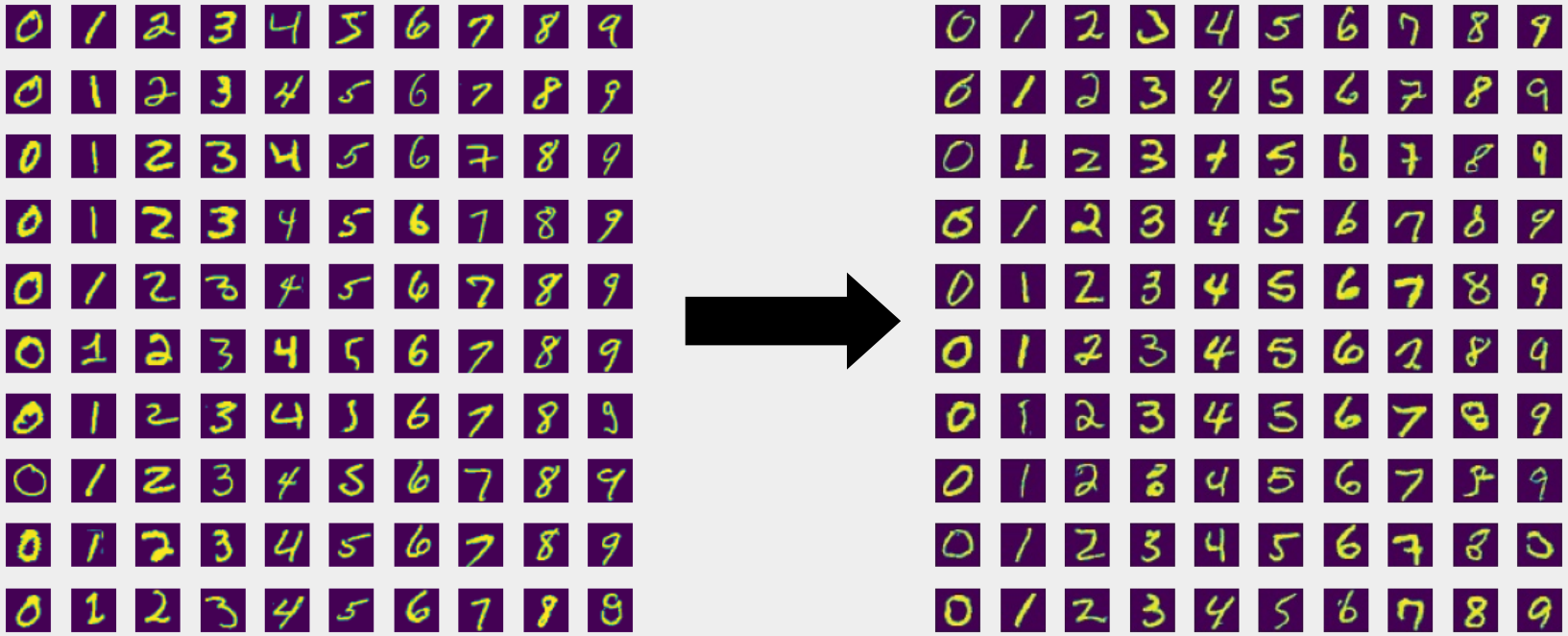
UNIVERSITÄT DUISBURG ESSEN

Offen im Denken



Denoising Diffusion Probabilistic Models

Project presentation for the course “Neural Networks and Deep Learning”

Marvin Henke – 1st September 2025

Supervisor: Dr. Nikodem Szpak

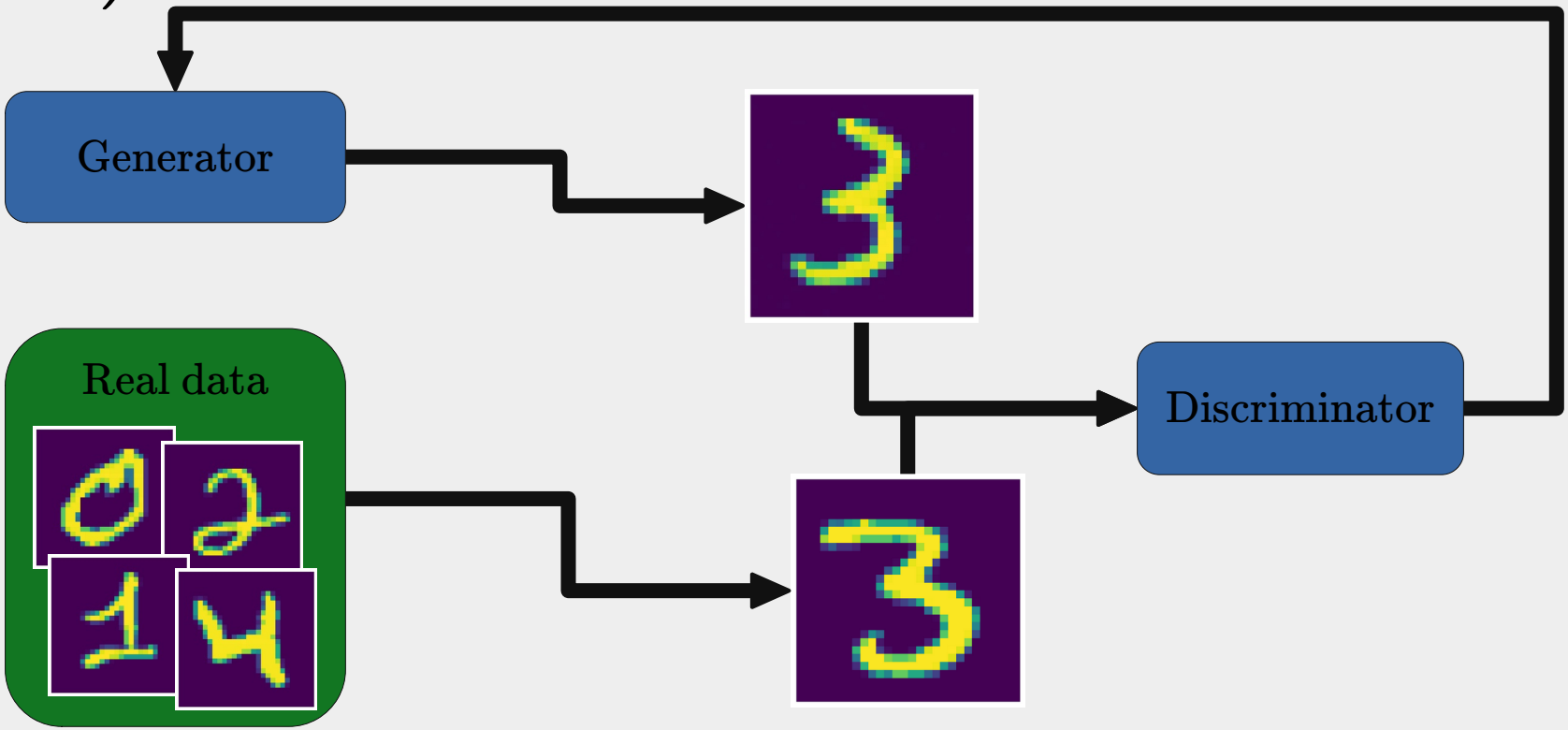
Contents

- Overview of Generative AI methods
- Intuition for DDPM
- DDPM Theory
- Implementation & Results
- Parameter study
- Summary
- Outlook

