Denoising Diffusion Probabilistic Model Implementation

02456 Deep Learning, Fall 2024, Final Project – Group 64

Eva Kaštelan (s232469), Jan Ljubas (s237214), Noa Margeta (s232470), Filip Penzar (s232452)

DDPM

Introduction

- Generative models
- Generate images by reversing a controlled noise process
- Two steps:
  - *Forward diffusion process – transforming an image to noise*

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

β_t is a variance schedule controlling the amount of noise added

  - *Reverse process – the model learns to undo noise addition in small steps*

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2\mathbf{I})$$

The model predicts the noise $\epsilon_\theta(x_t, t)$ so we reparametrize $\mu_\theta(x_t, t)$:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right)$$

- In training the primary goal is to approximate the probability distribution of the data
- The loss function is then modelled as a variational lower bound
- Minimize the difference between the actual noise added at each step and the noise predicted by the model using MSE:

$$L = \mathbb{E}_{t,x_0,\epsilon}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|^2\right]$$

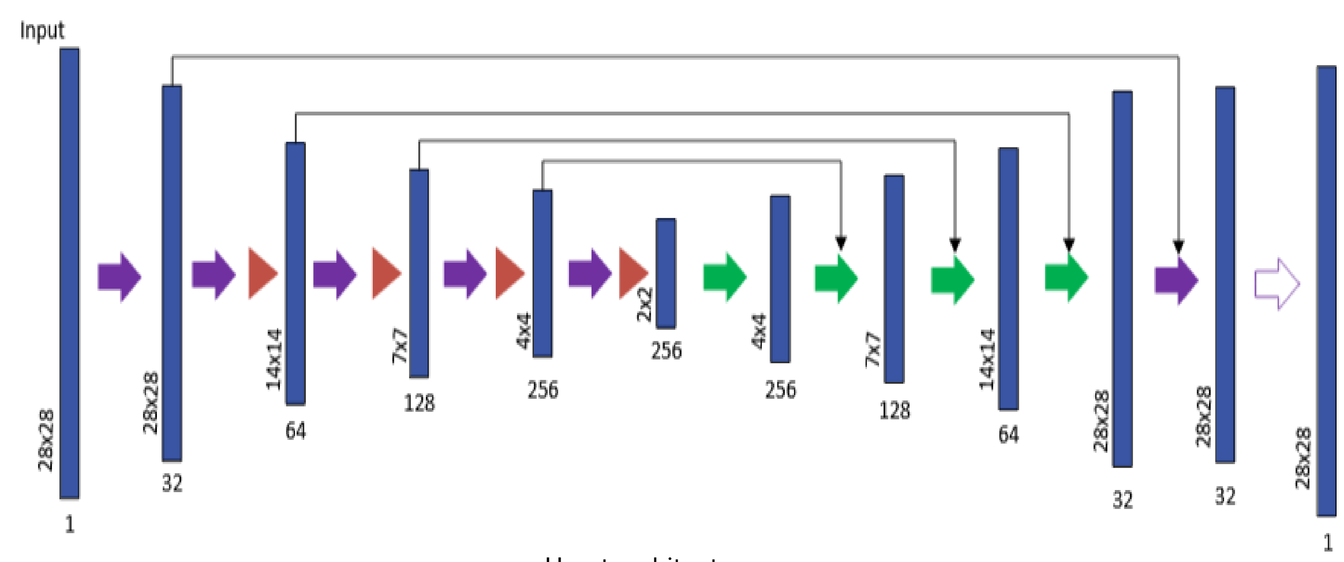
- During sampling, the reverse process iteratively applies:

$$x_{t-1} = \mu_\theta(x_t, t) + \sigma_t z, \quad z \sim \mathcal{N}(0, \mathbf{I})$$

