Diffusion model

Tobias Skipevåg Oftedal *Tobiasof@stud.ntnu.no*
Norwegian University of Science and Technology,
Department of Computer Science,
NO-7034 Trondheim, Norway

Ramtin Forouzandehjoo Samavat *Ramtinfs@stud.ntnu.no*
Norwegian University of Science and Technology,
Department of Computer Science,
NO-7034 Trondheim, Norway

Jeffrey Yaw Annor Tabiri *Jytabiri@stud.ntnu.no*
Norwegian University of Science and Technology,
Department of Computer Science,
NO-7034 Trondheim, Norway

Abstract

Diffusion models are generative models that learn to generate samples by gradually adding Gaussian noise to data and then learning to reverse that process. This paper presents a conditional diffusion model[1]trained on the MNIST dataset, utilizing a U-Net architecture to predict Gaussian noise added to images. The model incorporates classifier-free guidance, enhancing sample quality by training both conditional and unconditional models simultaneously. The main goal of this study is to provide insight into the effects of tuning a model trained on a low-resolution dataset. The model achieved optimal performance when trained with a linear scheduler for 1000 timesteps and a guidance weight of 0.2, resulting in a FID score of 16 and an Inception score of approximately 9.6.

[1]Github project: https://github.com/RamtinS/diffusion-model/tree/main

1 Introduction

In recent years, diffusion models have gained a lot of popularity for their ability to generate high-quality samples, such as images. These generative models learn to produce data by gradually adding Gaussian noise to structured data, making it more chaotic, and then learning to reverse that process to recover the original structure. Their design is inspired by concepts from non-equilibrium thermodynamics (Sohl-Dickstein et al., 2015). They can be used independently or as part of more complex models, such as in DALL-E, a text to image model (Ramesh et al., 2022).

This paper presents an implementation of a conditional diffusion model, discusses the necessary adjustments to enhance the image quality, and evaluates the model's ability to achieve high-quality samples. It seeks to answer the question: What is the optimal noise scheduler, guidance weight and timestep for the given diffusion model? The dataset used for training and evaluation consists of low-resolution images of handwritten digits from the MNIST dataset (Deng, 2012).

2 Related Work

Numerous studies have explored the topic of diffusion models, addressing their challenges and possibilities. The earliest research paper on diffusion models was by Sohl-Dickstein et al. (2015), which laid the theoretical foundation for diffusion probabilistic models.

Building on this foundation, Ho et al. (2020) introduced significant advancements, presenting a practical approach for generating high-quality samples. Their approach utilized the U-Net architecture, originally introduced by Ronneberger et al. (2015), to predict the noise added to an image at each time step. They also introduced a linear noise scheduler to gradually increase the noise added to the data.

In 2021, researchers at OpenAI published a paper where they introduced improvements to the implementation presented by Ho et al. (2020). Among other improvements, they introduced the cosine scheduler as a replacement for the linear scheduler to improve log likelihood (Nichol & Dhariwal, 2021).

Classifier guidance is a method used to enhance the sample quality of diffusion models by using a trained classifier. This approach replicates the effect of "low-temperature" sampling, seen in flow-based models like Glow and the truncated models like BigGAN. In 2022, researchers from Google took this idea further, and showed that guidance can be achieved without a trained classifier, in a method known as classifier-free guidance (Ho & Salimans, 2022).

3 Methods

This section outlines the methodologies employed to implement a diffusion model. The model consists of two processes, the forward diffusion process, which transforms a clear image to a noised state, and the reverse diffusion process, which reconstructs the original image from the noised form.

3.1 Forward Diffusion Process

The forward diffusion process uses a Markov chain to gradually add noise to the image for each step. A Markov chain is a mathematical process used to represent a sequence of states, where the probability of each future state solely depends on its preceding state. The process of getting the next image in the chain, was formulated by Ho et al. (2020), where β_t denotes the variance of the added Gaussian noise, determined by a noise scheduler.

