Deep Learning Project: Denoising Diffusion Probabilistic Models

First Author
Marija Brkic

Second Author
Mina Goranovic

Abstract

The following work presents the topic of Denoising Diffusion Probabilistic Models (we will call them Diffusion Models for simplicity). Diffusion models are a type of generative model aimed at generating high-quality data. A diffusion model can be represented as a sequence of time states forming a Markov Chain, over which a forward diffusion process and a backward diffusion process can be defined. The forward process involves the iterative noising of input data in order to approximate white Gaussian noise as closely as possible, while the backward process attempts to reverse the noising procedure and, using variational inference, arrive at the posterior distribution of the input data.

To estimate the parameters of the mentioned posterior distribution, we use a U-Net architecture, and we provide an explanation of the theoretical basis of diffusion models, as well as the training process and its results.

1 Introduction

Diffusion models are inspired by thermodynamic systems out of equilibrium (Weng, 2021). In thermodynamic systems out of equilibrium, a diffusion process occurs, during which molecules move from areas of lower concentration to areas of higher concentration in order to achieve equilibrium. Inspired by such processes, the concept of Probabilistic Diffusion Models for Denoising was formed.

Diffusion models can be represented as a Markov chain of diffusion steps over which forward and backward processes can be defined. In the forward process, noise is incrementally added to the data, while in the backward process, the model is trained to reverse the diffusion process in order to generate the desired data from the noise.

Diffusion models belong to the group of generative models, whose goal is to generate high-quality data (Ho et al., 2020; Chan, 2024).

2 Related Work

Denoising diffusion probabilistic models (DDPMs) have emerged as a powerful framework for generative modeling, introduced by Ho et al. (Ho et al., 2020), where image generation is approached via a learned denoising process. Recent tutorials such as Chan (Chan, 2024) and Weng (Weng, 2021) provide accessible explanations and theoretical insights into the mechanisms of diffusion models, emphasizing their applications in vision tasks, such as image generation.

Despite their success, diffusion models remain computationally intensive and often require careful architectural choices and training strategies. This motivates our work: to better understand and explore the architectural components and evaluation metrics of diffusion models.

3 Diffusion Models Structure

The structure of diffusion models is a sequence of states x_0, x_1,...,x_T that could be represented with Markov Chain which means that the current state depends only on the previous one:

$$p(x_t|x_{0:t-1}) = p(x_t|x_{t-1}) \tag{1}$$

where:

- x_0 is a input data (image)
- x_T is a latent variable that we would like to be Gaussian noise $x_T \sim N(0, I)$
- $x_1, x_2, ..., x_{T-1}$ are also latent variables that are not Gaussian noise

The structure of diffusion model could be shown on the next image:

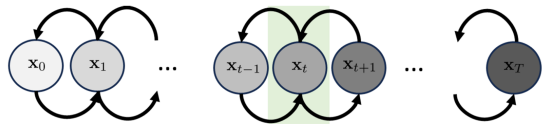

Figure 1: Structure of Diffusion Model(taken from (Chan, 2024))

A transitional block centered at time t consists of three states: x_{t-1}, x_t, and x_{t+1}. By observing the structure of the model, it can be concluded that state x_t can be reached in two ways: from state x_{t-1} (forward transition/forward process) and from state x_{t+1} (backward transition/backward process).
The forward transition occurs from state x_{t-1} to state x_t. For this reason, it is necessary to define a transition model, that is, a conditional probability density function $p(x_t|x_{t-1})$. This distribution is unknown and will therefore be approximated by a Gaussian distribution $q_\Phi(x_t|x_{t-1})$.
The backward transition occurs from state x_{t+1} to state x_t. This transition model, i.e., the probability density function $p(x_t|x_{t+1})$, is also unknown and will be approximated by a Gaussian distribution $p_\Theta(x_t|x_{t+1})$, with the mean of this distribution estimated by a neural network, which will be discussed later.