until a clean image is generated

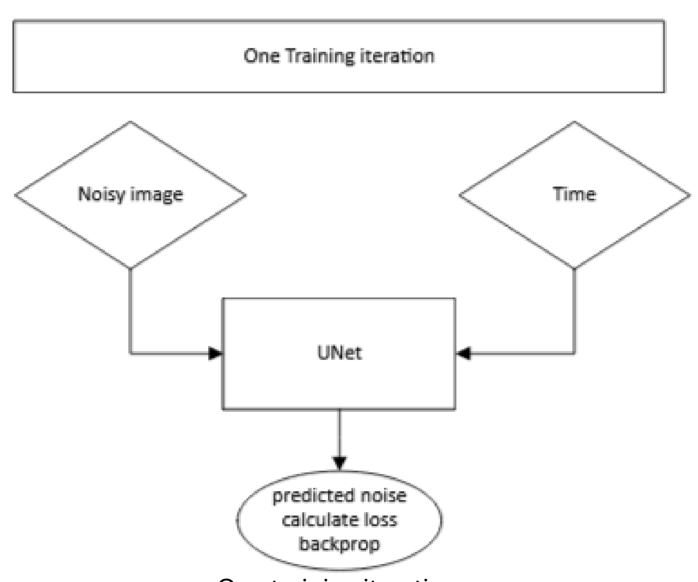
Implementation

- U-net architecture:

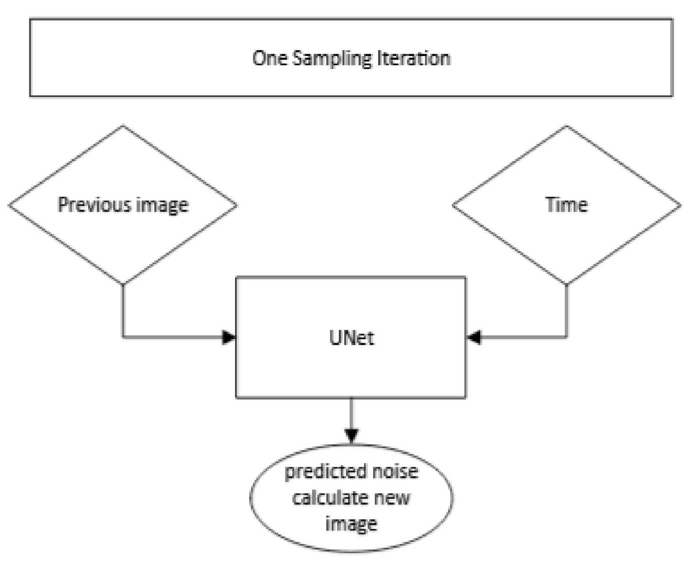


U-net architecture

- Embedding time-step:
  - Only once at the beginning
  - In-between every layer
  - Types
    - Linear
    - Sinusoidal
- AdamW optimizer, learning rate 2e-4