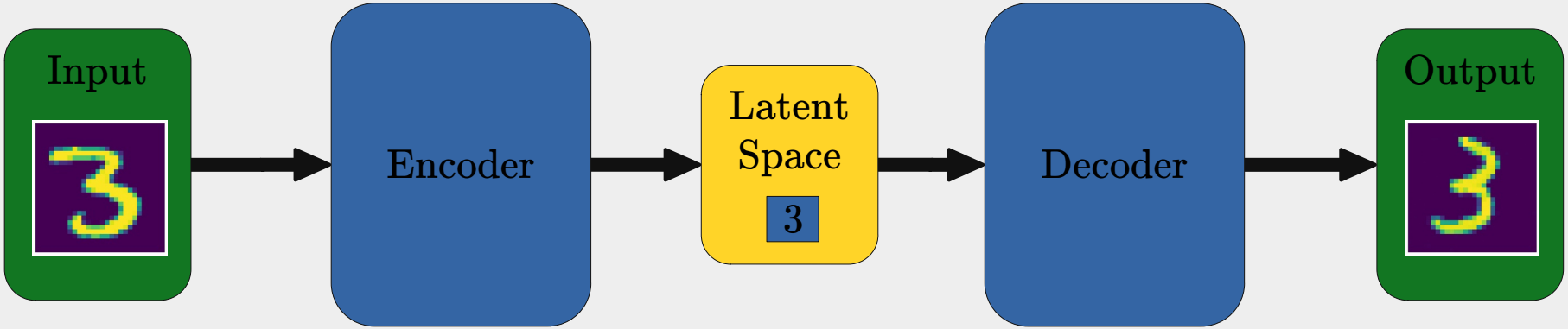
Generative AI methods

- Generative adversarial network (GAN)
  - Generator and Discriminator networks compete
- Variational autoencoder (VAE)
  - Encoder → Latent space → Decoder → Output
- Diffusion Models
  - Network learns to reverse a diffusion process

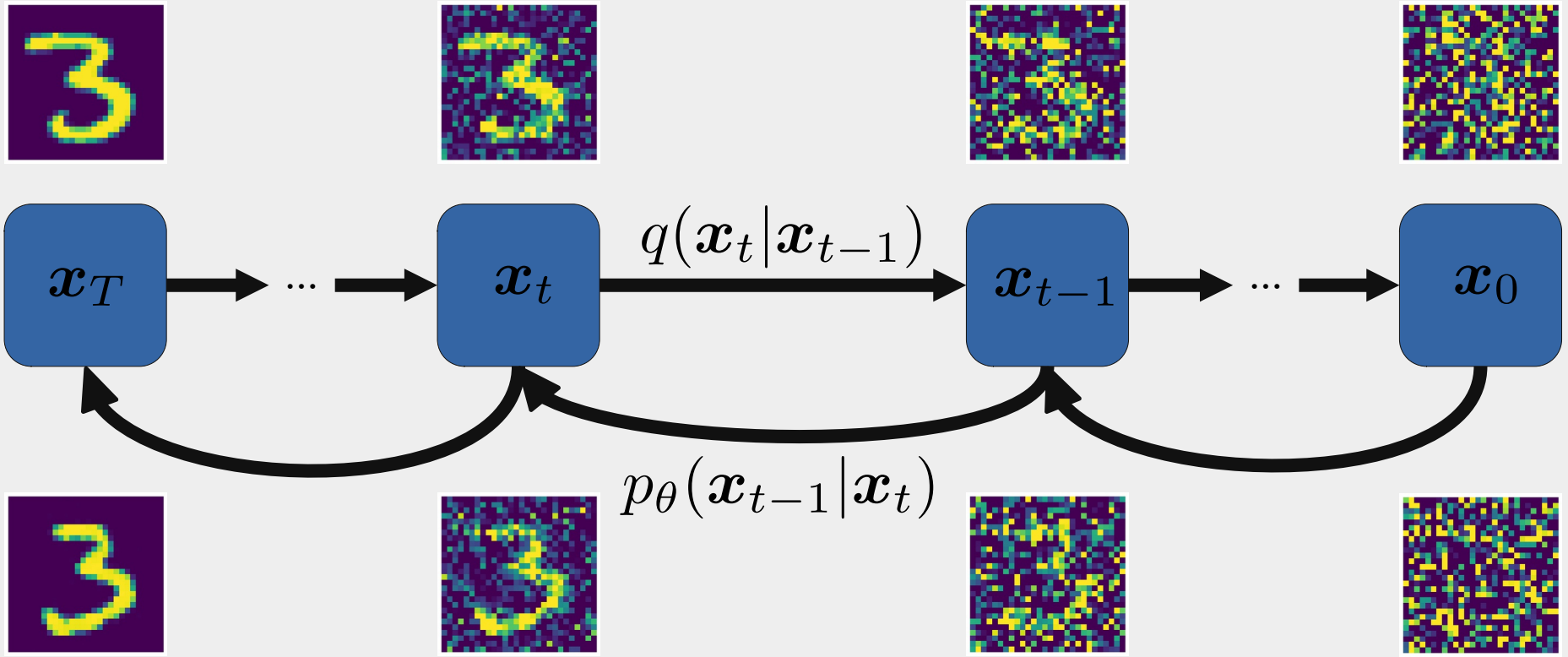
Generative adversarial network (GAN)



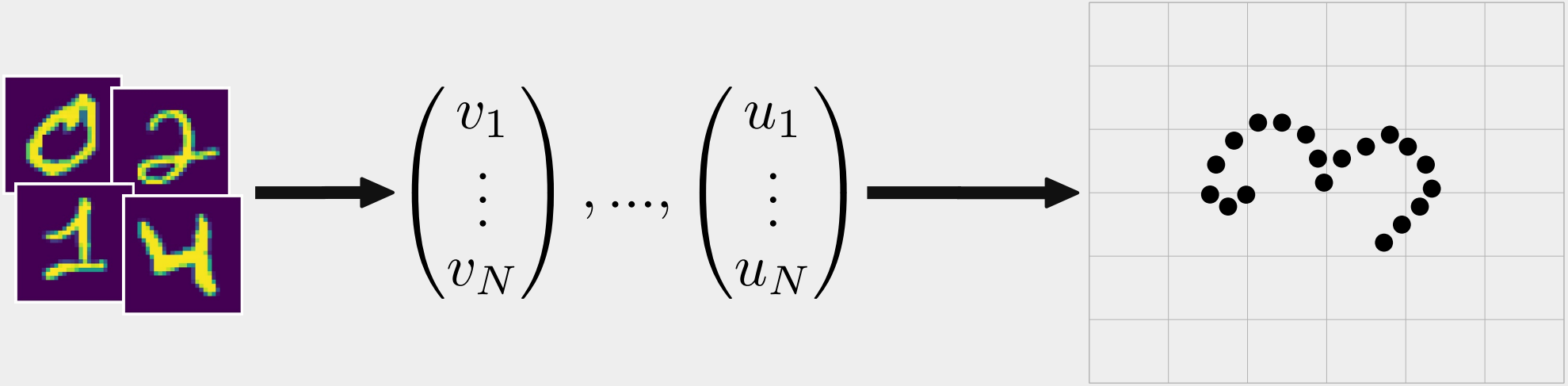
Variational autoencoder (VAE)



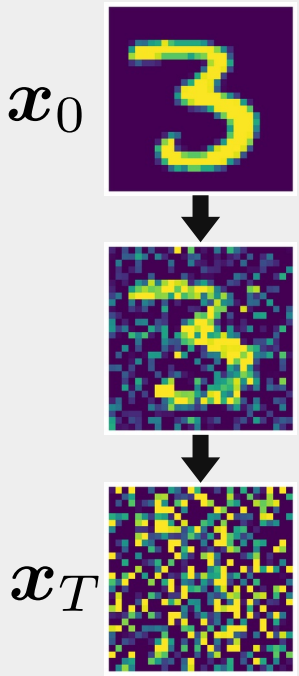
Diffusion Models



DDPM Intuition (Image representation)



DDPM Intuition (Forward process)