4 Forward Process

It has already been mentioned that the transition model form the state x_{t-1} to the state x_t could be defined with the Gaussian distribution $q_\Phi(x_t|x_{t-1})$:

$$q_\Phi(x_t|x_{t-1}) = N(x_t|\sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)I) \quad (2)$$

where:

- I is an identity matrix
- α_t is a scalar that controls the change of expectation and covariance matrix form one state to another
- $\sqrt{\alpha_t}x_{t-1}$ is an expectation of conditional probability distribution function
- $(1-\alpha_t)I$ is a covariance matrix of conditional probability distribution function

The whole idea of the forward process is to have a controlled noising of the input data x_0 with the goal to have a distribution $N(0, I)$ as a final state(Chan, 2024).

We can also show that in the noising process we could 'jump' from the state x_0 to a state x_t directly, without the iterative process:

$$q_\Phi(x_t|x_0) = N(x_t|\sqrt{\overline{\alpha}_t}x_0, (1-\overline{\alpha}_t)I) \quad (3)$$

where $\overline{\alpha}_t = \prod_{i=1}^{t} \alpha_i$(Chan, 2024).

This way we defined the complete forward process, and now we will implement it on the actual data. We will be using the MNIST dataset that contains 70000 images of handwritten digits. We will split it on 50000 images in training dataset, 10000 images in validation dataset and 10000 images in testing dataset. All of them are grayscale images, dimension 28x28, and we will be rescaling them to the range [-1,1] as shown in the paper (Weng, 2021). Some of the image examples could be shown on the next image:

Figure 2: Examples from MNIST dataset

After that we can add noise to images in the previously explained way. We will be using $\beta_t = 1-\alpha_t$ that grows linearly 0.0001 do 0.02 (Ho et al., 2020), for t that has values $[1, 300]$. We show states $t = 20, 50, 100, 300$, so we can observe the way the noise is iteratively and controllably added to original images:

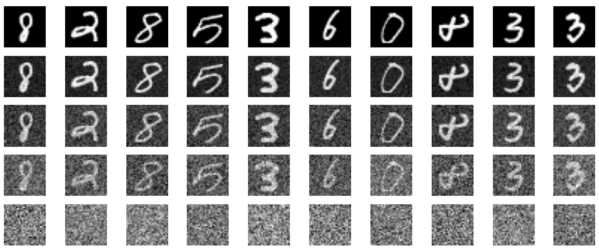

Figure 3: Noising of the MNIST dataset

5 Backward Process

In order to train our diffusion model we need a loss function, and since this model is based on variational inference, we want to use evidence lower

bound as a loss function. We could present evidence lower bound of the diffusion model as:

$$\begin{aligned}\text{ELBO}_{\Phi,\Theta}(x) = \mathbb{E}_{q_\Phi(x_1|x_0)}\Big[\log \underbrace{p_\Theta(x_0|x_1)}_{\text{initial transition block}}\Big] \\ - \mathbb{E}_{q_\Phi(x_{T-1}|x_0)}\Big[\underbrace{D_{\text{KL}}\Big(q_\Phi(x_T|x_{T-1}) \parallel p(x_T)\Big)}_{\text{last transition block}}\Big] \\ - \sum_{t=1}^{T-1} \mathbb{E}_{q_\Phi(x_{t-1},x_{t+1}|x_0)}\Big[\underbrace{D_{\text{KL}}\Big(q_\Phi(x_t|x_{t-1}) \parallel p_\Theta(x_t|x_{t+1})\Big)}_{\text{transition block}}\Big].\end{aligned} \tag{4}$$

In this case, the intuitive meaning of each of the three components of the evidence lower bound of the diffusion model could be explained.

First component is called reconstruction and refers to the quality of the initial block, i.e., how well the initial image x_0 can be reconstructed from the hidden variable x_1, which is why the first term involves the log-likelihood $p_\Theta(x_0|x_1)$.

