normalization. At last, they supplement the diffusion time for each block through sinusodial time embeddings, effectively encoding the time step t corresponding to the image $\mathbf{x}_t$ through a combination of sinus and cosine functions with different frequencies.

Given that U-Net models are not the primary focus of our work, we shifted from or custom U-Net implementation from the previous milestone to *UNetModel2D*[4], an implementation available at *Huggingface*. We employed two distinct models to handle the resolutions 64×64 and 32×32 with 6 or 5 down- and upward blocks respectively with 2 residual layers each. Further, the model now incorporates the aforementioned sinusoidal time embeddings which we ignored in our first report. We note that training for input and sampling sizes larger than 64×64 was not possible yet due to time and hardware constraints.

[3]`https://github.com/openai/pixel-cnn`
[4]`https://huggingface.co/docs/diffusers/api/models/unet2d`

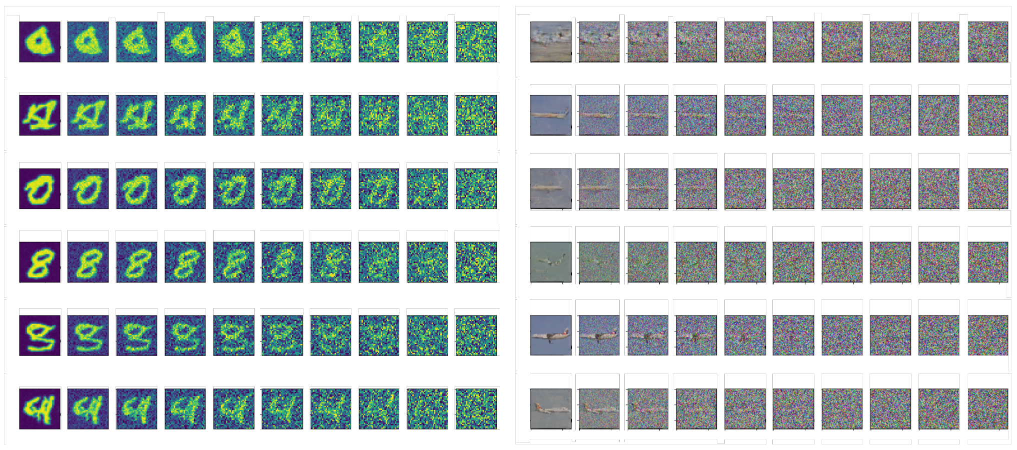

Figure 3: Model ability to generate images at different epochs during the training for the MNIST and FGVC-Aircraft Datasets.

2.3 Model Training

In the training phase, the U-Net is conditioned to predict the noise of an image $\mathbf{x}_t$ at time step t of the diffusion process. Each iteration involves corrupting a randomly sampled image $\mathbf{x}_0$ from the dataset with Gaussian noise, proportional to the current time step t. Since it is possible to sample the forward distribution $q(\mathbf{x}_t|\mathbf{x}_0)$ at arbitrary time steps, the training is simplified by predicting the noise for a uniformly sampled time step t in each iteration instead of going through all time steps $0, \ldots, T$. In the next step, the model predicts the noise for the corrupted image, which is compared to the actual noise in order to calculate the loss. Finally, the gradient of the loss is used to improve the model and the process is repeated until it converges.

Algorithm 1: Training

```
1 repeat
2     $\mathbf{x}_0 \sim q(\mathbf{x}_0)$;
3     $t \sim \text{Uniform}(\{1, \ldots, T\})$;
4     $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
5     Take gradient step on $\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$;
6 until converged;
```

We included this algorithm in our implementation with adjusted training parameters (documented in table 1) depending on the dataset. Further, we used the *SmoothL1* loss function (Som (2023)) instead of the MSE loss since it yieled better results in the image generation.

	MNIST	FGVC-Aircraft
Number of Images	60.000	10.000
Training Epochs	15	20
Batch Size	32	32
Time Steps T	300	500
Scheduler	linear $[10^{-4}, 0.02]$	linear $[10^{-4}, 0.02]$
Loss Function	SmoothL1	SmoothL1
Optimizer	ADAM	ADAM
Learning Rate	$2 \cdot 10^{-4}$	$2 \cdot 10^{-4}$
Dropout Probability	0.01	0.01

Table 1: The configured training parameters for the MNIST and FGVC-Aircraft datasets.