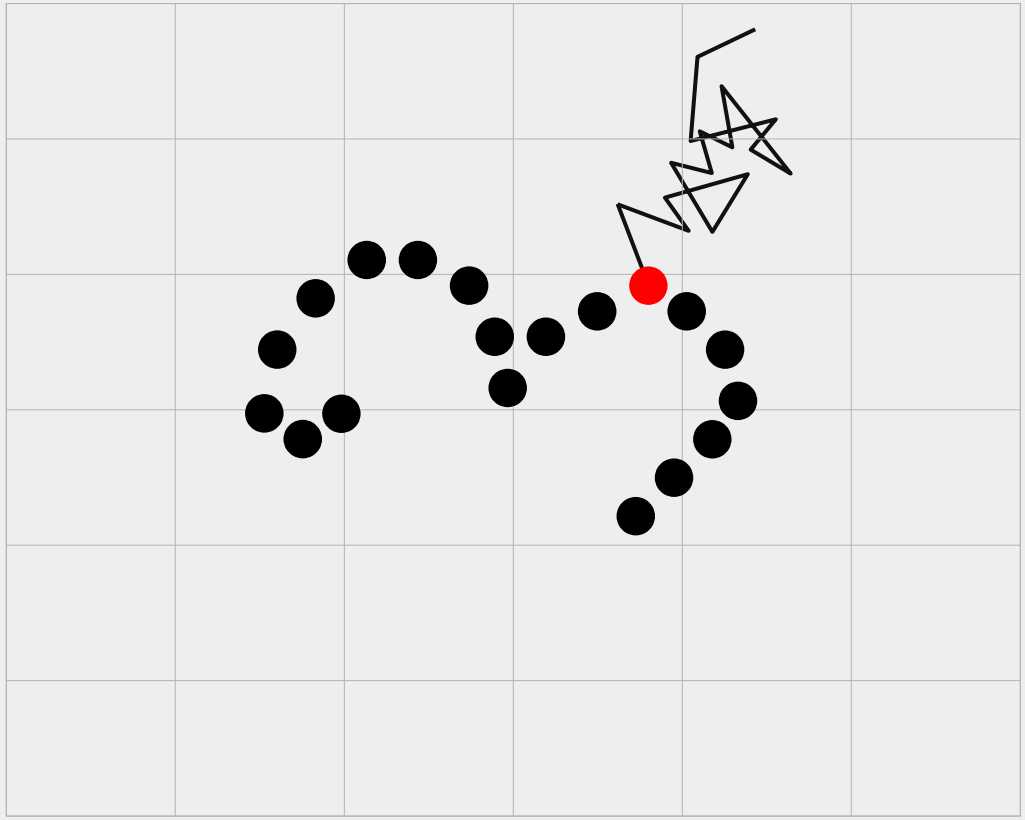


$\boldsymbol{x}_0$

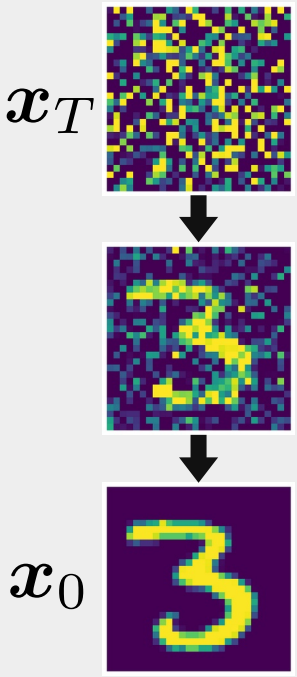
$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$

$= \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{\mu}_t, \beta_t)$

$\boldsymbol{x}_T$ $\mathcal{N}(\boldsymbol{x}_T; \mathbf{0}, 1)$

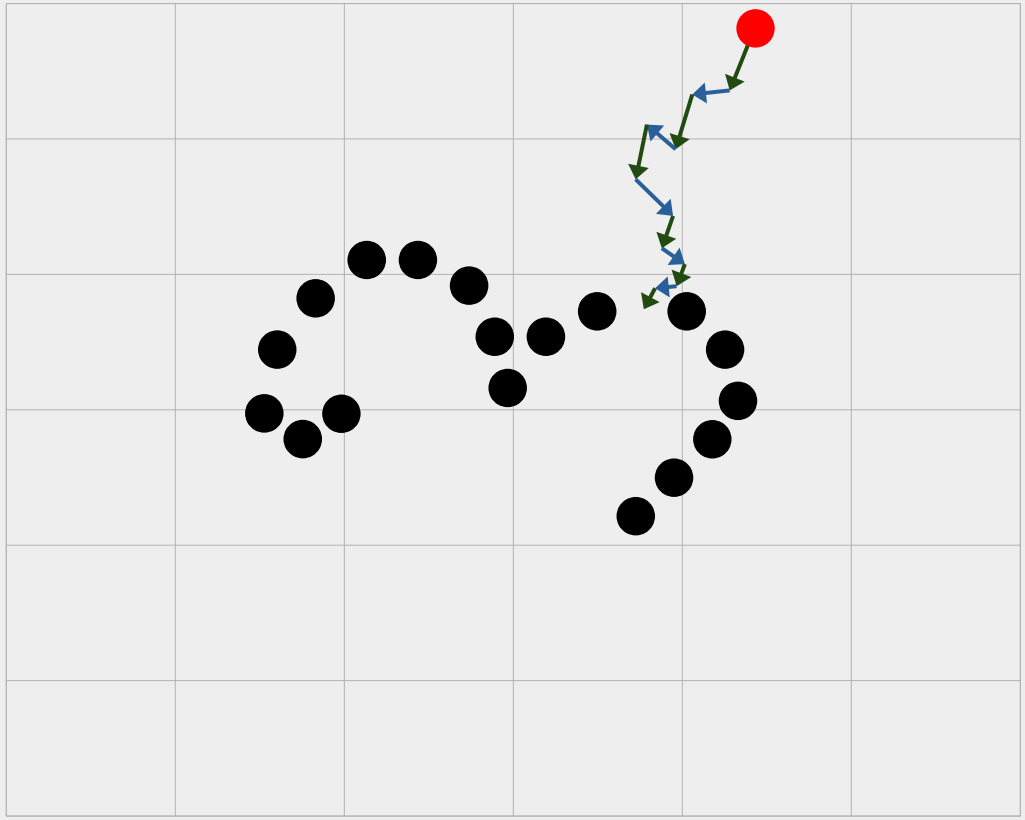


DDPM Intuition (Reverse process)



$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_{t-1}; \tilde{\boldsymbol{\mu}}, \tilde{\beta})$$

Model prediction



DDPM Theory (Forward Process)

Adding Noise in T steps using:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t)$$

How to sample $\boldsymbol{x}_t$ in one step? We will use:

$$\alpha_t = 1 - \beta_t \ , \ \bar{\alpha}_t = \prod_{s=0}^{t} \alpha_s$$

$$\mathcal{N}(x; \mu_1, \sigma_1^2) * \mathcal{N}(x; \mu_2, \sigma_2^2) = \mathcal{N}\big(x; \mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2\big)$$

DDPM Theory (Forward Process)

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t)$$

$$\Rightarrow \boldsymbol{x}_t = \sqrt{1-\beta_t}\boldsymbol{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\varepsilon}_t$$

Where $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, 1)$

Performing one recursive step:

$$\boldsymbol{x}_t = \sqrt{\alpha_t}\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\alpha_t\beta_{t-1}}\boldsymbol{\varepsilon}_{t-1} + \sqrt{\beta_t}\boldsymbol{\varepsilon}_t$$

$$= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\boldsymbol{\varepsilon}$$

DDPM Theory (Forward Process)

