DENOISING DIFFUSION PROBABILISTIC MODELS FOR ORTHOPHOTOS

Casper Mailund Nielsen (s244492)

Technical University of Denmark
Department of Applied Mathematics and Computer Science
DK-2800 Kgs. Lyngby

ABSTRACT

Denoising Diffusion Probabilistic Models (DDPMs) have emerged as highly effective generative models for image synthesis tasks. Recently, they have gained traction in remote sensing. However, their application to orthophotos has not been extensively studied. This study aims to implement and train a DDPM in PyTorch to generate orthophotos of protected natural habitats in Denmark. The dataset consisted of 358,910 geographical locations, each with 9 images captured in the spring every year from 2016 to 2024. The trained model was evaluated using Fréchet Inception Distance (FID), resulting in a score of 92.11. Implementation is available at: https://github.com/caspermailund/ddpm.

1. INTRODUCTION

Drawing inspiration from nonequilibrium thermodynamics, Denoising Diffusion Probabilistic Models (DDPMs) are a type of generative model proposed for image synthesis tasks [1]. Compared to similar generative models like Variational Autoencoders (VAEs) [2] and Generative Adversarial Networks (GANs) [3], DDPMs represent an advancement in artificial intelligence. By reversing the process of turning regular images into random noise, these models have showcased remarkable performance in generating high-quality images. Since the publication of the initial study in 2020 [1], DDPMs have gained widespread adoption in computer vision tasks such as super-resolution [4], image colorization [5], and inpainting [5]. Additionally, the principles of DDPMs have been applied to other fields, achieving state-of-the-art results in areas such as audio synthesis [6], natural language processing [7], and molecular design [8].

The success of DDPM has also paved the way for their application to various image processing tasks within the remote sensing (RS) domain [9]. This is particularly due to the presence of noise in RS images, which can be effectively mitigated by the denoising capabilities inherent in DDPMs. Moreover, the solid mathematical foundation and step-by-step learning process of DDPMs, combined with their ability to provide more stable training compared to GANs, make them particularly effective for learning complex data distributions and efficiently processing large-scale RS datasets. The majority of the existing literature has concentrated on the use of DDPMs for satellite images (images taken from space) [9]. This study aims to implement and train a DDPM in PyTorch to generate orthophotos (images taken from an aircraft), based on the foundational DDPM framework introduced in 2020 by Ho et. al. [1].

2. BACKGROUND

This section provides an overview of the DDPM framework as described by Ho et al. [1], structured into three parts: the *forward* process, the *reverse* process and sampling.

2.1 The forward process

In the forward process, Gaussian noise is incrementally added to an original image x_0 over T iterations. This results in a series of noise-contained images $x_1, x_2, \dots, x_T$. As each image x_t depends solely on x_{t-1}, the process can be defined as a Markov chain:

$$q(x_t|x_{t-1}) := N(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I}). \qquad (1)$$

The term $q(x_t|x_{t-1})$ denotes the transition probability of the Markov chain, describing Gaussian noise distribution added at each step. β_t is a hyperparameter that controls the variance of the Gaussian distribution, increasing linearly with time t. $\mathbf{I}$ denotes the identity matrix with the same dimensions as the input image x_0.

Though the process is described as a Markov chain, Ho et. al. [1] demonstrates that obtaining x_t from x_0 does not require sequential iteration through all intermediate timesteps, as x_t can be derived directly in closed form by reparameterization. Using the notation $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{i=1}^{t} \alpha_i$, equation (1) can be expanded as:

$$\begin{aligned} x_t &= \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1} \\ &= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-1}}\epsilon_{t-2}\right) + \sqrt{1-\alpha_t}\epsilon_{t-1} \end{aligned}$$

$$= \sqrt{\alpha_t \alpha_{t-1}} x_{t-2} + \sqrt{1 - \alpha_{t-1}\alpha_{t-2}}\bar{\epsilon}_{t-2}$$
$$= \cdots$$
$$= \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\bar{\epsilon}, \quad (2)$$

where $\epsilon_{t-1}, \epsilon_{t-2} \cdots \sim N(0, \mathbf{I})$ and $\bar{\epsilon}_{t-2}$ merges two Gaussians. When two Gaussians with different variance, $N(0, \sigma_1^2\mathbf{I})$ and $N(0, \sigma_2^2\mathbf{I})$, are being merged, the new distribution becomes $N(0, \sigma_1^2 + \sigma_2^2\mathbf{I})$. Thus, any noised image x_t satisfies:

$$q(x_t|x_0) = N(x_t; \sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)\mathbf{I}). \quad (3)$$

2.2 The reverse process

In the reverse process, solving $q(x_{t-1}|x_t)$ directly is not possible because it is intractable. Instead, the reverse process relies on a neural network to approximate the reversed transition probability $q(x_{t-1}|x_t)$.

