

8 Generative Artificial Intelligence

Leon Herrmann

Stefan Kollmannsberger

Chair of Data Engineering in Construction

Bauhaus-Universität Weimar

Paris, February 2025

Deep Learning in Computational Mechanics – an introductory course,

Herrmann et al. 2025



website



book



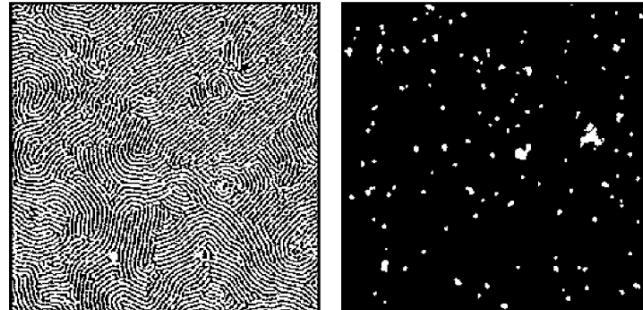
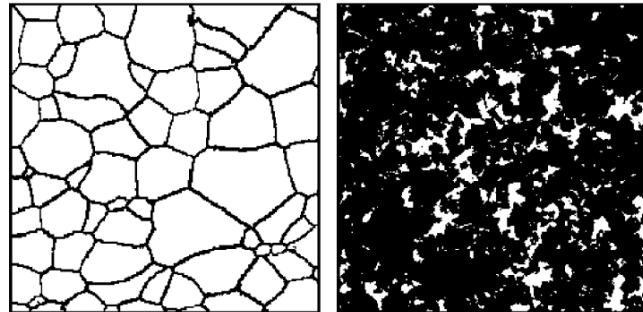
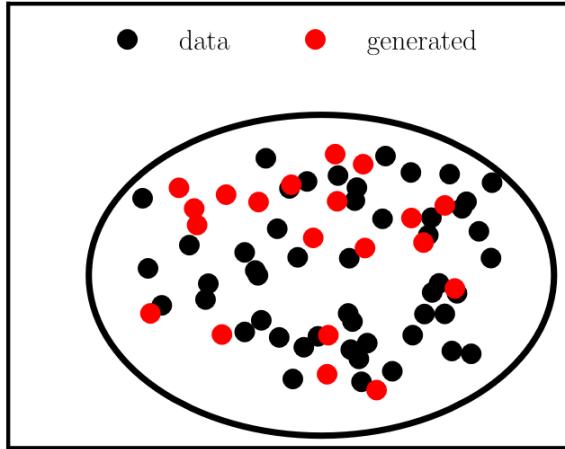
Contents

- [7 Material Modeling with Neural Networks](#)
- [8.1 Autoencoders](#)
- [8.2 Variational Autoencoder](#)
- [8.4 Autoencoders in Computer Vision: U-Net](#)
- [8.3 Generative Adversarial Networks](#)
- [8.5 Diffusion Models](#)
- [8.6 Transformers](#)
- [8.7 Applications in Computational Mechanics](#)
 - [8.7.1 Data Generation](#)
 - [8.7.3 Anomaly Detection](#)
 - [8.7.4 Conditional Generation](#)
- [9 Inverse Problems & Deep Learning](#)

8 Generative Artificial Intelligence

Generative modeling

- Generate new data points that resemble a given dataset (without simply reproducing given data points)
- Example: Generate new rim designs given a set of rims



- Variational autoencoders, generative adversarial networks, diffusion models, flow-based models, transformer models