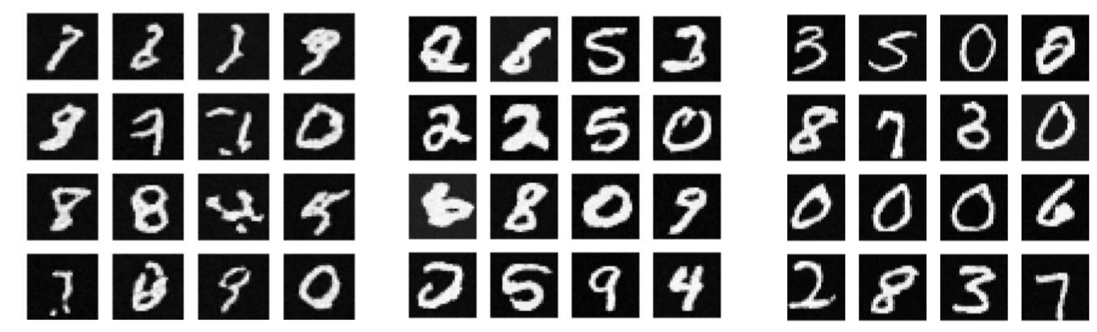
One training iteration



One sampling iteration

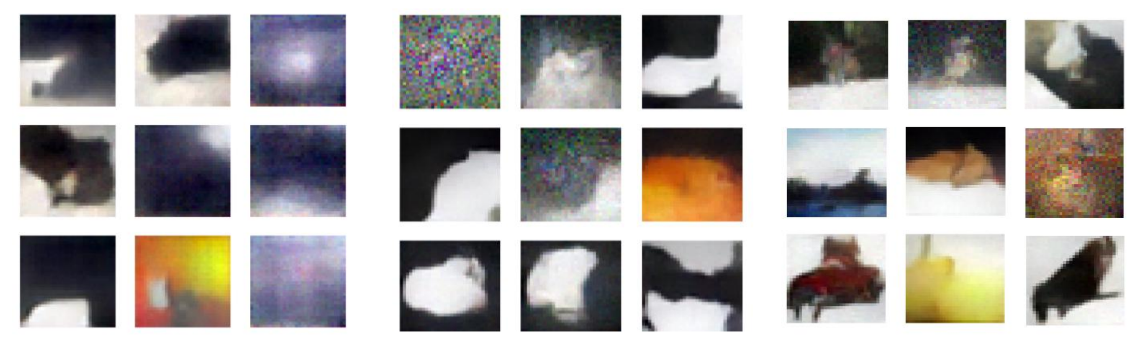
Results

MNIST model samples



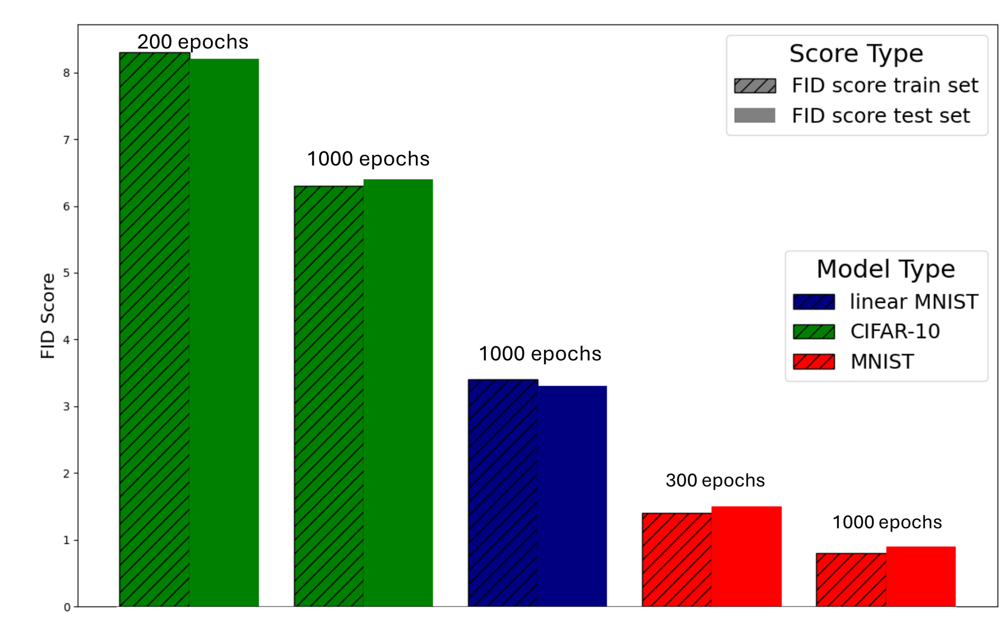
100 epochs | 500 epochs | 1000 epochs

CIFAR-10 model samples



100 epochs | 500 epochs | 1000 epochs

FID scores



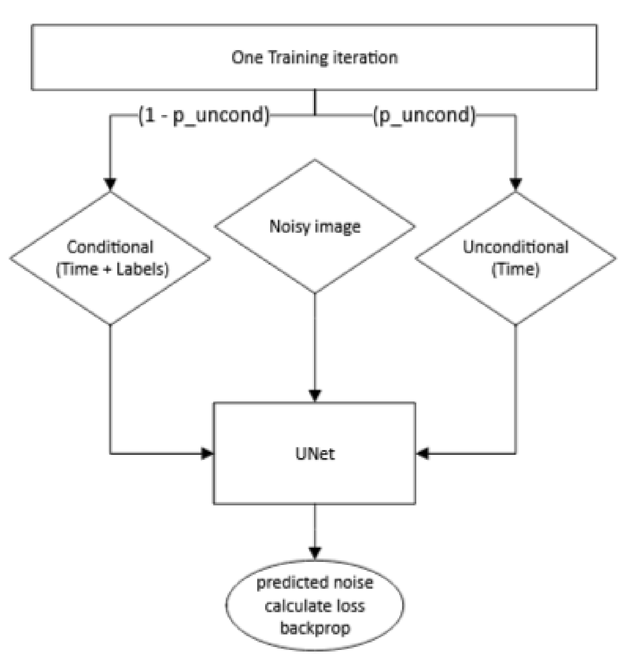
Conditional Diffusion Model

Introduction

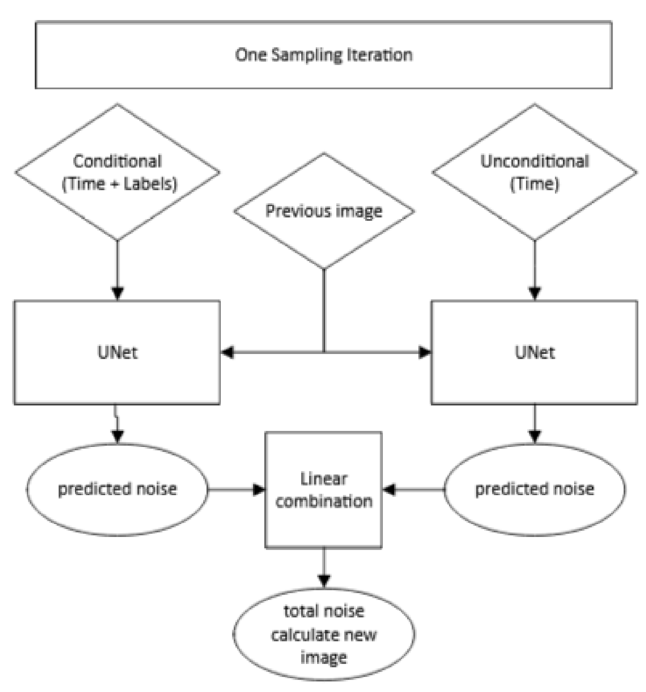
- Goal of generating "low temperature" samples
- Classifier guidance
  - Uses an additional classifier
  - Mixes diffusion model's score with classifier's input gradient log probability
- <u>Classifier-Free guidance</u>
  - No need for an additional classifier
  - Joint training of a conditional and an unconditional model
  - Conditional model, $p(\boldsymbol{z}|\boldsymbol{c})$, with $\epsilon_\theta(\boldsymbol{z}_\lambda, \boldsymbol{c})$
  - Unconditional model, $p(\boldsymbol{z})$, with $\epsilon_\theta(\boldsymbol{z}_\lambda)$
  - Single neural network for both, unconditional = conditional with $\boldsymbol{c} = \emptyset$
- Training
  - Training as a conditional model with p_{uncond} probability of training the unconditional model
- Sampling
  - Linear combination of the conditional and the unconditional models with guidance strength (w)
  - $\tilde{\epsilon}_\theta(\boldsymbol{z}_\lambda, \boldsymbol{c}) = (1+w)\epsilon_\theta(\boldsymbol{z}_\lambda, \boldsymbol{c}) - w\epsilon_\theta(\boldsymbol{z}_\lambda)$

Implementation

- Same architecture as our DDPM
- Sinusoidal time embedding
- Embedding for labels (28 dimensions)
- Time embedding and label embedding are added together, and appended to the model in forward process (conditional)
- Only time embedding is used for unconditional
- p_uncond = 0.1
- w = 3



