Deep Learning - Assignment 1

Bart van der Woude (s3498891)
Thijs Lukkien (s3978389)
Kyriakos Antoniou (s5715881)

August 29, 2025

1 Introduction

In this report, we will be discussing a paper called 'Denoising Diffusion Probabilistic Models' [1]. In this section, we will give a general introduction, before getting into the paper's technical details in Section 2. Once we have explained the technicalities, we will discuss how the paper fits in a broader context in Section 3.

As suggested in its name, the paper is centered around diffusion models. This is a family of models inspired by the diffusion process that is naturally exhibited by liquids when spreading out in another substance. In diffusion models, this process is represented by a Markov Chain that iteratively adds noise to an image.

So, the term 'diffusion' comes from the spreading out of liquids and is represented by a noise-adding Markov Chain. What about the 'model' aspect? In a broader sense, diffusion models are a type of latent variable prediction model. Here, 'latent variable' refers to some underlying, difficult-to-measure mechanism. Some examples of latent variables are 'measure of pride', 'political inclination', or '(conditional) Gaussian noise distributions applied to an image'. The latter of these is what the denoising diffusion probabilistic model of the paper attempts to predict.

In other words: our selected paper covers a diffusion model that aims to predict the noise that is contained in an input image. The predicted noise can then be subtracted from the noisy image, creating a slightly less noisy image. This method results in a noise removal process.

However, if one starts with pure noise, this method will remove noise until the original input image emerges from the original input image. The original input image was pure noise so the resulting image is not 'uncovered' but rather generated. This way, using pure noise as input, a denoising diffusion model can be used for image generation.

Our selected paper aims to improve the output quality of this type of model by applying and combining several optimization techniques.

2 Technical Summary

In Section 1 we discussed how a denoising diffusion probabilistic model is used to predict noise contained in an input image. This is realized in two processes. Firstly, a series of noisy images is needed, where it is known what the original image was, and what noise was added to it. This is done by iteratively applying Gaussian noise to an image with a (parameterized) Markov Chain.

It is called parameterized because certain parameters (β, see Section 2.1) influence the probability distribution function of the Gaussian noise over time. The generation of noisy images for training is called the forward process. This is discussed in more detail in Section 2.1

The second part is learning to predict the noise contained in an input image. This is done using a convolutional neural network called U-Net [2]. This is called the reverse process, and it is described in more detail in Section 2.2.

Intuitively: the forward process is gradually 'noisifying' an image, thereby creating a training set. The reverse process is training a network to predict the noise at a given timestep in the 'noise timeline', as depicted in Figure 1.

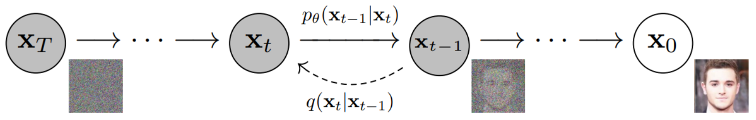
$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_{t-1}$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$\mathbf{x}_0$

Figure 1: The noise-adding Markov Chain, as shown in the original paper [1]. Counter-intuitively, the forward process is depicted from right to left, and the reverse process is depicted from left to right.

2.1 Forward process

The forward process refers to adding (Gaussian) noise to an image. At the time-step $t = 0$ the image is the original image $\mathbf{x}_0$. Each subsequent time step adds Gaussian noise to the image resulting in $\mathbf{x}_t$. The noised image at t can be calculated using Equation 1.

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon \tag{1}$$

Note, $\bar{\alpha}_t$ is the combined Gaussian of multiple time-steps as shown in Equation 2.

$$\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i = \alpha_1 \cdot \alpha_2 \cdot \ldots \cdot \alpha_t \tag{2}$$

Here, $\alpha_t = 1 - \beta_t$. Note that β_t is a scheduled parameter that increases linearly from 0.0001 to 0.02. The benefit of using Equation 1 is that there is no need for recursively adding noise. Rather, the noisy image at a certain time-step t can be reached in one computation.