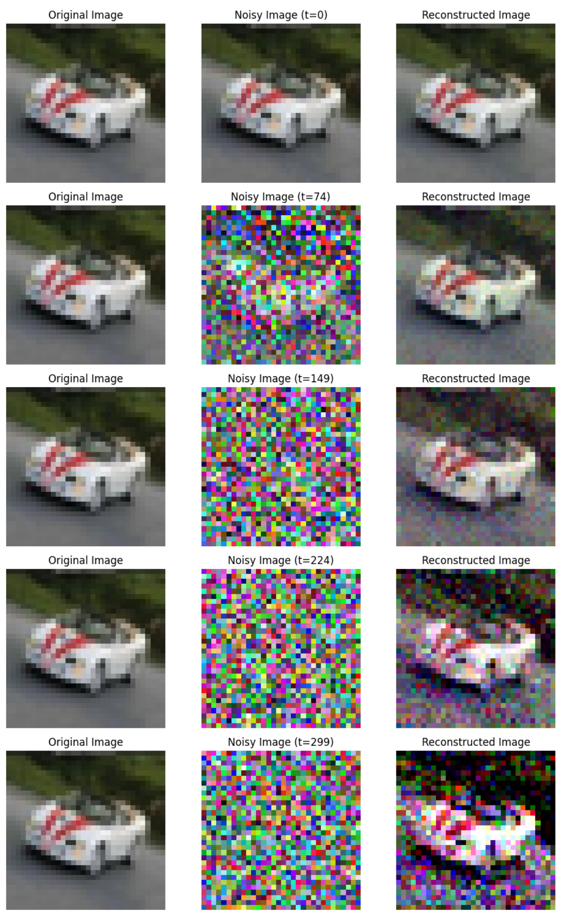

Figure 4: Reconstructed images from noisy samples based on the CIFAR-10 dataset at different time steps of the forward diffusion process.

2.4 Sampling

After training the noise prediction model, we can use it to generate new images that are not contained in the original dataset. First, pure noise $\mathbf{x}_T$ is sampled from a Gaussian distribution. Then, the sample is moved closer to the actual data distribution gradually by denoising $\mathbf{x}_t$ for each time step $t = T, \ldots, 1$ using the model's predictions. The process halts when the generated sample $\mathbf{x}_0$ is obtained.

Algorithm 2: Sampling

```
1 $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
2 for $t = T, \ldots, 1$ do
3     $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$ else $\mathbf{z} = \mathbf{0}$;
4     $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}t}} \boldsymbol{\epsilon}_\theta\left(\mathbf{x}_t, t\right) \right)$;
5     $\mathbf{x}_{t-1} = \boldsymbol{\mu}_\theta(\mathbf{x}_t, t) + \sigma_t \mathbf{z}$;
6 end
7 return $\mathbf{x}_0$
```

To introduce variability and prevent the results from becoming overly deterministic, a small amount of Gaussian noise is added back to $\mathbf{x}_{t-1}$ before repeating the loop. Other parameters such as the number of total diffusion steps T and the applied variance scheduler also affect the quality and diversity of the generated images. We plan to fine-tune those parameters through experimentation in the next milestone.

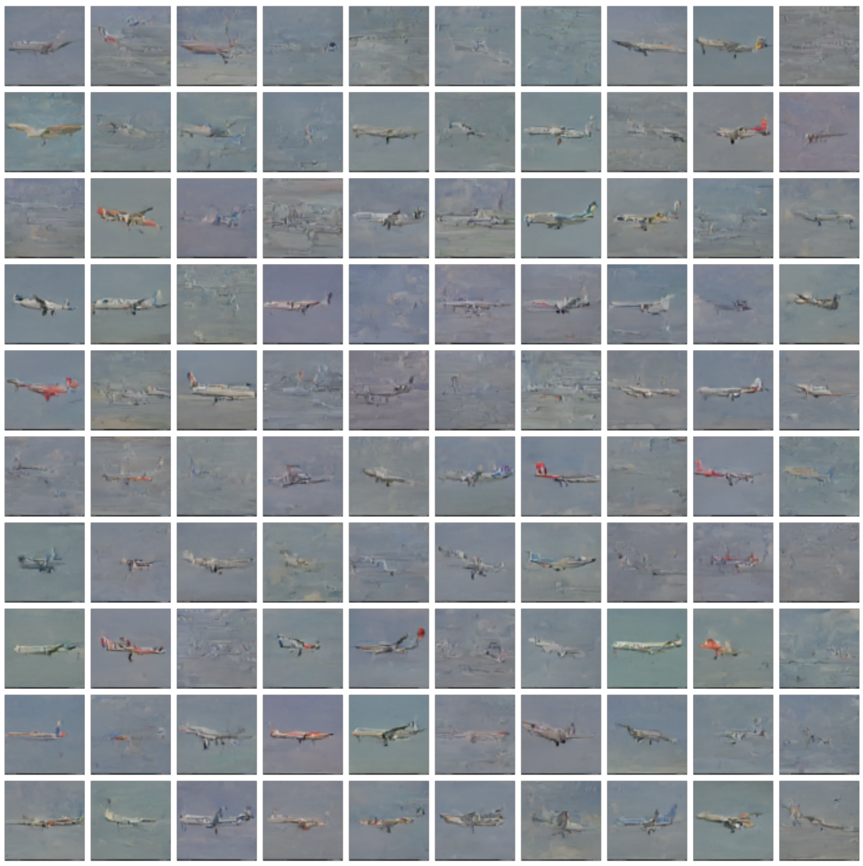
Figure 5: Sampled images based on the FGVC-Aircraft dataset.