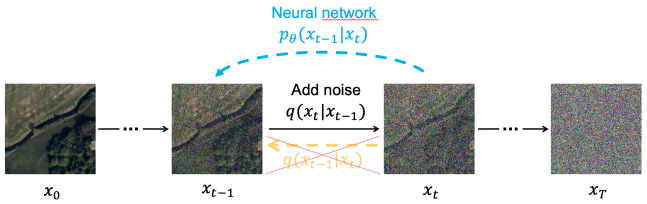

Fig 1. Illustration of how noise is added in the forward process $q(x_t|x_{t-1})$, while the corresponding reverse process $q(x_{t-1}|x_t)$ cannot be calculated directly. Therefore, a neural network is used to approximate the reversed transition probability $q(x_{t-1}|x_t)$.

The neural network seeks to learn the distribution:

$$p_\theta(x_{t-1}|x_t) = N\big(x_{t-1}; \mu_\theta(x_t, t), \sum_\theta(x_t, t)\big), \quad (4)$$

with θ being the parameters of the neural network to be optimized.

Training aims to minimize the Kullback-Leibler (KL) divergence between the backward diffusion process $p_\theta(x_0, x_1, \ldots, x_T)$ and the forward diffusion process $q(x_0, x_1, \ldots, x_T)$:

$$\begin{aligned}\mathcal{L}(\theta) &= \mathrm{KL}(q(x_0, x_1, \ldots, x_T)||p_\theta(x_0, x_1, \ldots, x_T)) \\ &= -\mathbb{E}_{q(x_{0:T})}[\log p_\theta(x_0, x_1, \ldots, x_T)] + C \\ &= -\mathbb{E}_{q(x_{0:T})}[-\log p_{(x_T)} - \textstyle\sum_{t=1}^{T} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}] + C,\end{aligned} \quad (5)$$

where C is a constant independent of θ.

Introducing the prior x_0 into $q(x_{t-1}|x_t)$, Bayes' Rule can be applied to convert it:

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t, x_0, x_{t-1})}{q(x_t, x_0)}$$
$$= \frac{q(x_0)q(x_{t-1}|x_0)q(x_t|x_{t-1}, x_0)}{q(x_0)q(x_t|x_0)}$$
$$= q(x_t|x_{t-1}, x_0)\frac{q(x_{t-1}|x_0)}{q(x_t|x_0)}, \quad (6)$$

where $q(x_{t-1}|x_t, x_0)$ is conditioned on the original input image x_0, making it tractable, unlike $q(x_{t-1}|x_t)$. Once simplified, equation (6) becomes:

$$q(x_{t-1}|x_t, x_0) = N(x_{t-1}; \tilde{\mu}_t(x_t), \tilde{\beta}_t\mathbf{I}), \text{ where} \quad (7)$$

$$\tilde{\mu}_t(x_t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\bar{\epsilon}) \text{ and } \tilde{\beta}_t = \frac{1-\alpha_{t-1}}{1-\alpha_t}\beta_t. \quad (8)$$

In equation (8) both α_t and β_t are constants, only $\bar{\epsilon}$ can be parameterized by the neural network as:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, \mathrm{t})). \quad (9)$$

Thus, $\epsilon_\theta(x_t, \mathrm{t})$ is the only learnable part of the reverse process.

On the basis of equation (9) and (2), the optimization goal can be described by the following (simple) form according to Ho et. al. [1]:

$$\mathcal{L}_{simple}(\theta) = \mathbb{E}_{x_0,\epsilon,t}[||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\bar{\epsilon}, t)||^2]. \quad (10)$$

The loss function reveals that the fundamental purpose of DDPMs is to minimize the difference between the predicted noise ϵ_θ and the actual noise ϵ.