Second component is called prior matching and refers to the last building block of the diffusion model. The second component involves the Kullback-Leibler (KL) divergence between the estimated probability density function $q_\Phi(x_T|x_{T-1})$ obtained from the forward process and the desired probability density function at the final block $p(x_T)$, which is $N(0, I)$.

Third component is called consistency and refers to the quality of the estimate of all transition blocks. As already explained, the samples x_t can be obtained in two ways: through the forward distribution $q_\Phi(x_t|x_{t-1})$ and the backward distribution $p_\Theta(x_t|x_{t+1})$, which is why this component involves the KL divergence between these two distributions.

In this way, the evidence lower bound covers the quality assessment of the entire model (Chan, 2024).

After a long calculation we can transform the Evidence Lower Bound into:

$$\begin{aligned}\text{ELBO}_\Theta(x) = -\sum_{t=1}^{T} \mathbb{E}_{q(x_t|x_0)}\Big[\frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2 \overline{\alpha}_{t-1}}{(1-\overline{\alpha}_t)^2} \\ \times ||\hat{\epsilon}_\Theta(x_t) - \epsilon_0||^2\Big]\end{aligned} \tag{5}$$

where:

- ϵ_0 is added noise to the original image for the step t
- $\hat{\epsilon}_\Theta(x_t)$ is estimated noise added to the original image for the step t
- $\sigma_q(t) = \frac{(1-\alpha_t)(1-\sqrt{\overline{\alpha}_{t-1}})}{1-\overline{\alpha}_t}$
- $q(x_t|x_0) = q_\Phi(x_t|x_0)$ only Φ is removed since it is not learned in the network

Now, the training algorithm of the neural network proceeds as follows:

1. A time step is chosen from the distribution $t \sim U(1, T)$.
2. The noisy image x_t is computed from the distribution $N(X_t|\sqrt{\overline{\alpha}_t}x_0, (1-\overline{\alpha}_t)I)$, that is:

$$x_t = \sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}z \tag{6}$$

where:

$$z \sim N(0, I) \tag{7}$$

3. The model is trained using gradient descent:

$$\nabla_\Theta ||\hat{\epsilon}_\Theta(x_t) - \epsilon_0||^2 \tag{8}$$

4. The process is repeated until the loss function converges.

Once the model is trained, inference of the conditional distributions $p_\Theta(x_{t-1}|x_t)$ can be performed for each time step, from which samples $x_T, x_{T-1}, ..., x_1$ can be obtained recursively through the reverse diffusion process as follows:

$$x_{t-1} \sim p_\Theta(x_{t-1}|x_t) = N(x_{t-1}|\mu_\Theta(x_t), \sigma_q^2(t)I) \tag{9}$$

$$= \mu_\Theta(x_t) + \sigma_q^2(t)z \tag{10}$$

$$= \frac{1}{\sqrt{\alpha_t}}x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}\sqrt{\alpha_t}}\hat{\epsilon}_\Theta(x_t) + \sigma_q(t)z \tag{11}$$

$$= \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\hat{\epsilon}_\Theta(x_t)) + \sigma_q(t)z \tag{12}$$

where:

$$z \sim N(0, I) \tag{13}$$

The final inference algorithm in the diffusion probabilistic model when inferring noise is as follows:

1. Start from white Gaussian noise $x_T \sim N(0, I)$.
2. Repeat the process for each $t = T, T-1, ..., 1$.
3. Get the network output $\hat{\epsilon}_\Theta(x_t)$.
4. Obtain the samples x_{t-1} by sampling as follows:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\hat{\epsilon}_\Theta(x_t)) + \sigma_q(t)z \tag{14}$$

where:

$$z \sim N(0, I) \tag{15}$$

This algorithm was proposed in (Ho et al., 2020):

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent 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

Algorithm 2 Sampling

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

Figure 4: Training and Inferencing taken from (Ho et al., 2020)

6 Network Architecture

The type of network trained using the previously explained algorithm is a U-Net (Weng, 2021).