8.1 Autoencoders

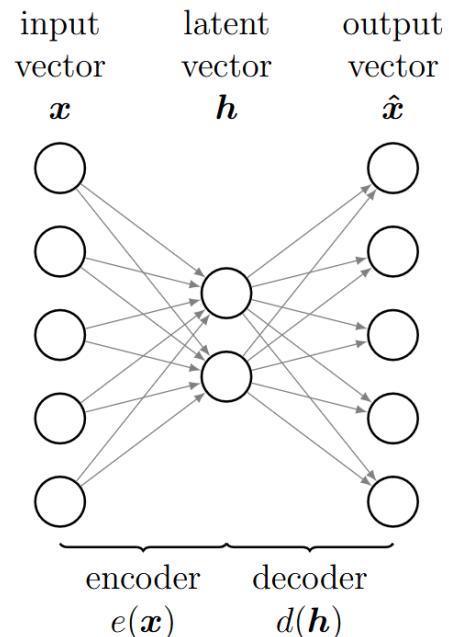
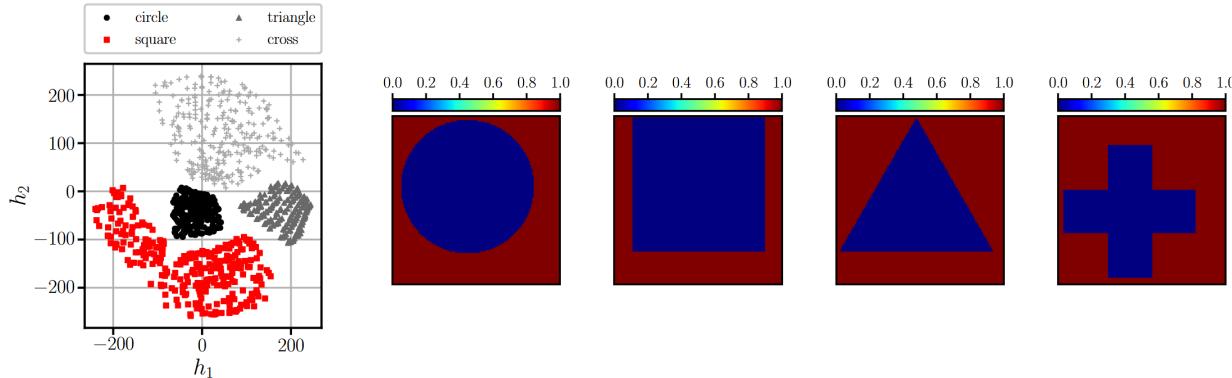
Autoencoders learn a **compression** (and the corresponding decompression)

- **Encoder** e projects data from **high-dimensional** input x to **low-dimensional latent space** h
- **Decoder** d reconstructs x from the latent space h , i.e., $\hat{x} = d(e(x))$
- Cost function

$$C = \|d(e(x)) - x\|^2$$

This is a dimensionality reduction technique (and not a generative model):
Generation is theoretically possible by randomly sampling the latent space & decoding it

- Autoencoders are problematic for generation, due to discontinuities between different objects in the latent space



8.2 Variational Autoencoder

Variational autoencoders (VAEs) are a variation of an autoencoder for generative modelling

- After training the autoencoder, the **latent space** $h \sim \mathcal{N}(\mu, \sigma^2)$ is sampled randomly and decoded $\hat{x} = d(h)$
- Discontinuities** in the latent space are overcome μ is the mean and σ the standard deviation of a normal distribution
- By mapping to $\mu(x), \sigma(x)$ instead of a deterministic h (and sampling $h \sim \mathcal{N}(\mu, \sigma^2)$ during training)
- The probabilistic framework is trained with the **Kullback-Leibler (KL) divergence**

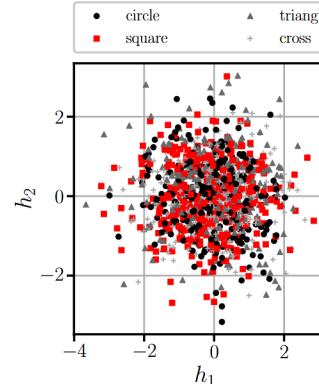
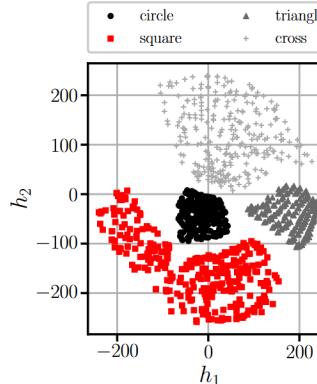
$$\text{KL}(\mathcal{N}(\mu, \sigma^2) || \mathcal{N}(0, I)) = \frac{1}{2} \sum_{i=1}^d (\sigma_i^2 + \mu_i^2 - \ln \sigma_i^2 - 1)$$