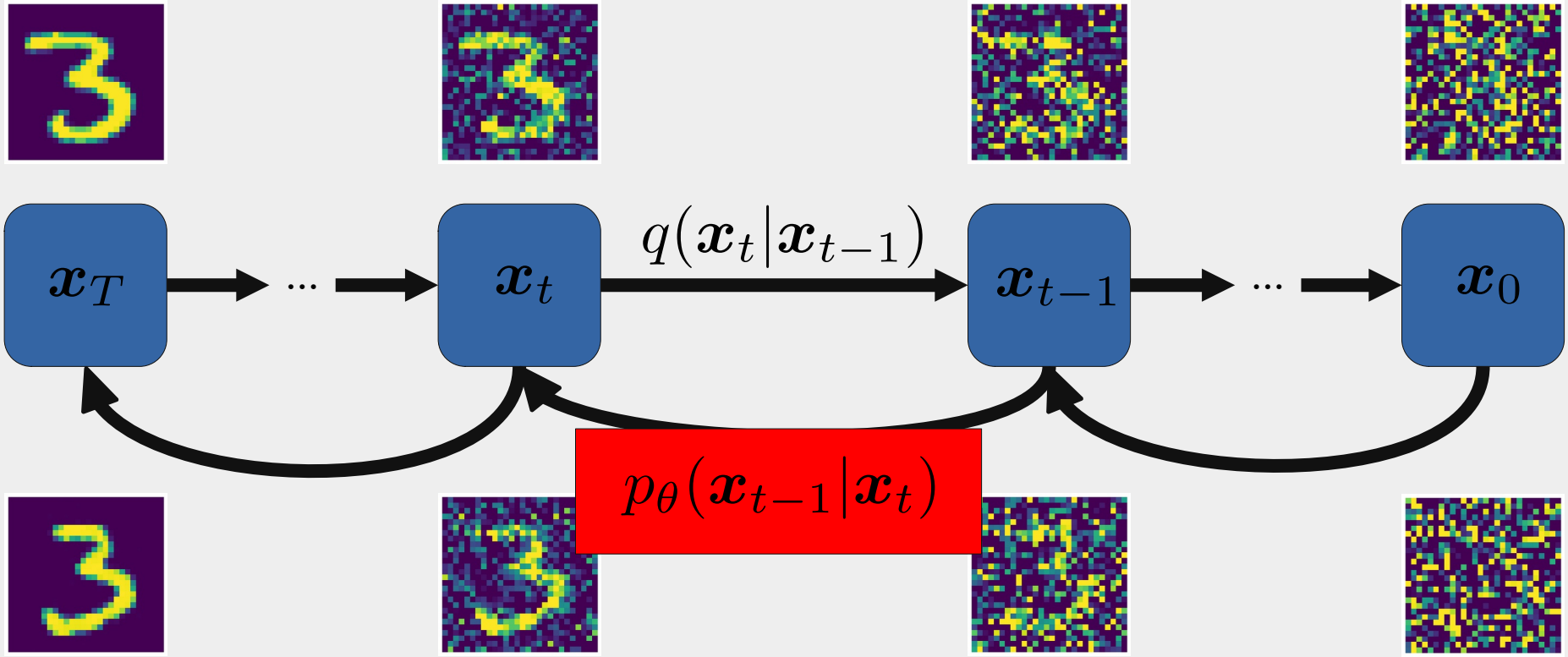
After t recursion steps:

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}$$

$\bar{\alpha}_t$ is precomputable $\Rightarrow$ we can compute $\boldsymbol{x}_t$ directly from $\boldsymbol{x}_0$

$$\boldsymbol{x}_t \sim q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t, \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, 1-\bar{\alpha}_t)$$

DDPM Theory (Reverse process)



DDPM Theory (Reverse process)

We need $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$. Bayes' theorem:

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)q(\boldsymbol{x}_t|\boldsymbol{x}_0) = q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0)q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)$$

The markov property of the forward process:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0) = q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$$

Since $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ is independent of $\boldsymbol{x}_{t-1}$:

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \propto q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)$$

DDPM Theory (Reverse process)

Second term is known from the forward process.

For the first term rewrite $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ in terms of $\boldsymbol{x}_{t-1}$:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t)$$

$$\propto \exp\left(-\frac{1}{2\beta_t}\left(\boldsymbol{x}_t - \sqrt{\alpha_t}\boldsymbol{x}_{t-1}\right)^2\right)$$

$$\propto \exp\left(-\frac{\alpha_t}{2\beta_t}\left(\boldsymbol{x}_{t-1} - \frac{\boldsymbol{x}_t}{\sqrt{\alpha_t}}\right)^2\right)$$

DDPM Theory (Reverse process)

$$\implies q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$$

$$\propto \mathcal{N}\left(\boldsymbol{x}_{t-1}; \frac{\boldsymbol{x}_t}{\sqrt{\alpha_t}}, \frac{\beta_t}{\alpha_t}\right) \mathcal{N}\left(\boldsymbol{x}_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0, 1-\bar{\alpha}_{t-1}\right)$$

The product of two gaußians is proportional to a gaußian with the variance and mean:

$$\frac{1}{\sigma^2} = \frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2} \quad , \quad \frac{\mu}{\sigma^2} = \frac{\mu_1}{\sigma_1^2} + \frac{\mu_2}{\sigma_2^2}$$

DDPM Theory (Reverse process)

Analytic tractable form:

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \propto \mathcal{N}\left(\boldsymbol{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\boldsymbol{x}_t, \boldsymbol{x}_0), \tilde{\beta}_t\right)$$

With variance and mean:

$$\tilde{\beta}_t = \frac{\beta_t(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}$$

$$\tilde{\boldsymbol{\mu}}_t(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\boldsymbol{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x}_t$$

DDPM Theory (Reverse process)

Express the mean in terms of $\boldsymbol{x}_0$ only:

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon} \implies \boldsymbol{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\boldsymbol{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}(\boldsymbol{x}_t, t)\right)$$

Simplified formula for the mean:

$$\tilde{\boldsymbol{\mu}}_t(\boldsymbol{x}_t) = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\varepsilon}(\boldsymbol{x}_t, t)\right)$$

$\boldsymbol{\varepsilon}(\boldsymbol{x}_t, t)$ now depends on $\boldsymbol{x}_t$ and t
$\implies$ perfect quantity to be predicted by our model $\boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t)$

Implementation (Noise schedule)