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

$$x_t = \sqrt{1-\beta_t} \cdot x_{t-1} + \sqrt{\beta_t} \cdot \varepsilon_t \tag{2}$$

This equation can be simplified by introducing the variable alpha, defined as $\alpha_t = 1 - \beta_t$ and the cumulative product of alpha, defined as $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$ to find the noised image directly instead of iteratively (Ho et al., 2020).

$$q(x_t \mid x_0) := \mathcal{N}\left(x_t; \sqrt{\bar{\alpha}_t} \cdot x_0, (1-\bar{\alpha}_t) \cdot I\right) \tag{3}$$

$$x_t = \sqrt{\bar{\alpha}_t} \cdot x_0 + \sqrt{1-\bar{\alpha}_t} \cdot \varepsilon_t \tag{4}$$

Figure 1: Linear noising process for T = 1000

3.2 Reverse Diffusion Process

The reverse diffusion process uses a Markov chain along with a convolutional neural network to iteratively predict and remove noise from an image. After a predefined number of t iterations, it returns an approximation of the original image. The process can be expressed mathematically as follows, where $\mu_\theta(x_t,t)$ is learned and denotes the predicted mean of the image for each time step, and $\Sigma_\theta(x_t,t)$ which is fixed and denotes the predicted variance (Ho et al., 2020):

$$p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1};\mu_\theta(x_t,t),\Sigma_\theta(x_t,t)) \tag{5}$$

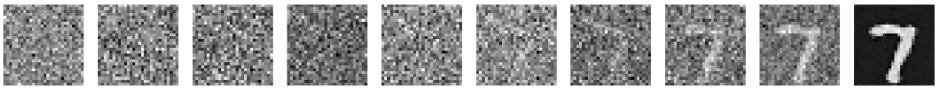

Figure 2: Reverse process for T = 1000

3.2.1 U-Net Architecture

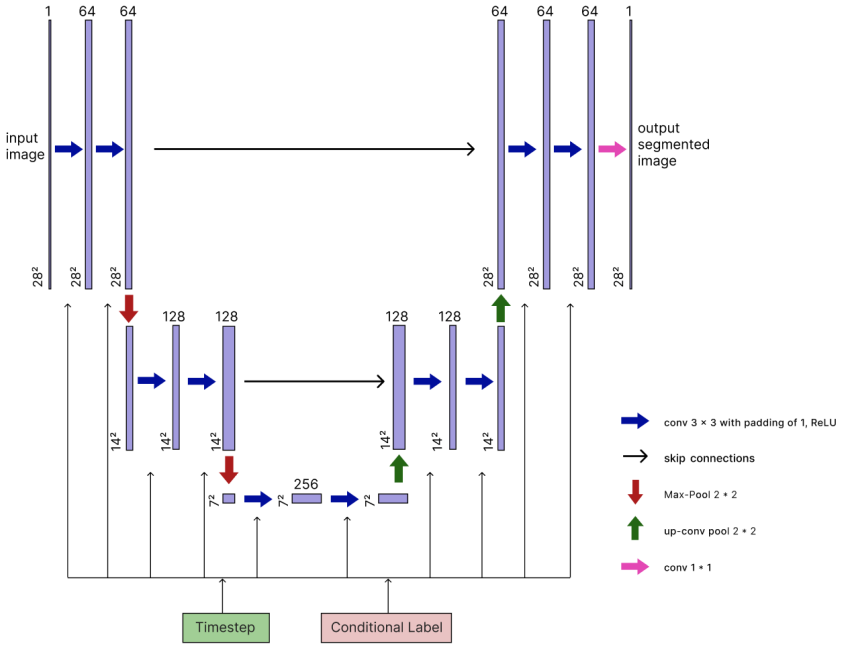

Figure 3: U-Net illustration

The neural network employed in the reverse diffusion is called a U-Net, named after its U-shaped architecture. The U-Net consists of the contracting path (left side) and the expanding path (right side) connected by a bottleneck. In the original implementation introduced by Ronneberger et al.

(2015) the architecture consists of four blocks in each section. However, due to the lower resolution of the MNIST dataset images, the architecture is downsized to two blocks.

The blocks and bottleneck contains two 3x3 convolutions with padding 1, followed by ReLU activations. In the contracting path, a 2x2 max pooling with stride 2 is applied between each block, reducing the feature maps by half while doubling the number of channels. In the expanding path, a transposed convolution is used between blocks to increase resolution of feature maps. At the end of the path, a 1x1 convolution is applied to map the output channels to the desired number of classes. Before passing through each block in the expanding path, the output of the corresponding block in the contracting path is concatenated with the output within in the expanding path to preserve important details. The time embedding and conditional labels are integrated into each block of the network, helping the model generate output based on time-dependent data and specific labels.