2.2 Reverse process

For the reverse process, a U-Net is implemented that predicts the noise that was added to $\mathbf{x}_t$. By subtracting this predicted noise from the noised image we aim to retrieve the original iamge. The reverse process, denoted p_θ, is shown in Equation 3. Given that the variance of the noise, i.e. $\boldsymbol{\Sigma}_\theta\left(\mathbf{x}_t, t\right)$, is known due to the dependence on the linear scheduler β_t, it is only necessary to predict the mean of the noise, i.e. $\boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t\right)$.

$$p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)=\mathcal{N}\left(\mathbf{x}_{t-1} ; \boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t\right), \boldsymbol{\Sigma}_\theta\left(\mathbf{x}_t, t\right)\right) \tag{3}$$

The U-Net is an encoder-decoder model whose main characteristic is that the output consists of the same dimensions as the input. It accomplishes this by having a sequence of down-sampling convolutions (encoder) followed by a sequence of up-sampling convolutions (decoder). Generally, the encoder creates a feature representation of the input image and the decoder uses this feature representation to create an output image. Residual connections make sure that the feature representation does not completely erase the structure of the input. In addition to the standard U-Net, this research added attention blocks in between the encoder and decoder for added context dependencies between features from the encoder. Group normalization as well as time-step sinusoidal embeddings (positional encoder) were added.

The loss function to compare the noise of the input image (known) and the predicted noise is based on the negative log-likelihood. By adding a KL-divergence term to this negative likelihood an upper bound is created for this loss function, see Equation 4. The KL-divergence term and the negative log-likelihood can then be refactored to find L_t, which results in Equation 5. The authors [1] found empirically that the scaling factor was not particularly relevant, resulting in the simpler Equation 6. L_0 can be found through refactoring resulting in Equation 7.

$$\begin{aligned}\log \frac{q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right)}{p_\theta\left(\mathbf{x}_{0:T}\right)} &= \underbrace{D_{\mathrm{KL}}\left(q\left(\mathbf{x}_T \mid \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_T\right)\right)}_{L_T} + \\ \sum_{t=2}^{T} & \underbrace{D_{\mathrm{KL}}\left(q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)\right)}_{L_{t-1}} \underbrace{-\log p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right)}_{L_0}\end{aligned} \tag{4}$$

$$L_t=\frac{\left(1-\alpha_t\right)^2}{2 \alpha_t\left(1-\bar{\alpha}_t\right) \sigma_t^2}\left\|\epsilon_t-\epsilon_\theta\left(\mathbf{x}_t, t\right)\right\|^2 \tag{5}$$

$$L_t \approx \left|\epsilon_t-\epsilon_\theta\left(\mathbf{x}_t, t\right)\right\|^2 \tag{6}$$

$$L_0 \approx \left\|\epsilon_1-\epsilon_\theta\left(\mathbf{x}_1, 1\right)\right\|^2 \tag{7}$$

These math derivations were added for completeness. The key point is that a loss function can be derived that compares actual noise with predicted noise as in Equation 8. This equation is essentially the MSE between actual and predicted noise matrices. ϵ_t is the actual noise at a time-step t, $\epsilon_\theta(x_t, t)$ outputs the expected noise given an input image x_t and time-step t (the U-Net), $\bar{\alpha}_t$ is the combined Gaussian based on scheduler parameter β as described in Equation 2. With this loss function, it is now possible to create an algorithm that performs gradient descent to learn to recognize the noise that needs to be removed from a (noisy) input image in order to retrieve an image at an earlier time-step (less noise).

$$L_{simple} = \left\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t\right)\right\|^2 \tag{8}$$

2.3 General prediction model

Table 1 shows the pseudocode for both training and inference of the general noise-predicting model. In training, an input image has noise added to it. Gradient descent is then applied based on the loss function 8. The final model can infer images step-wise from random noise as described in the inference pseudo code.

Algorithm 1 Training

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

Algorithm 2 Inference

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

Table 1: Pseudocodes for training and inference as given in the paper [1]. This table summarizes the technical details as discussed in Section 2.2.

3 Broader context

Now that we have discussed the paper in more detail, we will use this section to put it in a broader context.

3.1 The problem

Image generation was quite challenging with the methods available at the time of publication. The main advances were made with models such as Generative Adversarial Networks (GANs) or Variational Auto-Encoders (VAEs). These models were able to generate high-resolution images but were also costly and difficult to train [3]. Diffusion models, on the other hand, were relatively easy and efficient in both training and inference. However, so far, previous research had been unable to generate high-quality images using diffusion models, making them undesirable to use. This paper aimed to show how diffusion models can be used to generate high-quality images.