U-Net is a type of autoencoder network that consists of two parts: an encoder and a decoder.

The goal of the encoder is to gradually reduce the image resolution through several encoder blocks, in order to extract the most important features and characteristics. An encoder block consists of several components including convolutional layers, batch normalization, and a MaxPool layer. The convolutional layer is composed of a set of 3x3 kernels that perform a convolution operation on the input data. The elements of these kernels are the parameters that are learned during the training process. Batch normalization ensures that all resulting data are normalized so that no feature becomes overly dominant compared to the others. The MaxPool layer applies a 2x2 kernel that selects the maximum value from every group of four pixels in the image. In this way, the most relevant features are effectively extracted, and the resolution is halved at the end of each encoder block.

The goal of the decoder is to reverse the encoder's process and restore the image to its original resolution while preserving the key features extracted by the encoder. The decoder also consists of decoder blocks with a structure similar to that of encoder blocks. The main difference is that instead of a MaxPool layer, a special convolutional layer is used that increases the resolution of the image (in this case, it doubles it). Furthermore, the input to each decoder block is not just the output of the previous decoder block, but it is also concatenated with the output from the encoder block at the same level. This type of connection is known in the literature as a skip connection. The purpose of the skip connection is to prevent vanishing gradients phenomenon, which relates to the saturation of network parameters. Additionally, it helps retain information during the decoding process by incorporating outputs from the encoder.

Moreover, as the resolution decreases, each subsequent encoder block contains twice as many convolutional kernels as the previous one, while as the resolution increases, each decoder block contains half as many kernels as the previous one. In addition, the ReLU activation function is used in every convolutional layer (Ronneberger et al., 2015).

Additionally, we used two encoder and two decoder blocks, with bottleneck layer, and the reason for smaller architecture than we would usually use is the dimension of images. Because od the dimension 28x28, we could not use four encoder and decoder blocks as it is usually used in U-Net architectures, since the dimension is too small and could not be reduced that many times.

We used the Adam optimizer with an initial learning rate of 10^{-3}. To prevent overfitting, we also implemented early stopping, a technique that monitors the validation loss in addition to the training loss. If the validation loss does not improve for five consecutive epochs, we assume that the model has stopped generalizing effectively, and training is stopped.

The structure of the U-Net can be seen in the following figure:

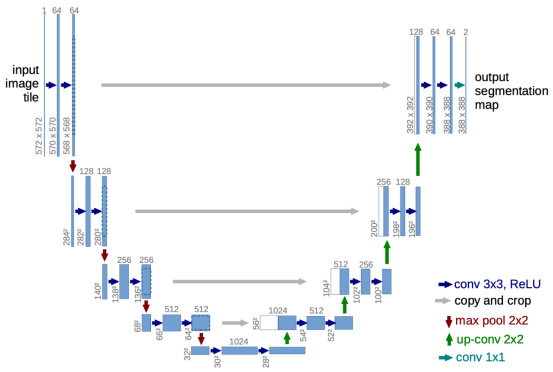

Figure 5: U-Net architecture (taken from (Ronneberger et al., 2015))

7 Results and Evaluation

We trained the model for 50 epochs, but the early stopping was triggered at the epoch 39. We can look at the loss on the training and validation

dataset during the training:

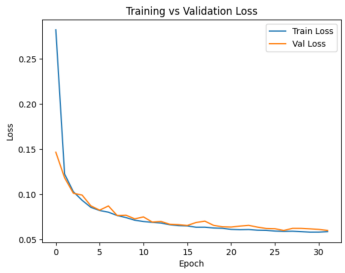

Figure 6: Loss function on training and validating dataset during training

We can notice that both training and validation loss are decreasing, but the early stopping was triggered and the best parameters (with the smallest validation loss) were preserved, which is the point where the previous plot stops.