3.3 Classifier Free Guidance

Classifier guidance improves sample quality by using a trained classifier (Dhariwal & Nichol, 2021). However, it presents challenges, including the need to train a classifier, which complicates the training pipeline and introduces a dependency on an accurate classifier. To address these complications, researchers from Google introduced classifier-free guidance, where conditional and unconditional diffusion models are trained together to achieve similar results without requiring an external classifier (Ho & Salimans, 2022).

The unconditional model is parameterized through $\varepsilon_\theta(x_t, t)$, while the conditional model is parameterized through $\varepsilon_\theta(x_t, t, c)$. By randomly setting the class identifier c to zero, it is possible to train the unconditional and conditional model within the same neural network. During sampling, a linear combination of the two models is used, with ω controlling the guidance strength (Weng, 2021):

$$\tilde{\varepsilon}_\theta(x_t, t, c) = (1+\omega) \cdot \varepsilon_\theta(x_t, t, c) - \omega \cdot \varepsilon_\theta(x_t, t) \tag{6}$$

3.3.1 Training

Algorithm 1 Training

repeat
  $x_0, c \sim$ Dataset ▷ Sample input and label from dataset
  $t \sim \text{Uniform}(\{1, \ldots, T\})$ ▷ Randomly select a timestep
  $\varepsilon \sim \mathcal{N}(0, I)$ ▷ Sample Gaussian noise
  $x_t = \sqrt{\bar{\alpha}_t} \cdot x_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \varepsilon$ ▷ Generate noisy input
  $\mathscr{L}_{L_1} = \mathbb{E}_{x_t, t, \varepsilon} \left[\| \varepsilon - \varepsilon_\theta(x_t, t, c) \|_1 \right]$ ▷ Determine the loss
  $\nabla_\theta \mathscr{L}_{L_1}$ ▷ Calculate the gradient of the loss
  Take gradient step on $\nabla_\theta \mathscr{L}_{L_1}$
until converged

Algorithm 1 presents a simplified version of the training procedure. The training is performed using batches of 32 images as input.

3.4 Sampling

Algorithm 2 Conditional sampling

$x_T \sim \mathcal{N}(0, I)$ ▷ Sample random noise image
for $t = T, \ldots, 1$ **do**
  $\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t \cdot \varepsilon_\theta(x_t, t)}{\sqrt{1 - \bar{\alpha}_t}} \right)$ ▷ Determine prediction without conditioning
  $\mu_\theta(x_t, t, c) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t \cdot \varepsilon_\theta(x_t, t, c)}{\sqrt{1 - \bar{\alpha}_t}} \right)$ ▷ Determine prediction with conditioning
  $\tilde{x}_t = (1 + \omega) \cdot \mu_\theta(x_t, t, c) - \omega \cdot \mu_\theta(x_t, t)$ ▷ Determine guided prediction
  if t == 0 **then**
    $x_{t-1} = \tilde{x}_t$ ▷ Set predicted image as output (final image)
  else
    $\sigma_t^2 = \frac{\beta_t \cdot (1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t}$ ▷ Determine variance of current time step
    $\varepsilon \sim \mathcal{N}(0, I)$ ▷ Sample noise from a standard normal distribution
    $x_{t-1} = \tilde{x}_t + \sqrt{\sigma_t^2} \cdot \varepsilon$ ▷ Add noise to predict the previous time step
  end if
end for
return x_0 ▷ Return final denoised image

Algorithm 2 presents a simplified version of the sapling procedure.

3.5 Evaluation

The inception score (IS) is commonly used to evaluate diffusion models. The goal is to give a score that correlates well with the human perception of images, as well as ensuring image diversity. The main part of the evaluation involves taking the expected value of the Kullback & Leibler (1951)

divergence between the predicted class distribution $p(y \mid \mathbf{x})$ and the marginal class distribution $p(y)$ (Salimans et al., 2016).

$$\text{IS} = \exp\left(\mathbb{E}_{\mathbf{x}} \mathbf{KL}(p(y \mid \mathbf{x}) \| p(y))\right) \tag{7}$$

The IS is high when the conditional distribution is peaked (indicating confident predictions) and the marginal distribution is spread (signifying diverse images). This ensures that models generating both clear and diverse images receive a high score.

The standard implementation uses the Inception-v3 model, trained on ImageNet dataset, to calculate image labels. However, since its classes do not overlap well with all datasets, such as MNIST, this can result in high entropy in the $p(y \mid \mathbf{x})$ distribution, leading to artificially low IS scores.

