Denoising Diffusion Probabilistic Models for Orthophotos

Casper Mailund Nielsen (s244492)

Deep Learning 02456

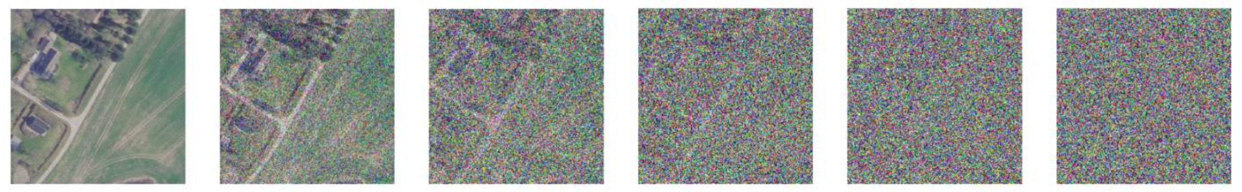
Department of Applied Mathematics and Computer Science, Technical University of Denmark

Introduction, motivation and dataset

Based on Ho et al. (2020)[1], this project seeks to build a diffusion model in PyTorch using orthophotos, closely following the authors' approach.

The decision to use orthophotos is motivated by the desire to explore how a diffusion model performs within the remote sensing domain, as geospatial data plays an increasingly critical role in environmental monitoring.

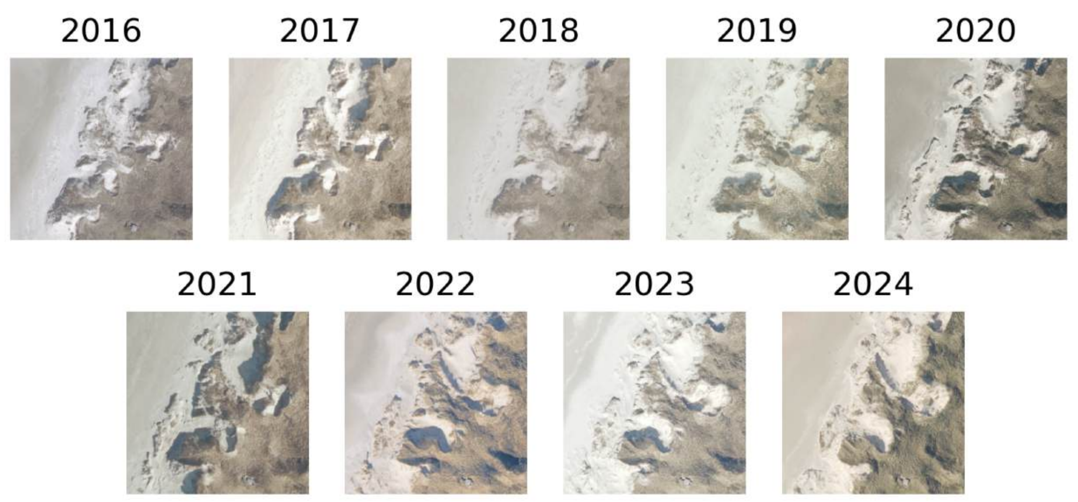
$$x_0 \rightarrow x_1 \rightarrow \ldots \rightarrow x_T$$



$$x_0 \leftarrow x_1 \leftarrow \ldots \leftarrow x_T$$

The dataset:

- Orthophotos of protected nature habitats in Denmark[2]
- Each image is tied to a geographical location, with a total of 358,910 different locations
- Each location has 9 images, one taken in the spring every year from 2016 to 2024
- RGB images with size 256x256 pixels, covering an area of 128x128 meter
- 1 pixel corresponds to an area of 50x50 cm
- Collected from an API monitored by Klimadatastyrelsen

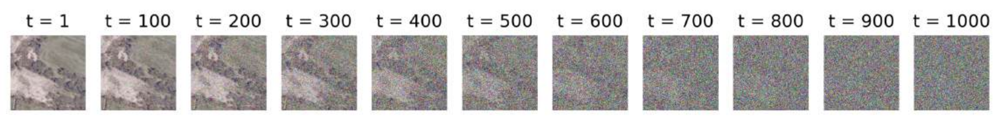


The forward process

The forward process is a Markov chain that gradually adds Gaussian noise to the input image x_0 step by step with a variance defined by $\beta_1, \ldots, \beta_T$. The process will produce a sequence of noisy image samples $x_1, \ldots, x_T$.