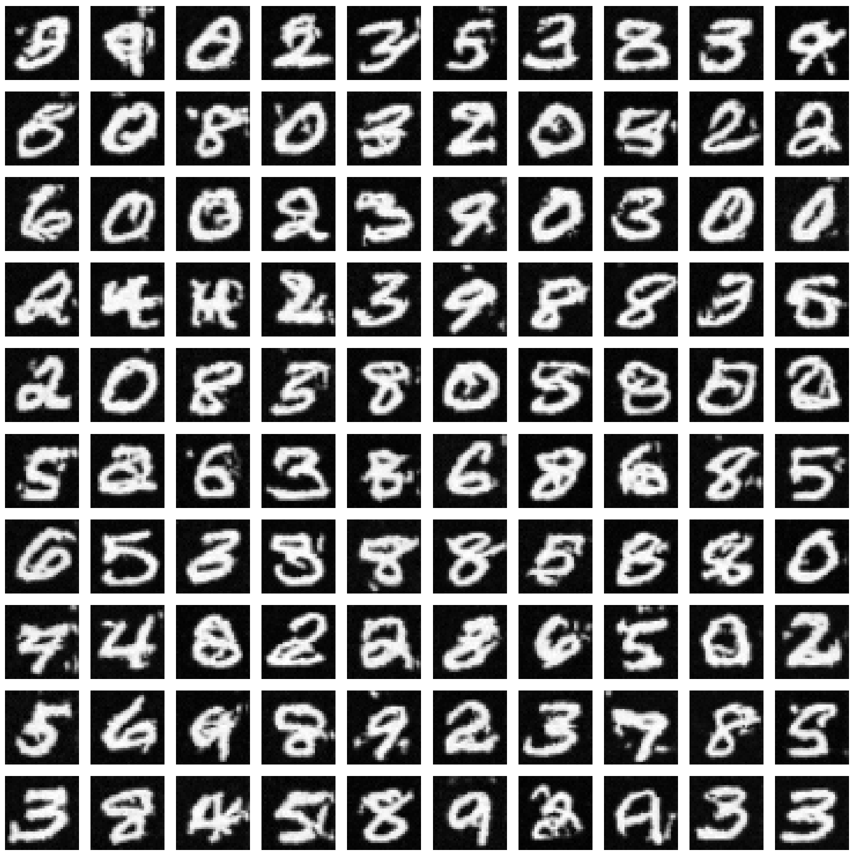
Figure 6: Sampled images based on the MNIST dataset.

3 Evaluation

In this section we present the generated images of our model, introduce the *Inception Score* as the metric for our evaluation and discuss our results.

3.1 Inception Score

The *Inception Score* (IS), originally developed for evaluating General Adversial Networks (GANs) by Salimans et al. (2016), provides a measure to automatically evaluate the quality of image generative models. This is done utilizing the InceptionV3[5] model by Szegedy et al. (2015), which is a convolutional neural network trained on the ImageNet [6] dataset. It measures two properties: The *Quality*, which is the "goodness" of the generated images, i.e. images that are classifiable by the Inception network with a high confidence; and the *Diversity*, which is the variety of images across all different classes.