To address this, a custom CNN classifier pre-trained on MNIST images was used instead of Inception-V3. It is a simple CNN model, consisting of two pooling and convolution layers as well as one linear. Despite being such a simple model, it achieves about 99% accuracy, even outperforming an Inception-v3 model retrained using transfer learning (Colianni, 2017). While this evaluation may not be directly comparable to other papers using the Inception-v3 model, it follows the same procedure and helps evaluating the model in search for optimal hyperparameters.

Another commonly used metric to evaluate diffusion models is the Fréchet Inception Distance (FID). Similarly to IS, it uses the same inception model, but instead of relying on class probability outputs, it compares the image representations in the last pooling layer by calculating the Frechét distance. This puts focus on the features of the images rather than what they look like directly. This makes FID less susceptible to errors when the dataset differs from ImageNet. FID has also been shown to more reliably reflect the remaining noise in the image compared to IS (Heusel et al., 2018).

While FID is better at evaluating the distribution of generated samples, IS better reflects how realistic the samples appear to humans. Optimally, both the IS and FID should be evaluated using around 50.000 images (Heusel et al., 2018; Salimans et al., 2016)

4 Results and Discussion

The evaluation of the diffusion model was conducted by measuring both FID and IS with different hyperparameters. FID was evaluated using an existing library (Seitzer, 2020) by creating 10.000 images and comparing them to the 10.000 images from the MNIST test dataset. IS was also evaluated using approximately the same amount of images.

Due to hardware limitations, generating 50.000 images was deemed too time-consuming. Our hypothesis was that since the images are of lower resolutions and consists of only 10 classes, the scores would give the same result on a smaller amount of images. As shown in table 1, running the same FID procedure on a trained model 5 times, resulted in a calculated standard deviation of only 0.11, signaling that 10.000 images was adequate.

Iters	**T**	**Schedule**	**Guidance weight (ω)**	**FID**
100K	1000	linear	0.2	16.19
100K	1000	linear	0.2	16.05
100K	1000	linear	0.2	15.97
100K	1000	linear	0.2	16.09
100K	1000	linear	0.2	15.92
Mean ± Std. Deviation				**16.04 ± 0.11**

Table 1: Mean and standard deviation of FID calculated 5 times on the same instance of the model

Iters	**T**	**Schedule**	**Guidance weight (ω)**	**FID**	**IS**
100K	1000	linear	0.2	**16.19**	**9.56**
100K	4000	linear	0.2	20.91	8.68
100K	1000	cosine	0.2	26.82	9.00
100K	4000	cosine	0.2	21.70	8.73

Table 2: Training Configurations

Iters	**T**	**Schedule**	**Guidance weight (ω)**	**FID**	**IS**
100K	1000	linear	0	21.12	9.13
100K	1000	linear	0.2	16.19	9.56
100K	1000	linear	0.5	24.67	9.14

Table 3: Different guidance weight

Considering the limited resources available for training and testing, the results shows that the implemented diffusion model demonstrates promising performance in generating high-quality samples. The results in Table 2 indicate that the model achieves its best performance with 1000 timesteps, a guidance weight of 0.2 and using a linear scheduler.

The linear scheduler achieves better performance with 1000 timestep rather than 4000. This outcome might arise due to the increased complexity introduced by high values of timesteps, leading the model to mimic the distribution of the added noise rather than the distribution of the actual dataset. This theory builds on research conducted by Song & Kingma (2021).

Furthermore, it can be observed that the cosine scheduler performs better with more timesteps. The cosine scheduler interpolates smoothly, allowing the model to retain details of the original dataset. As a result, the model can learn the actual data distribution more accurately.

The observations in Table 3 suggest that a stronger guidance weight may lead to worse FID results while slightly improving the IS. On the other hand, a lower guidance weight may result in better FID but with a slight decrease in IS. A balanced combination of both, however, may improve both metrics. This can be explained by the fact that when the guidance weight is set to zero, the predicted noise is determined solely by the conditional network. As a result, causing the generated samples to be overly similar and thus reducing diversity. However, if the weight is increased to more extreme values, the network will rely more on the unconditional information, which may introduce characteristics from other numbers, leading to a broader range of generated numbers and greater diversity in the generated samples.

4.1 Comparisons

In Table 4, we compare our model's performance against what is considered the current state-of-the-art generative models (Papers With Code, 2024) trained on the MNIST dataset. While our model generates samples of reasonably high quality, it still has a long way to go to match the performance of other models.