- Cosine schedule

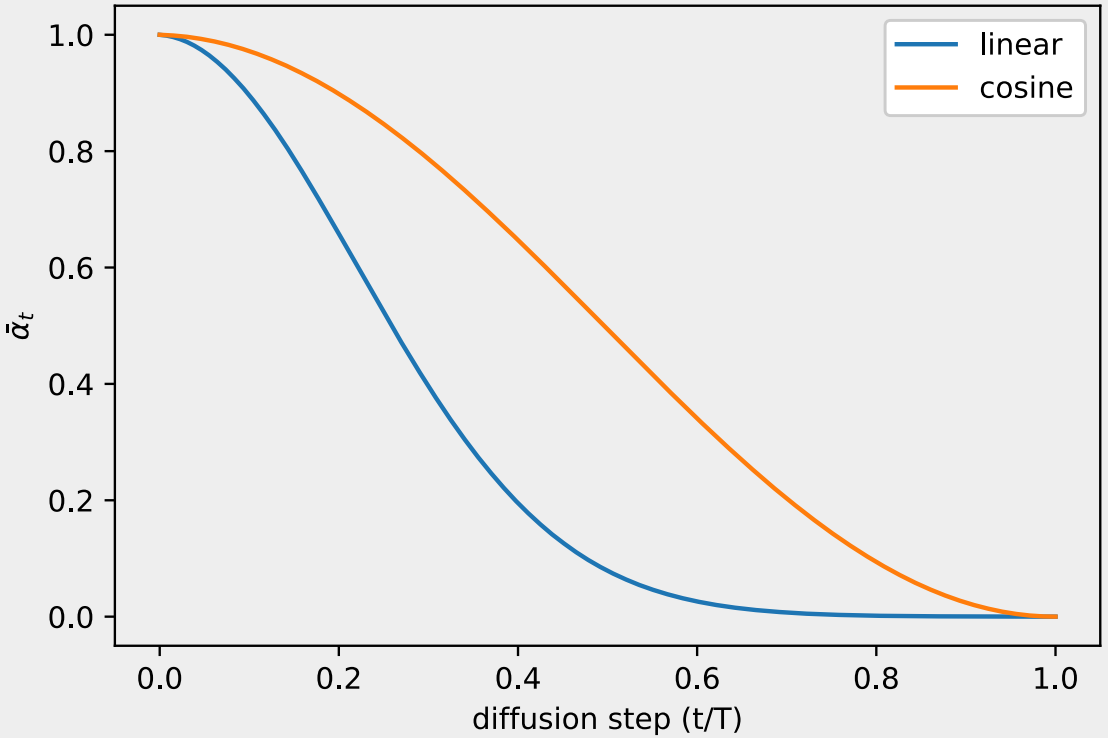
$$\bar{\alpha}_t = \frac{f(t)}{f(0)},$$

$$f(t) = \cos\left(\frac{t/T+s}{1+s} \cdot \frac{\pi}{2}\right)^2$$

- Linear schedule

$$\beta_t = \beta_1 + \frac{\beta_T - \beta_1}{T} t$$

$$\beta_1 = 10^{-4},\ \beta_T = 0.02$$



Nichol, A., & Dhariwal, P. (2021)

Implementation (Noise schedule)

linear

cosine

Nichol, A., & Dhariwal, P. (2021)

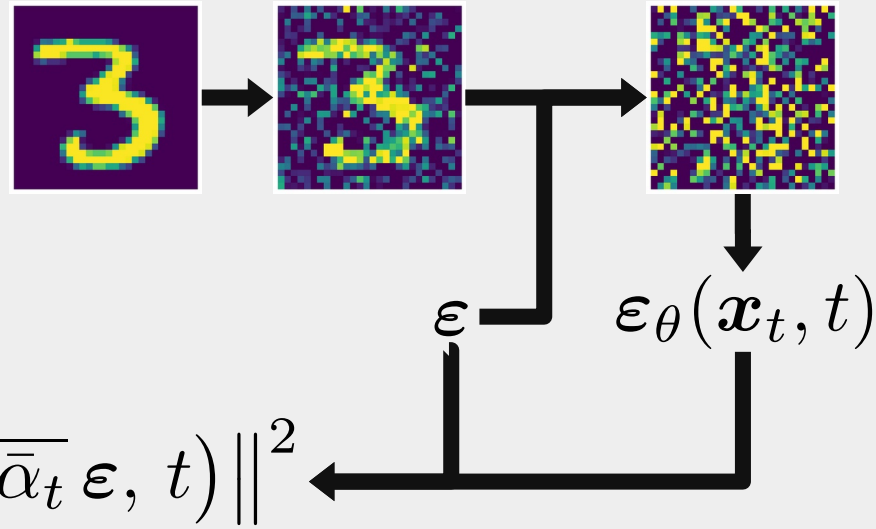
Implementation (Algorithm)

Training

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

$$\nabla_\theta \left\| \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\, \boldsymbol{x}_0 + \sqrt{1 - \bar{\alpha}_t}\, \boldsymbol{\varepsilon},\, t\right) \right\|^2$$

6: until converged



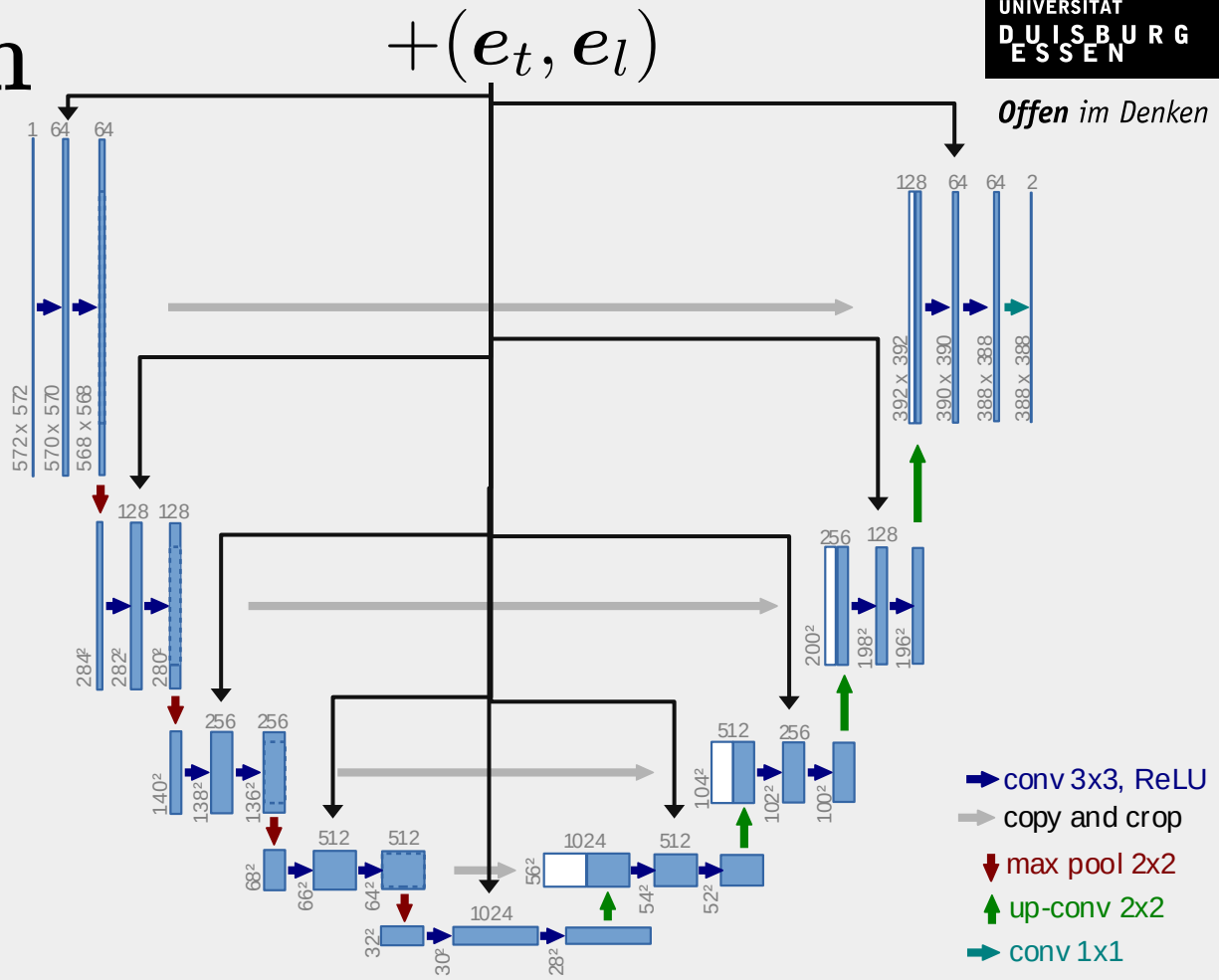
Implementation (Algorithm)

Sampling

$\boldsymbol{x}_T \sim \mathcal{N}(0, 1)$
for $t = T, \ldots, 1$ **do**
    $\boldsymbol{z} \sim \mathcal{N}(0, 1)$ if $t > 1$, else $\boldsymbol{z} = 0$
    $\boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t) \right) + \sigma_t \boldsymbol{z}$
end for
return $\boldsymbol{x}_0$

Implementation (Unet-model)

- Encoder-decoder structure
- Convolution blocks
- Down- & upsampling
- Skip connections



Ronneberger, O., Fischer, P., & Brox, T. (2015)