$$\underbrace{q(x_t|x_{t-1})}_{\text{Distribution of the noised images}} = N(\underbrace{x_t}_{\text{Output}}; \underbrace{\sqrt{1-\beta_t}x_{t-1}}_{\text{Mean } \mu_t}, \underbrace{\beta_t \mathbf{I}}_{\text{Variance } \Sigma_t})$$

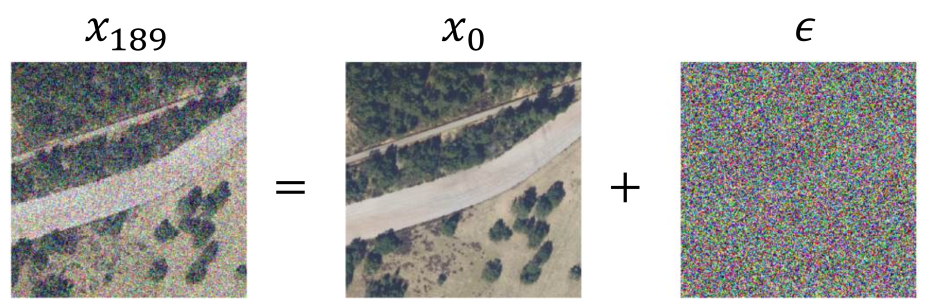
Like the authors Ho et al. (2020)[1], I set $T = 1000$ and linearly increase $\beta_1 = 0.0001$ to $\beta_T = 0.01$.



Alternative methods for noise scheduling exist, such as the cosine schedule, which is commonly used because it introduces noise at a slower rate[3].

The forward process admits sampling x_t at an arbitrary timestep t in closed form. By defining $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, the closed form formula ultimately looks like:

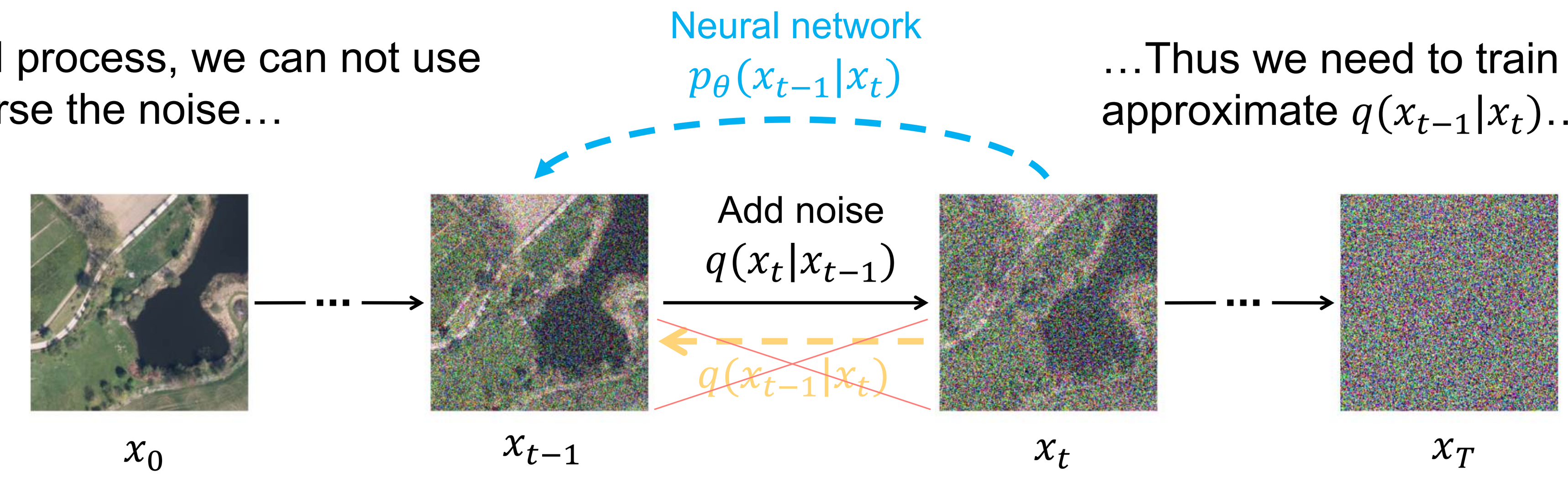
$$\boxed{x_t} = \sqrt{\bar{\alpha}_t}\boxed{x_0} + \sqrt{1-\bar{\alpha}_t}\boxed{\epsilon}$$