2.3 Sampling

When the neural network has been trained, the distribution p_θ is learned. This distribution is used to sample (i.e. generate) a new image starting from Gaussian noise. It involves iteratively sampling T to $t = 1$ and thereby reaching x_0. According to equation 7, we have:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right) + \sigma_t z, \quad (11)$$

where $z \sim N(0, \mathbf{I})$. The neural network outputs $\epsilon_\theta(x_t, t)$, while α_t and $\bar{\alpha}_t$ are calculated based on β_t as described in section 2.1 regarding the forward pass. The sampling process is illustrated in Fig. 4.

3. DATA

The dataset consists of orthophotos capturing protected natural habitats across Denmark. The protected areas under § 3 of the Nature Conservation Act include lakes, bogs,

freshwater meadows, salt marshes, heathlands, pastures, and streams [10].

Each image corresponds to a specific geographical location, with a total of 358,910 distinct locations identified. These locations were determined by overlaying Denmark with 10x10 km tiles and extracting coordinates for § 3 areas. It was possible to extract § 3 areas, as Danish municipalities have outlined these habitats with polygons using coordinates, as part of their efforts to monitor protected natural habitats [11]. The identified areas were collected as image data through an API managed by Klimadatastyrelsen [12], and divided into non-overlapping 256x256 pixel images, each covering 128x128 meters.

In the dataset, each unique geographical location has a set of nine RGB images, one taken in the spring every year from 2016 to 2024. As shown in Fig. 3, the nature of the area can vary slightly, with images exhibiting differences in colour and shadowing.

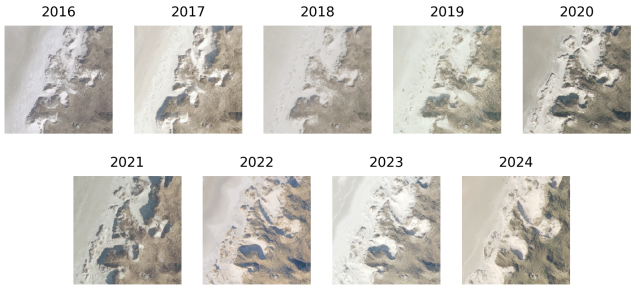

Fig 2. Display of nine images from the years 2016 to 2024, all of the same geographical location, portraying saltwater marsh.

4. METHODS

The neural network used is based on a U-Net architecture, implemented in PyTorch, closely following Ho et al.'s TensorFlow implementation [13].

4.1 The noise scheduler

As described in section 2.1 regarding the forward process, a hyperparameter β_t needs to be defined in order to control the Gaussian noise distribution that is being incrementally added to the image. Following the approach of Ho et. al. (2020), I set $T = 1000$ and linearly increase $\beta_1 = 0.0001$ to $\beta_T = 0.01$. It is worth noting that recent studies suggest using a cosine scheduler to increase β_t might lead to better results [14]. The reason behind this is that the linear scheduler causes the image to lose its information fairly quickly with the image starting to resemble pure noise halfway through, while the cosine scheduler adds noise more slowly.

4.2 Positional encoding

During the training of a DDPM, it is not enough for the model to simply receive a noisy image. For the model to understand where the image is positioned in the diffusion process, it must be informed of the current timestep. For this reason, positional encoding, or timestep encoding, needs to be implemented, and there are various options for how this can be done. This paper takes inspiration from the positional encoding proposed by Vaswani et al. [15], which involves precomputing the positional encodings using sine and cosine functions:

$$PE_{(pos,2i)} = \sin\left(pos/10000^{2i/d_{model}}\right)$$
$$PE_{(pos,2i+1)} = \cos\left(pos/10000^{2i/d_{model}}\right), \quad (12)$$

where pos represents the position and i denotes the dimension. Each dimension of the positional encoding corresponds to a sinusoid. The wavelenghts form a geometric progression from 2π to $10000 * 2\pi$. The even-indexed dimensions are assigned sine values, while the odd-indexed dimensions are assigned cosine values.

These positional encodings are mapped to the model's feature space, ensuring that the timestep is represented in the same dimensionality as the feature maps in the network. Along with other features (from the image), the timestep embeddings are passed through the U-Net.