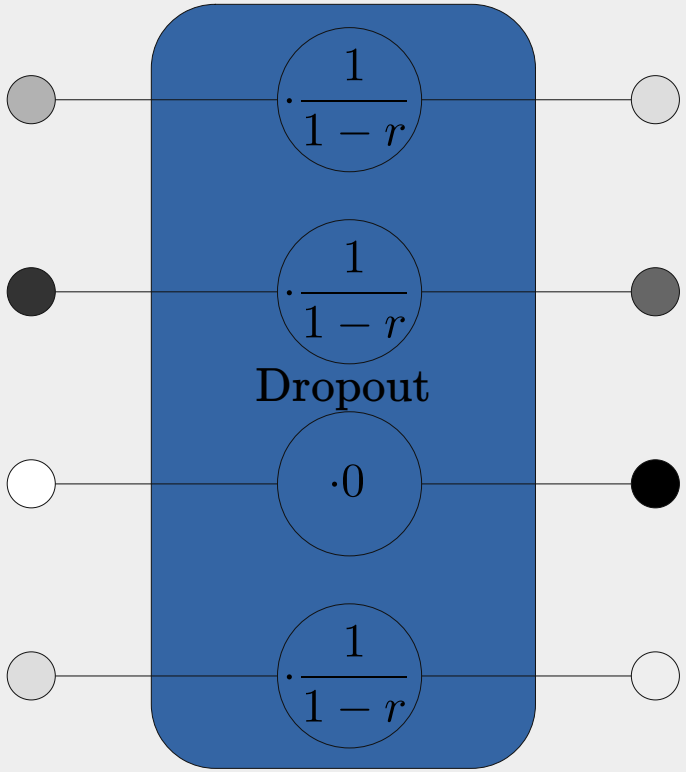
Implementation (Embedding)

- Timestep t and digit label l necessary
- Input embedding vectors:
  $\boldsymbol{e}_t = (t/T, \beta_t, \bar{\alpha}_t) \qquad \boldsymbol{e}_l = (\delta_{0l}, \delta_{1l}, \ldots, \delta_{8l}, \delta_{9l})$

1) Passed into their own dense neural network
2) Added to the first featuremap of each convolution block

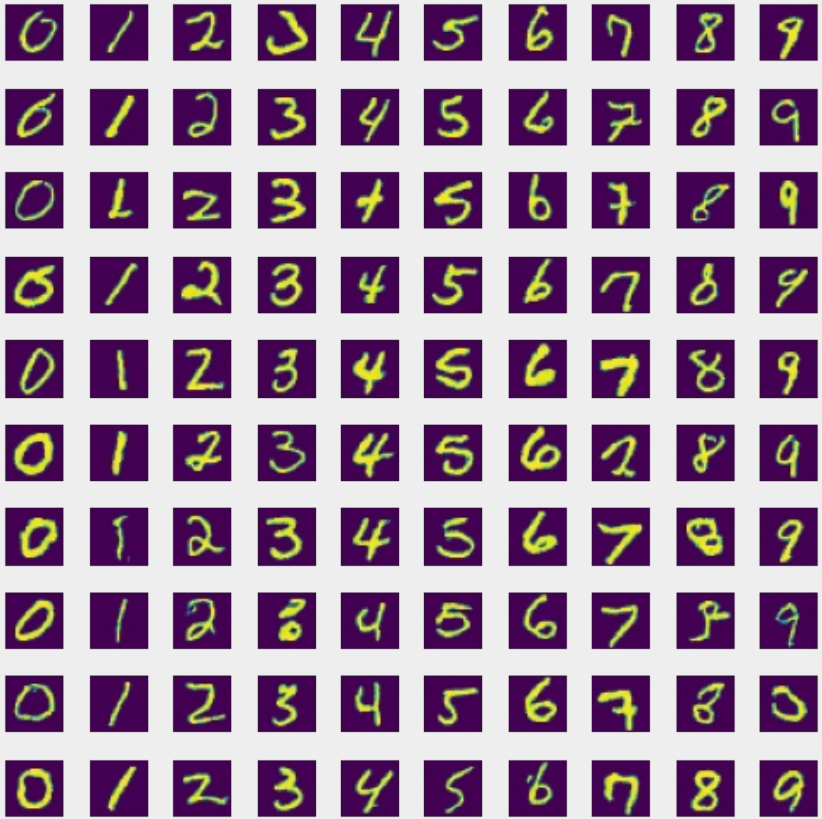
Implementation (Dropout layer)

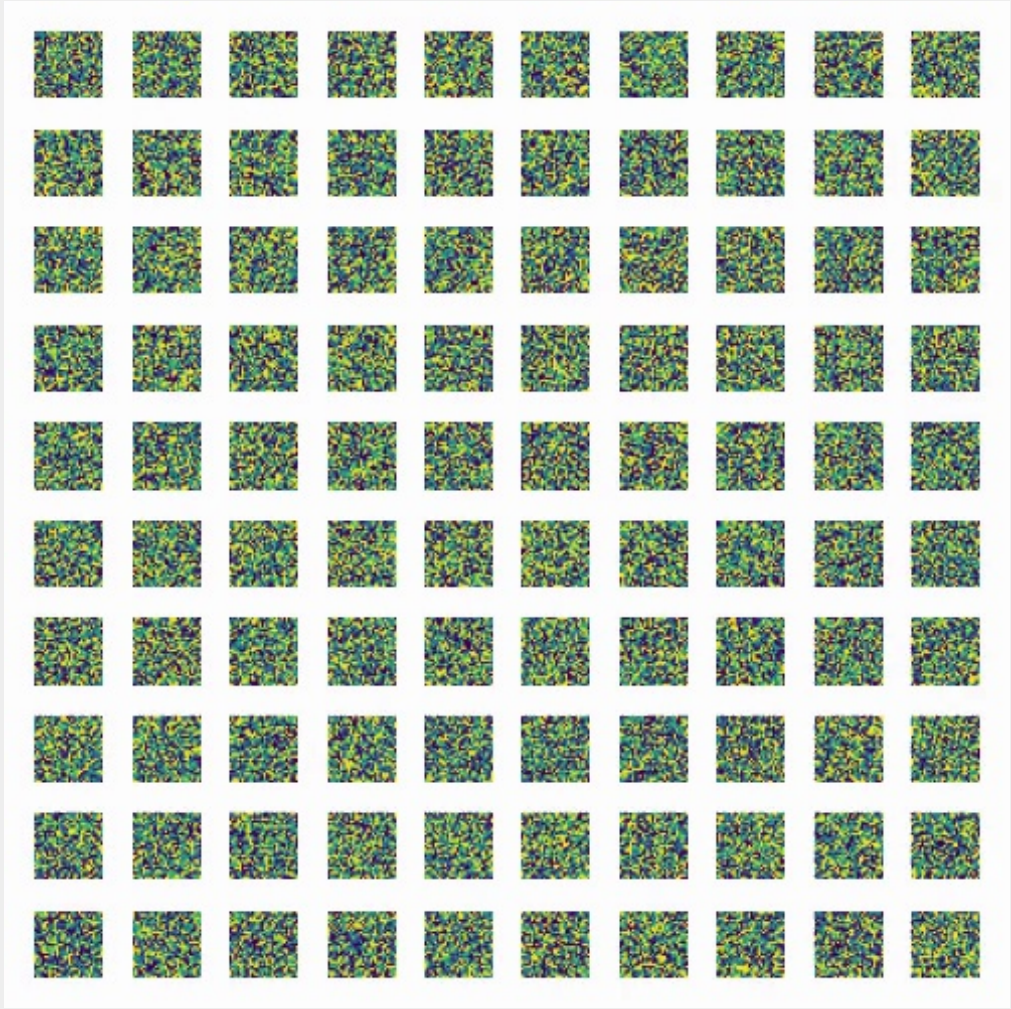
- Randomly disables a fraction r of the neurons
- Helps to prevent overfitting
- One layer is added to each convolution block