Model	**FID**
Sliced Iterative Generator	4.5
HypGAN	7.87
JKO-iFlow	7.95
Guided Diffusion (**Ours**)	16.19
Spiking-Diffusion	27.61
PresGAN	38.53

Table 4: Comparison of generative models on the MNIST dataset.

5 Conclusion and Future Work

In conclusion, this paper introduced an implementation of a conditional diffusion model for generating high-quality samples from the MNIST dataset. From the results, it can be concluded that the optimal combination of configurations was achieved by using a linear scheduler with a timestep of 1000 and a guidance weight of 0.2.

One parameter subject to future work is the complexity of the dataset, as creating images of higher resolution and with a larger color space adds increased difficulty for the model. With more training time it can be explored if the current choice of U-Net architecture and hyperparameters are sufficient to effectively handle the added complexity.

References

Stuart Colianni. How Good is Inception-v3 at MNIST?, 2017. URL https://www.kaggle.com/code/scolianni/how-good-is-inception-v3-at-mnist.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021. URL https://arxiv.org/abs/2105.05233.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018. URL https://arxiv.org/abs/1706.08500.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. URL https://arxiv.org/abs/2207.12598.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 1951.

Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models, 2021. URL https://arxiv.org/abs/2102.09672.

Papers With Code. Mnist benchmark (image generation), 2024. URL https://paperswithcode.com/sota/image-generation-on-mnist. Accessed: 2024-11-17.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022. URL https://arxiv.org/abs/2204.06125.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. URL https://arxiv.org/abs/1505.04597.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016. URL https://arxiv.org/abs/1606.03498.

Maximilian Seitzer. pytorch-fid: FID Score for PyTorch. https://github.com/mseitzer/pytorch-fid, August 2020. Version 0.3.0.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015. URL https://arxiv.org/abs/1503.03585.

Yang Song and Diederik P. Kingma. How to train your energy-based models, 2021. URL https://arxiv.org/abs/2101.03288.

Lilian Weng. What are diffusion models? *lilianweng.github.io*, Jul 2021. URL https://lilianweng.github.io/posts/2021-07-11-diffusion-models/.

A Generated images

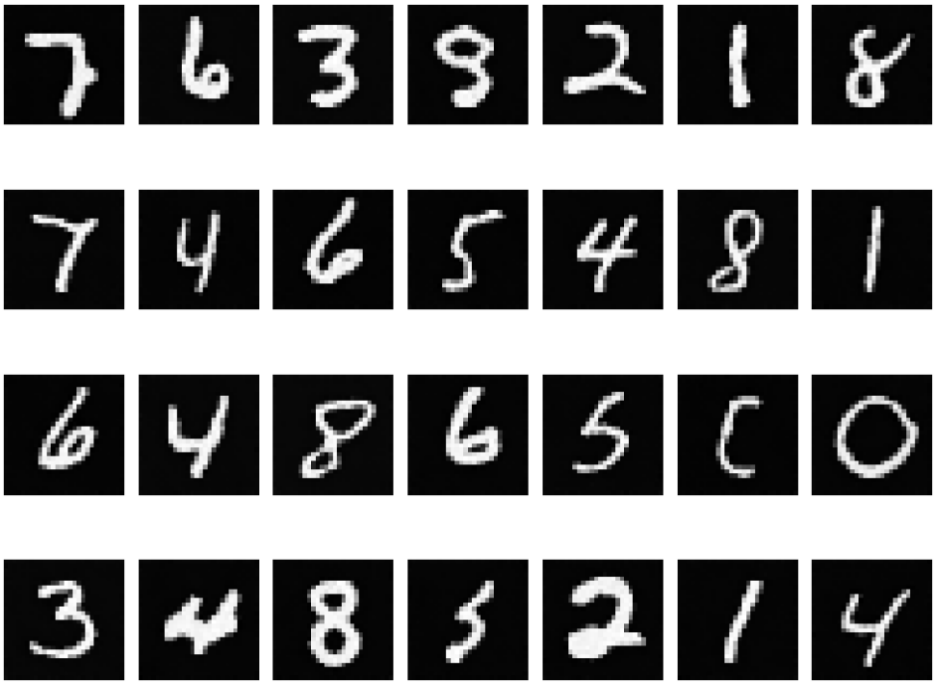

Figure 4: Images generated from model using a linear scheduler with 1000 timesteps and a guidance weight of 0.2