4.3 Model architecture and training

As previously mentioned, the architecture employed was a U-Net. It was unconditional, as the images in the dataset were not labelled, allowing for multiple § 3 areas to appear in a single image.

The trained U-Net use five feature map resolutions with two convolutional residual blocks per resolution level. It incorporated Wide ResNet blocks based on Zagoruyko et. al. [16]. The model injected timestep embeddings into all residual blocks using a multi-layer perceptron. Furthermore, self-attention blocks were implemented at the 16x16 resolution in both the downsampling and the upsampling based on Vaswani et al. [15]. Group normalization was applied instead of batch normalization. Finally, Swish activation functions were used instead of the alternative ReLU activation function, as Swish has been found to perform better in deeper models [17].

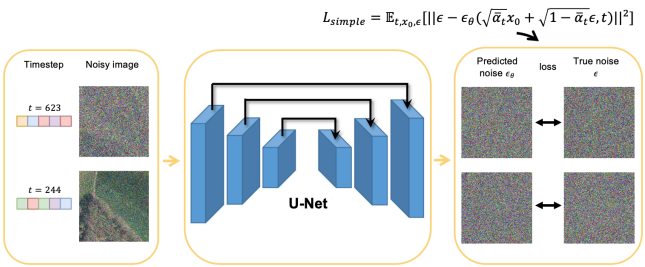

Fig 3. Simple illustration of the training procedure of a DDPM. Specifically, the neural network receives a noisy image and its timestep, pass it through a U-Net, and then minimizes the distance between the predicted noise ϵ_θ and the true noise ϵ in order to optimize the network.

The model was trained for 15 epochs before converging with a final loss of 0.0201. The learning rate was set to 0.00002, along with the Adam optimizer for training. Each epoch consisted of 358,910 images, with one image per geographical location. During training, the images were randomly drawn from different years. Thus, the year aspect served as a form of data augmentation, accounting for the variations in lighting and shadowing across different time periods.

5. RESULTS

The model evaluation begins with a sampling procedure.

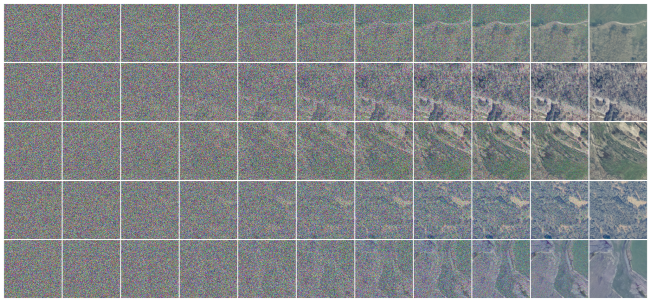

Fig 4. Exemplifying five sampling processes, moving from left to right, with each image corresponding to a timestep, jumping 100 steps between each, starting at x_{1000} and ending at x_0.

When evaluating a DDPM, both qualitatively and quantitatively considerations can be done. Firstly, qualitatively considerations will be discussed, followed by the quantitative measurement of the Fréchet Inception Distance (FID) [18].

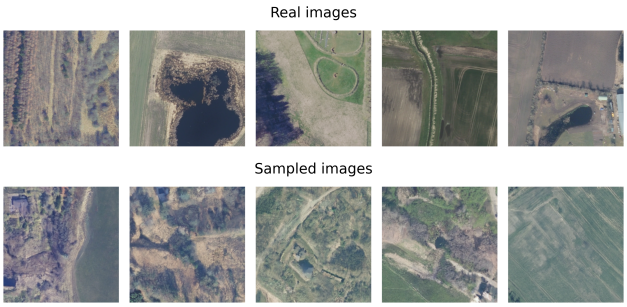

Fig 5. Comparison of five randomly drawn training images and five randomly sampled images.