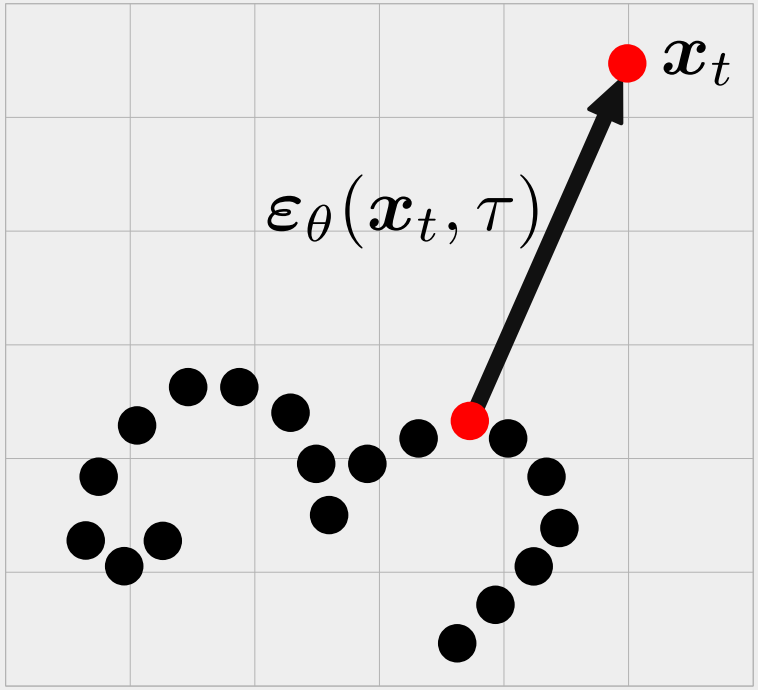
Implementation details & results

```
IMG_SHAPE = (28, 28, 1)
BATCH_SIZE = 64
TRAINING_SET_SIZE = 60000
VALIDATION_SET_SIZE = 10000
EPOCHS = 64
NOISE_STEPS = 1000
EMBED_DIM = 256
TIME_EMBED_NEURONS = [64,64,64,128]
LABEL_EMBED_NEURONS = [64,64,128]
ENCODER_BLOCK_FILTERS = [128,256,512]
DECODER_BLOCK_FILTERS = [256,128]
CONV_KERNEL_SIZE = 3
DROPOUT_RATE = 0.1
```





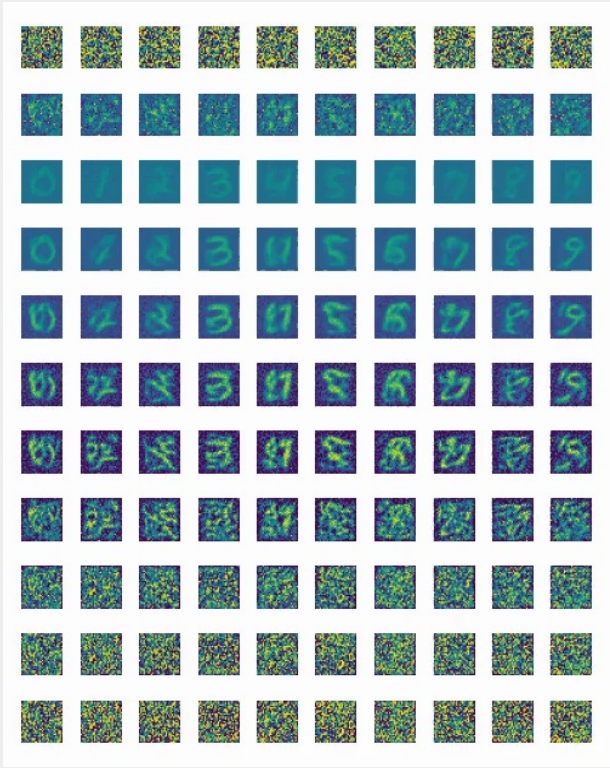
Implementation (Visualization)



$$y(t, \tau) = x_t - \varepsilon_\theta(x_t, \tau)$$

x_t

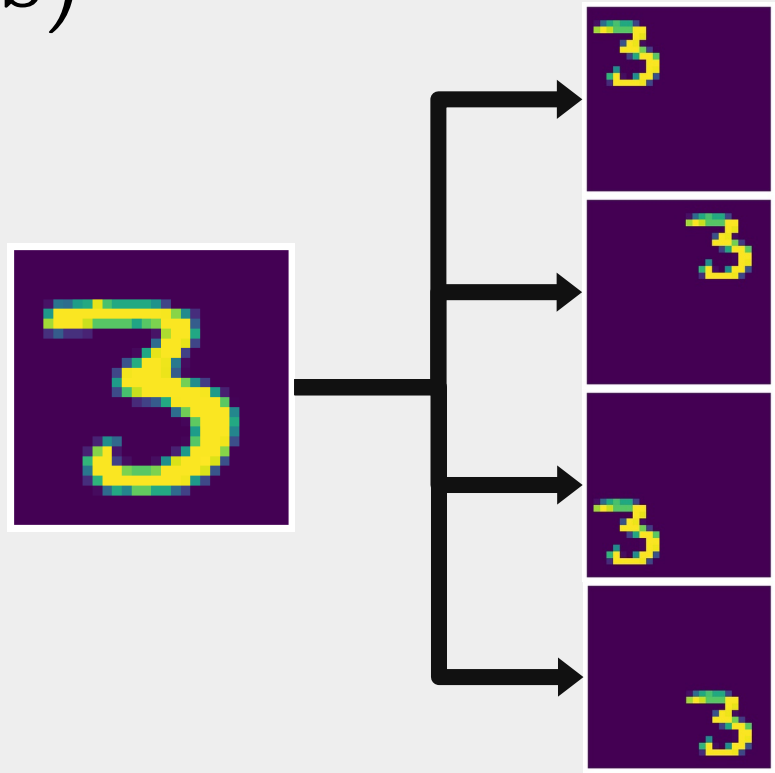
$y(t, \tau)$



$\tau = T$

$\tau = 0$

Implementation (Shifted digits)



Implementation (Shifted digits)

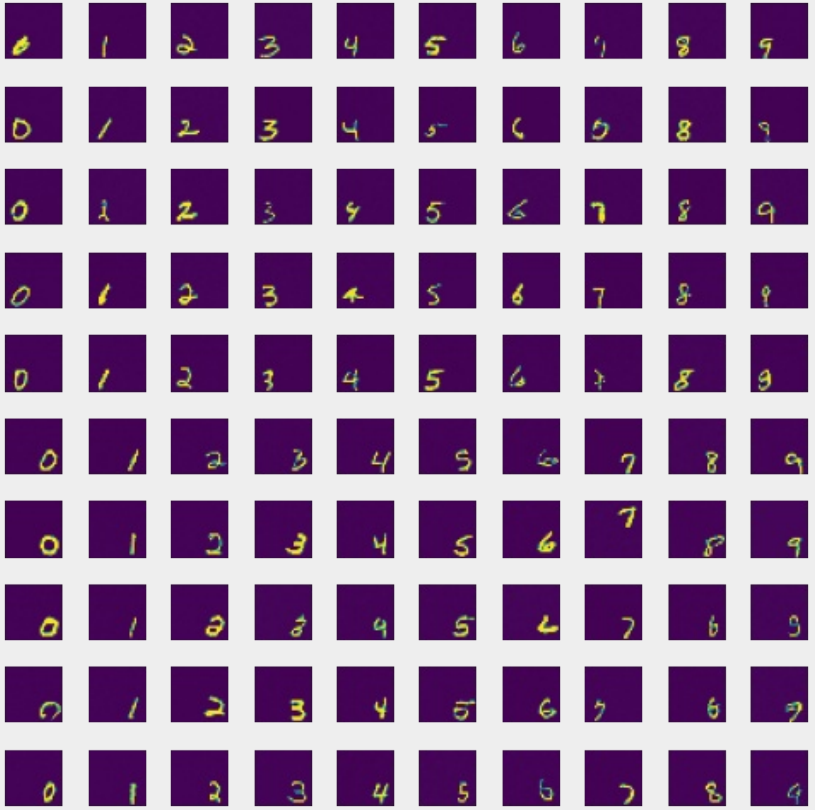
- Modified label embedding:

  $\boldsymbol{e}_l = (\delta_{0l}, \delta_{1l}, \ldots, \delta_{8l}, \delta_{9l}, \delta_{0p}, \delta_{1p}, \delta_{2p}, \delta_{3p})$

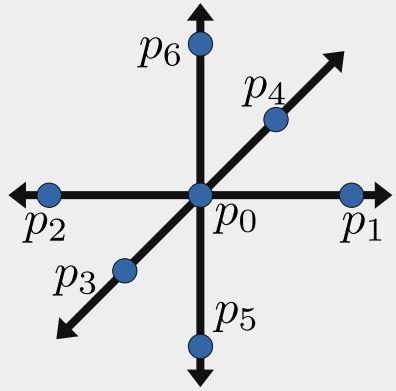
- Position encoded in $p \in \{0,1,2,3\}$
- Modified training data:
  - Downscaled and shifted
  - For 7 only positions p=0,1 are labeled
  - Position labels are left out for half of the data

Implementation (Shifted digits)

- Model was able to generalize
  - Generation of digit 7 at positions that were not labeled during training
- Not fully accurate



Parameter study (method)



Vary the following parameters:

- Noise steps
- Batch size
- Size of the embedding network
- Number of filters of the convolution blocks
- Kernel size of the convolution layers
- Dropout rate

Parameter study (method)

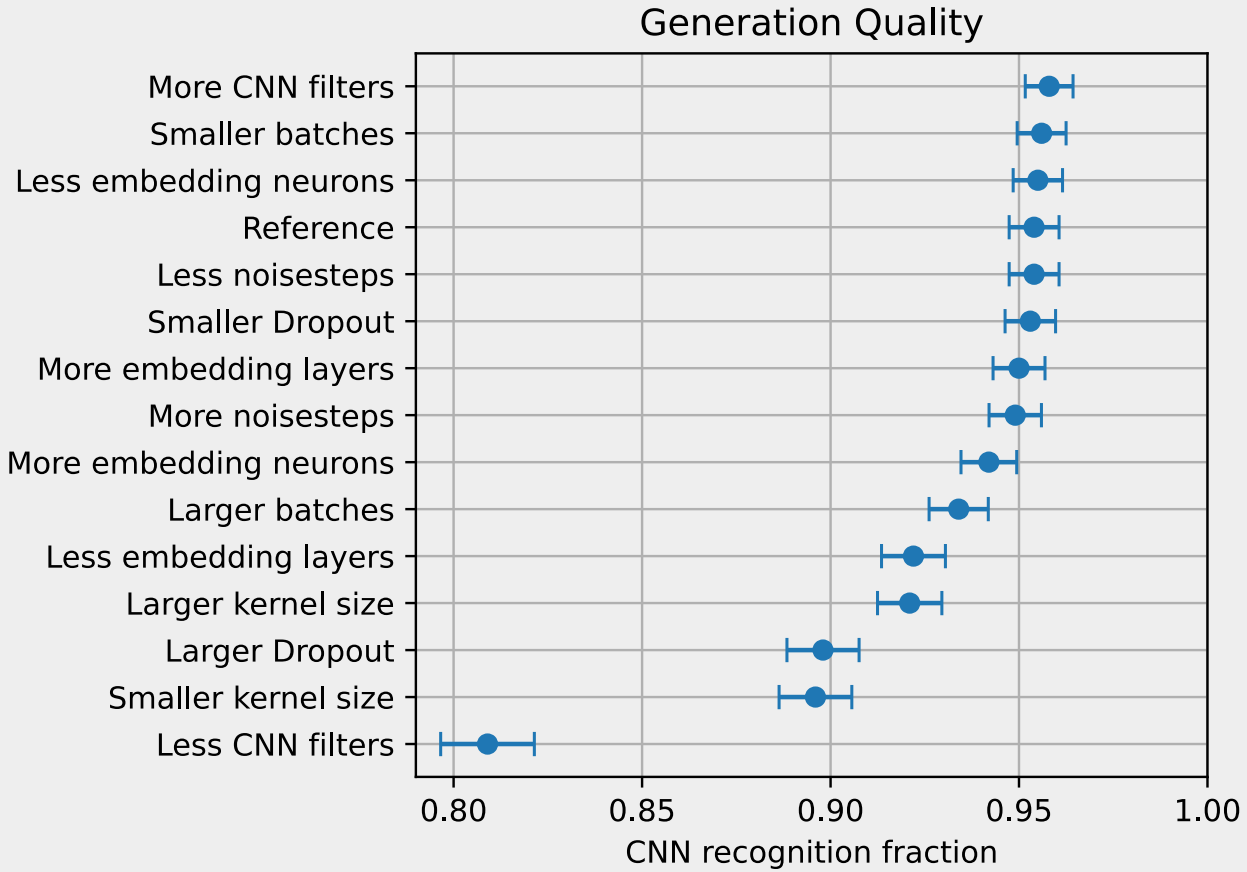
UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

How to evaluate the trained model?

- Generate N images with the diffusion model
- Use second CNN trained on MNIST to recognize digits (the number of correctly recognized digits is n)
- The score for the trained model is given by: $s=n/N$

Parameter study (results)



Parameter study (discussion)

- CNN is able to recognize even “ugly” digits

  → almost all digits are recognized → small score differences

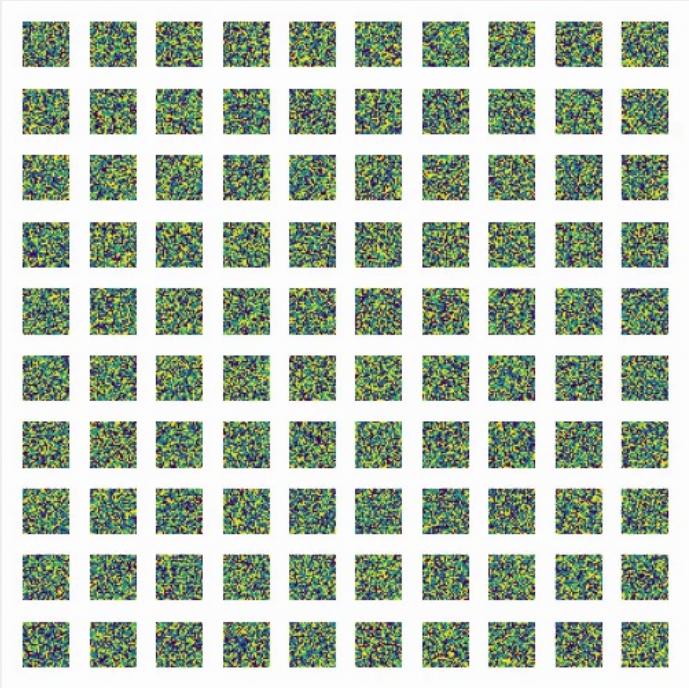
  → train CNN to decide if an image is real or generated

- Training loss of models shows high fluctuations from run to run

  → average over multiple training runs for the same parameter set

Summary

- Diffusion processes can be used for image denoising
  - This can even be used to generate new images
- A neural network is employed to remove the noise
- Labels can be used for targeted image generation



Outlook

- DDIM
  - Reverse process $\rightarrow$ deterministic (i.e. one step from noise to image is possible)
- GAN
  - Use a CNN as a discriminator $\rightarrow$ help with training
- Better representation of transformations

  $\rightarrow$ symbolic regression