Finally, after the network is trained, we can do the inverse, iterative sampling process, that represents image generating. The next figure shows the result for some of the images from the training dataset at the time steps $t = 300, 100, 50, 20, 0$:

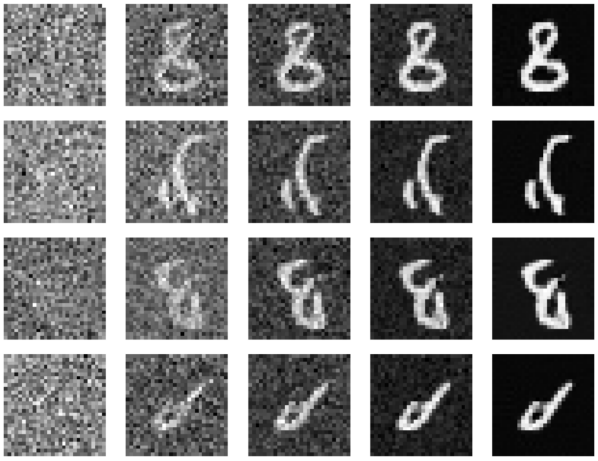

Figure 7: Denoising results

We can notice that all of the generated images are free of any noise. Some of them are quite obviously digits, some a little less, which could potentially be improved with more epochs or different network parameters. However, we are very pleased with the results we achieved.

As for the evaluation metric, the paper (Ho et al., 2020) proposes FID.

Fréchet Inception Distance (FID) is a commonly used metric for evaluating the quality of images generated by generative models. It compares the distribution of generated images to that of real images by modeling their feature representations as multivariate Gaussians. These features are typically extracted from the penultimate layer of a pre-trained Inception network.

Formally, if (μ_r, Σ_r) and (μ_g, Σ_g) are the mean and covariance of the real and generated image features respectively, the FID is defined as:

$$\text{FID} = \|\mu_r - \mu_g\|^2 + \text{Tr}\left(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}\right) \tag{16}$$

A lower FID indicates that the generated images are more similar to the real ones in terms of feature distribution. However, recent work has shown that FID can be sensitive to various factors such as the choice of feature extractor and dataset statistics, motivating the development of improved alternatives (Jayasumana et al., 2024).

For our evaluation we decided to use FID and the result we achieved on the test dataset is 24.6031, which is quite satisfying. Additionally, in order to calculate this metric we had to reshape the data: transfer it to RGB and to the range [0,255].

8 Conclusion

In this work, we presented our implementation of Diffusion Models and found the results to be highly satisfactory. Diffusion Models have demonstrated strong potential in various creative domains, including art, technology, and music. They can complement other generative approaches, such as GANs. However, unlike GANs, diffusion-based generation is significantly slower due to its iterative image generating, that remains an active area of research. Additionally, like all generative models, diffusion models carry the risk of misuse, such as generating fake or misleading content for political or deceptive purposes. This highlights the importance of ethical considerations in the development and deployment of such technologies (Ho et al., 2020).

9 Work Distribution

In this project we both worked on the code and the report at the same time, so the work was evenly distributed.

References

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. *Denoising Diffusion Probabilistic Models*. In *Ad-*

vances in Neural Information Processing Systems, volume 33, pages 6840–6851. Curran Associates, Inc. Available at: https://arxiv.org/abs/2006.11239.

Stanley H. Chan. 2024. *Tutorial on Diffusion Models for Imaging and Vision*. arXiv preprint arXiv:2403.18103. Available at: https://arxiv.org/abs/2403.18103.

Lilian Weng. 2021. *What are Diffusion Models?* Lil'Log. Available at: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241. Springer, Cham. Available at: https://arxiv.org/abs/1505.04597.

Sadeep Jayasumana, Srikumar Ramalingam, Andreas Veit, Daniel Glasner, Ayan Chakrabarti, and Sanjiv Kumar. 2024. *Rethinking FID: Towards a Better Evaluation Metric for Image Generation*. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9307–9315. Available at: https://arxiv.org/abs/2401.09603.