In adaptation of the mathematical notation of Barratt and Sharma (2018), we define the formula for the Inception Score as follows:

Let Ω_X be the space of images and Ω_Y the (finite) space of labels. We evaluate the probability distribution of p_G over Ω_X of a generator G with the formula

$$IS(G) = \exp\left(\mathbb{E}_{\mathbf{x}\sim p_G}\left[D_{KL}(p(y|\mathbf{x})\|p(y))\right] \right) \tag{12}$$

where D_{KL} denotes the Kullback-Leibler divergence between the distributions p and q, $p(y|\mathbf{x})$ is the conditional class distribution and $p(y) = \int_{\mathbf{x}} \mathbf{x} p(y|\mathbf{x}) p_G(\mathbf{x})$ is ginal class distribution. Since the true marginal distribution is unknown, we approximate it empirically using the samples $\mathbf{x}^{(i)}$:

$$\hat{p}(y) = \frac{1}{N}\sum_{i=1}^{N} p(y|\mathbf{x}^{(i)}) \tag{13}$$

Thus, the estimator for the Inception Score becomes:

$$IS(G) \approx \exp\left(\frac{1}{N}\sum_{i=1}^{N} D_{KL}\left(p(y|\mathbf{x}^{(i)})\|\hat{p}(y)\right) \right) \tag{14}$$

If the generative model satisfies both the traits Diversity and Quality, the KL-divergence between $p(y)$ and $p(y|\mathbf{x})$ grows larger, resulting in a higher score.

The Inception Score can be interpreted as a measure between the images generated by G and ginal class distribution over y and is bound by $1 \leq IS(G) \leq 1000$ (Barratt and Sharma (2018)). A higher score corresponds to more realistic images, and it has been shown to correlate well with human scoring based on the CIFAR-10[7] dataset (Salimans et al. (2016)).

3.2 Results

The generated images depicted in figures 5 and 6 show that our diffusion model is able to successfully generate images with reduced noise. However, a significant gap persists between these outputs and the fidelity of the original models by Ho et al. (2020). This may be attributed to the training time of the models, the fixed selection of hyper-parameters and the low resolutions imposed by hardware limitations.

Images sourced from the FGVC-Aircraft dataset (fig. 5) exhibit a diverse range of outcomes. While certain images maintain a convincing appearance even at lower resolutions, others fall short in accurately depicting the airplane. This disparity may arise from the fact that airplanes typically occupy approximately 30% of the image, leading the model to potentially emphasize pixel generation for sky regions rather than focusing on rendering the aircraft. We will investigate this further in milestone 3, particularly

[5] `https://huggingface.co/docs/timm/models/inception-v3`
[6] `https://www.image-net.org/`
[7] `https://www.cs.toronto.edu/~kriz/cifar.html`

through experimentation with scenery-based image datasets such as LSUN. The images generated based on the MNIST dataset (fig. 6) exhibit recognizable numerical patterns, effectively emulating the intended numbers. Yet, a portion of these images suffers from blurriness and distortions, deviating from the desired clarity and accuracy.

Our observations are also partially confirmed by the Inception Scores of 1.795 and 2.076 for two generated batches based on the FGVC-Aircraft dataset, which are substantially lower compared to the score of 9.46 for the generated images by Ho et al. (2020) based on the unconditional CIFAR-10 dataset. We want to point out that the comparison is a bit flawed since FGVC-Aircraft is a single-class dataset, which naturally results in lower diversity of objects of the generated results and therefore in a lower overall inception score in contrast to models trained on the CIFAR-10 dataset containing 1000 classes of objects. Hence, we plan to asses the capabilities of our model more accurately by training our model more diverse datasets in the next milestone.

4 Conclusion

In this milestone, we implemented the reverse process and sampling procedure and trained our updated U-Net for two different datasets. Our first generated results, while partially blurry and inconsistent, depict plausible images that spark our interest for improvements. For the next and last milestone, we will emphasize the refinement and optimization of the training process in order to enhance the quality and diversity of the images. We will integrate the different schedulers presented in our first report, and perform a model selection by fine-tuning its hyperparameters. Additionally, we will analyze differences between generated images and the original dataset, conducting more evaluations for deeper insights and improvements.

References

S. T. Barratt and R. Sharma. A note on the inception score. *ArXiv*, abs/1801.01973, 2018. URL `https://api.semanticscholar.org/CorpusID:38384342`.

W. Feller. *An Introduction to Probability Theory and Its Applications, Volume II.* John Wiley & Sons, 1966.

P. Hanjin. Denoising diffusion probabilistic models. `https://paramhanji.github.io/posts/2021/06/ddpm/`, 2021. Accessed: 2024-01-05.

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf`.

A. Q. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016. URL `http://arxiv.org/abs/1606.03498`.

V. Singh. An in-depth guide to denoising diffusion probabilistic models – from theory to implementation. `https://learnopencv.com/denoising-diffusion-probabilistic-models/#Writing-DDPMs-From-Scratch-In-PyTorch`, 2023. Accessed: 2024-01-05.

Som. Understanding l1 and smoothl1loss, Aug 2023. URL `https://medium.com/mlearning-ai/understanding-l1-and-smoothl1loss-f5af0f801c71`.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. URL `http://arxiv.org/abs/1512.00567`.