Considering the variety of natural protected habitats in the training data, it would be ideal if the model could generate a distribution of images, as well as individual images, with a broad range of variations. The model was able to achieve this to some extent. For example, when looking at the leftmost sampled image in Fig. 5, it appears to include a house-like object, some vegetation, a coastline, and a sea-like element, all of which reflect diverse features found in the training dataset. This variety suggests that the model is capable of reproducing multiple forms of nature in a single image. However, upon closer inspection of the sampled images, it becomes evident that the details are not as sharp or accurate as in the training data. Fine structures, such as the intricate textures of vegetation, the houses or building-like structures, the smoothness of the coastline, and the subtle transitions between different elements, appear somewhat blurred or inconsistent. This suggests that the model may face challenges in capturing finer details, despite its ability to generate diverse scenes.

A calculated FID score of 92.11 was obtained by comparing 9,000 training images to 9,000 sampled images. This score summarizes the similarity between the two sets of images by comparing their statistics on computer vision features. Lower FID scores indicate that the two groups of images are more similar, with a perfect score of 0.0 [18].

To test the robustness of the measurement, an FID score of 5.69 was computed between 9,000 training images and 9,000 validation images (unseen during training). The FID score between the sampled images and validation images were very similar to the FID score between the training and sampled images.

To summarize, the model showed solid results, with FID scores suggesting similarity between the generated images and the training set. Nevertheless, better results might be achievable by training with higher-resolution images. While the API [12] supports up to 1024px images, the constraints of time and available computing power hindered the exploration of this option. For those interested in viewing examples of 100 training images and 100 generated samples, they can be found in my GitHub repository [19].

6. REFERENCES

[1] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models” in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 33, 2020, pp. 6840–685.

[2] C. Doersch, “Tutorial on variational autoencoders” *arXiv preprint* arXiv:1606.05908, 2016.

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks”, *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[4] C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi, “Image super-resolution via iterative refinement”, *arXiv preprint* arXiv:2104.07636, 2021

[5] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, “Score-based generative modeling through stochastic differential equations”, *arXiv preprint* arXiv:2011.13456, 2020

[6] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, “Diffwave: A versatile diffusion model for audio synthesis”, *arXiv preprint* arXiv:2009.09761, 2020.

[7] X. Li, J. Thickstun, I. Gulrajani, P. S. Liang, and T. B. Hashimoto, “Diffusion-lm improves controllable text generation” *in Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 35, 2022, pp. 4328–4343.

[8] B. Jing, G. Corso, J. Chang, R. Barzilay, and T. Jaakkola, “Torsional diffusion for molecular conformer generation” in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 35, 2022, pp. 24 240–24 253.

[9] Liu, Y., Yue, J., Xia, S., Ghamisi, P., Xie, W., and Fang, L., “Diffusion models meet remote sensing: Principles, methods, and perspectives”, *arXiv preprint* arXiv:2404.08926, 2024.

[10] Miljø- og Ligestillingsministeriet, “Bekendtgørelse af lov om naturbeskyttelse”, *Retsinformation*, March 3, 2019. (Online): https://www.retsinformation.dk/eli/lta/2019/240

[11] Danmarks Miljøportal, ”Beskyttede naturtyper”, *Arealdata*. (Online): https://arealdata.miljoeportal.dk/datasets/urn:dmp:ds:beskyttede-naturtyper

[12] Klimadatastyrelsen, "Ortofoto forår WMS", *Datafordeler*. (Online): https://datafordeler.dk/dataoversigt/geodanmark-ortofoto/ortofoto-foraar-wms/

[13] hojonathanho, “diffusion”, *GitHub.* (Online): https://github.com/hojonathanho/diffusion

[14] A. Nichol and P. Dhariwal, “Improved denoising diffusion probabilistic models”, *arXiv preprint* arXiv:2102.09672, 2021.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need”, *Advances in neural information processing systems*, vol. 30, 2017.

[16] S. Zagoruyko and N. Komodakis, “Wide residual networks”, *arXiv preprint* arXiv:1605.07146, 2017.

[17] P. Ramachandran, B. Zoph, and Q. V. Le, “Swish: A self-gated activation function”, *arXiv preprint* arXiv:1710.05941, 2017.

[18] M. Heusel, H. Ramsauer, T. Unterthiner, and B. Nessler, “GANs trained by a two time-scale update rule converge to a local Nash equilibrium”, *arXiv preprint* arXiv:1706.08500, 2018.

[19] caspermailund, “ddpm”, *GitHub.* (Online): https://github.com/caspermailund/ddpm