The reverse process

Unlike the forward process, we can not use $q(x_{t-1}|x_t)$ to reverse the noise…

…Thus we need to train a neural network to approximate $q(x_{t-1}|x_t)$…

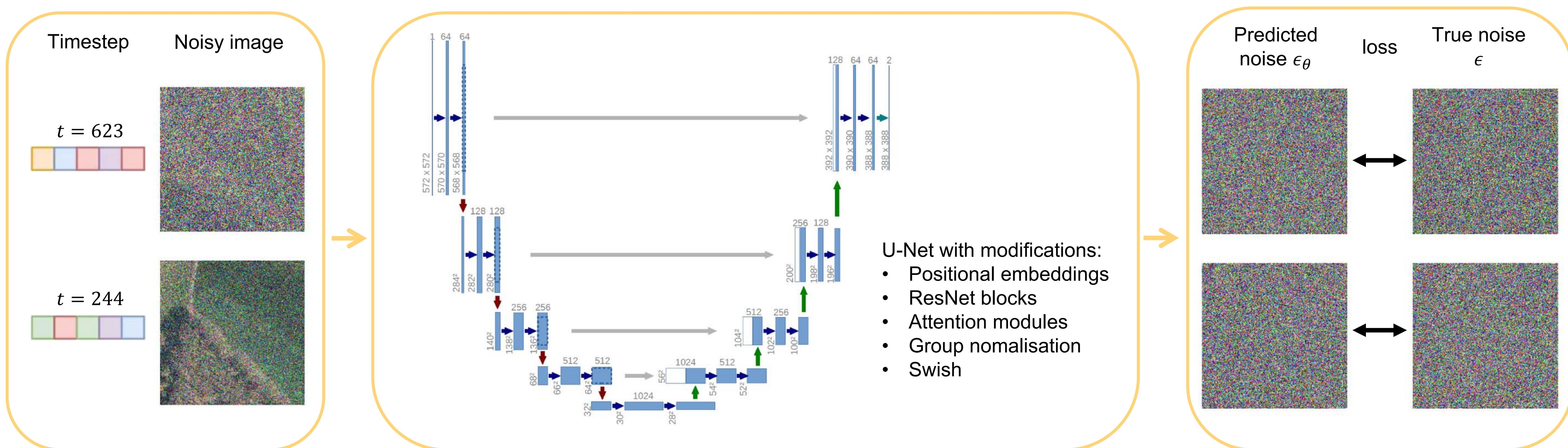


…It turns out, after doing a lot of math, p_θ is Gaussian in the form of $p_\theta(x_{t-1}|x_t) = N\left(x_{t-1}; \mu_\theta(x_t,t), \mathbf{\Sigma}_\theta(x_t,t)\right)$. Hence, we need to find $\mu_\theta(x_t,t)$ and $\mathbf{\Sigma}_\theta(x_t,t)$. The mean $\mu_\theta(x_t,t)$ is given by $\mu_\theta(x_t,t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t,t)\right)$. Fixing the variance $\mathbf{\Sigma}_\theta(x_t,t)$ to a constant $\sigma_t^2 = \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$, makes $\epsilon_\theta(x_t,t)$ the only learnable part of the reverse process.

Model

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

Timestep encoding



Sampling and preliminary results

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t,t)\right) + \sigma_t \mathbf{z}, \text{ where } \mathbf{z} \sim N(\mathbf{0}, \mathbf{I})$$

References

[1] Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. arXiv. https://doi.org/10.48550/arXiv.2006.11239

[2] Miljø- og Ligestillingsministeriet. (03/03/2019). Bekendtgørelse af lov om naturbeskyttelse. Retsinformation. https://www.retsinformation.dk/eli/lta/2019/240

[3] Nichol, A. Q., & Dhariwal, P. (2021). Improved denoising diffusion probabilistic models. arXiv. https://doi.org/10.48550/arXiv.2102.09672