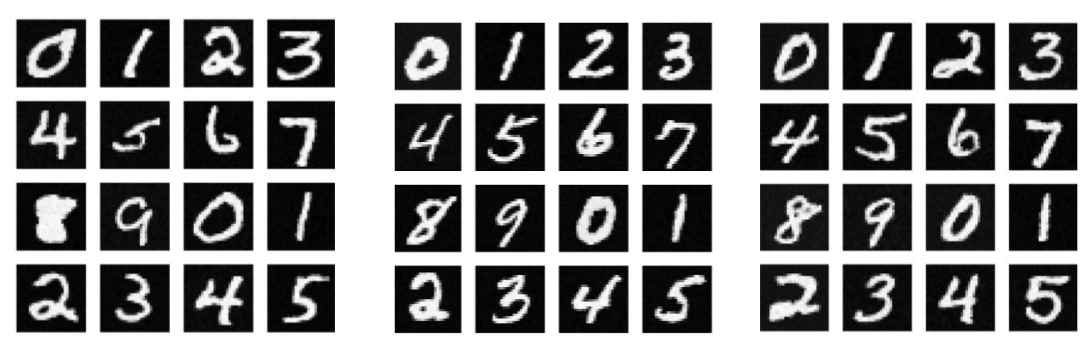
One training iteration



One sampling iteration

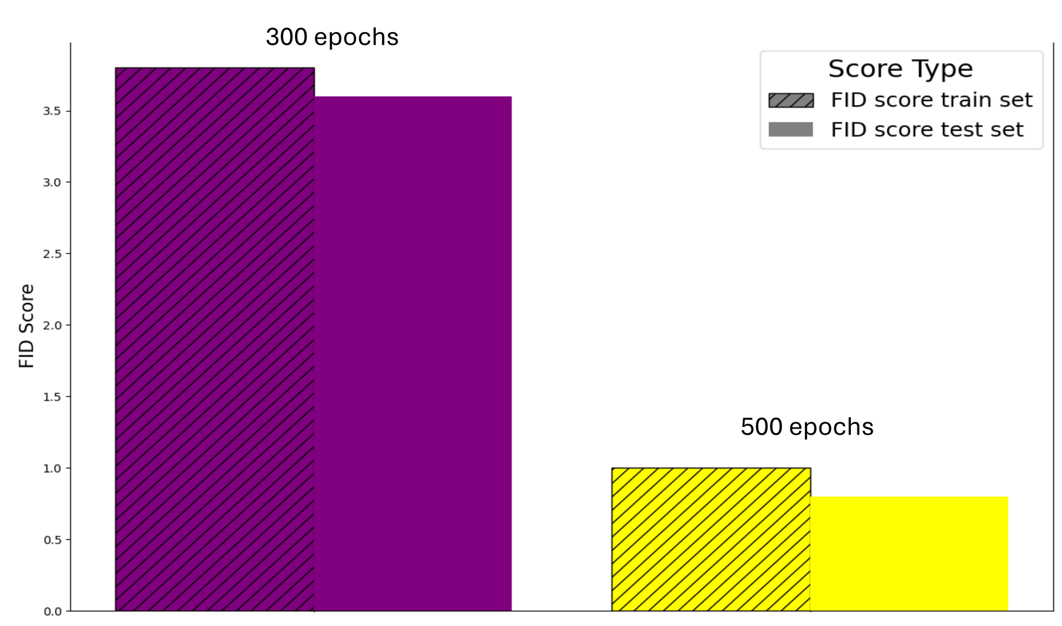
Results

CFG-MNIST model samples

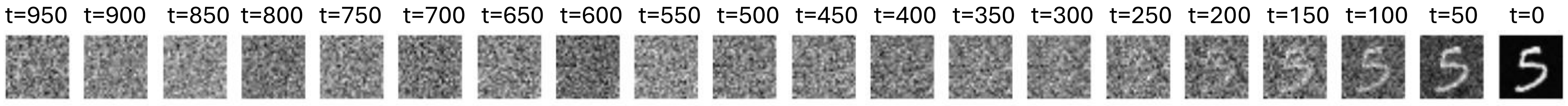


100 epochs | 300 epochs | 500 epochs

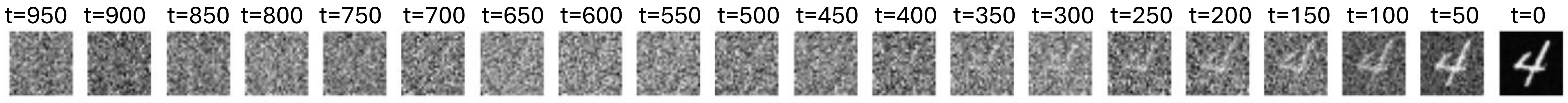
FID scores



Unconditional MNIST progressive generation:



Conditional MNIST progressive generation (label=4):



References

1) Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising Diffusion Probabilistic Models." *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*
2) Jonathan Ho and Tim Salimans. "Classifier-Free Diffusion Guidance." *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*
3) Paszke, A., Gross, S., Chintala, S., & Chanan, G. (2016). "PyTorch: An imperative style, high-performance deep learning library"
4) Outlier. (2022, June 6). Diffusion models | Paper explanation | Math explained [Video]. YouTube. https://www.youtube.com/watch?v=HoKDTa5jHvg