3.2 Importance of the problem

While there is no direct problem when a model is unable to produce high-quality output, it may defeat the purpose of using such a model. The problem of efficiency in machine learning, on the other hand, could directly pose a problem. However, we will expand on this later in Section 3.4 as we find it more fitting there.

One of the types of models that was performing well when this paper was written was the GAN. These models are especially difficult to train [3], which can be undesirable for several reasons. The main reason is simply the impracticality of having a difficult training process. Some others include model instability and hyperparameter sensitivity. For these reasons, it could be important to solve the problem of low-quality output as it could reduce possibly forced reliance on undesirably methods.

3.3 Previous works

As mentioned earlier, previous research on diffusion models did not achieve high-quality images. The method of diffusion models was first introduced by [4] and has (as far as the authors know and we could find) since then not had any major improvements regarding image generation.

3.4 Sufficiency of the current solutions

As mentioned in Section 3.3, the main solution to low-quality output of a diffusion model is to simply use another type of model for image generation. Examples of this were discussed in Section 3.1. In other words, there were no solutions before this paper. On the other hand, using models that are difficult to train could pose new problems. An argument against such approaches is that their carbon footprint increases quickly with model size and training time [5]. Hence, it is worth it to look into methods that focus on efficiency rather than performance.

3.5 The new approach

As mentioned in Section 3.3, diffusion-based image generation has not had any major improvements since its introduction five years before this paper. For that reason, it is difficult to define what research gap this paper investigated and what is proposed to be done differently. This paper suggested a possible configuration with which high-quality images can be generated. Apart from using U-Net, this paper quite closely follows the original implementation.

3.6 Advantages of the new approach

Diffusion models have multiple advantages, the most crucial of which are the fact that diffusion models can generate high-quality images of similar and sometimes better quality than other state-of-the-art methods while at the same time being simple to define and efficient to train. Furthermore, the authors mentioned how diffusion models can be much more stable compared to GANs while also being able to generate more diverse data with fewer artifacts.

3.7 Future direction

This paper has shown that, while retaining efficient training, diffusion models can be used to produce high-quality output. Moreover, this paper has shown the potential of diffusion models. This way, it has paved the way for future research to investigate more methods to improve performance, as well as the applicability in different fields. One such possible application mentioned by the authors is data compression.

Moreover, should diffusion models perform to a sufficient extent, they could be used in a variety of environments. For example for the lossy compression of internet traffic or data storage. Alternatively, diffusion models can be used in creative fields, such as art or music.

As this paper comes from the year 2020, we can see what research followed this paper. We can also see that some of the possible future directions have been realized. Many famous research projects have followed since this paper was written. Some examples are Stable Diffusion [6], Dall-E 2 [7], and the recent video generation model Sora [8]. All of these models build upon the proof that diffusion models have the potential for high performance.

4 Relation to the field of deep learning

One definition of deep learning is: "A class of machine learning techniques that exploit many layers of non-linear information processing for supervised or unsupervised feature extraction and transformation, and for pattern analysis and classification" [9].

The method that was presented in this paper used a U-Net; a machine learning technique using many layers of non-linear information processing. It performed supervised (latent) pattern analysis and classification (though rather 'prediction') by predicting the noise that was produced by the Markov Chain. Hence, if we adhere to the above definition, this paper is a clear example of deep learning.

References

[1] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” 2020.

[2] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” 2015.

[3] D. Saxena and J. Cao, “Generative adversarial networks (gans survey): Challenges, solutions, and future directions,” 2023.

[4] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks, “Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7327–7347, 2022.

[5] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean, “Carbon emissions and large neural network training,” *arXiv preprint arXiv:2104.10350*, 2021.

[6] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, “High-resolution image synthesis with latent diffusion models,” 2022.

[7] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, “Hierarchical text-conditional image generation with clip latents,” 2022.

[8] Y. Liu, K. Zhang, Y. Li, Z. Yan, C. Gao, R. Chen, Z. Yuan, Y. Huang, H. Sun, J. Gao, L. He, and L. Sun, “Sora: A review on background, technology, limitations, and opportunities of large vision models,” 2024.

[9] L. Deng and D. Yu, 2014.