Penalizes distribution $p(h|x)$ for deviating from the standard normal distribution $\mathcal{N}(0, I)$: avoids μ being large and σ being small

- Sampling operations are non-differentiable → **reparametrization**

$$h = \mu(x) + \sigma(x)\epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

Trained with
reconstruction loss



Trained with Kullback-
Leibler divergence

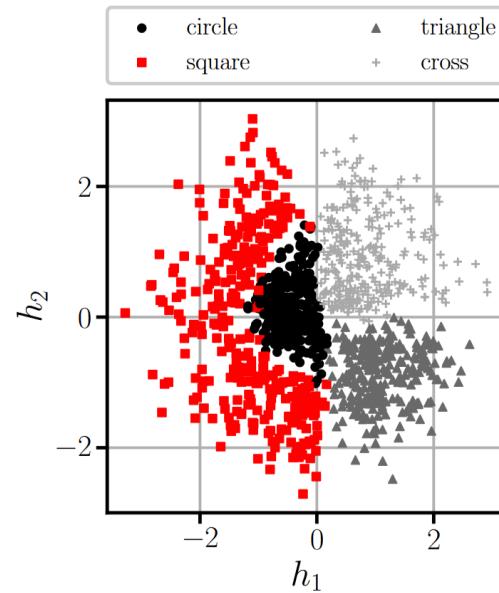
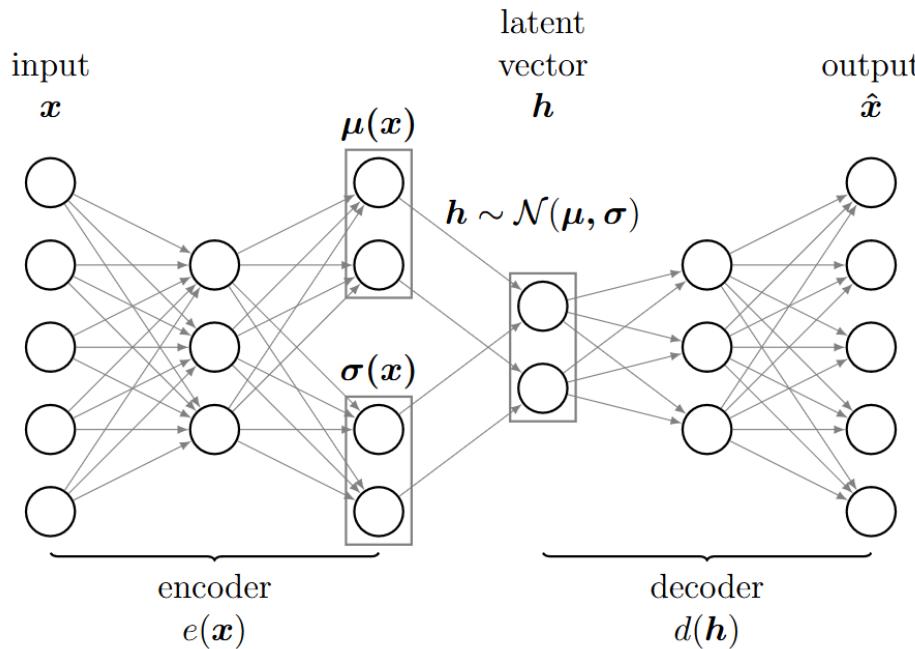
Encodings are distributed
randomly around zero (and
entangled)

8.2 Variational Autoencoder

Cost function of variational autoencoder as combination between **reconstruction loss** & **Kullback-Leibler divergence**

$$C = \|x - \hat{x}\|^2 + \text{KL}(\mathcal{N}(\mu(x), \sigma), \mathcal{N}(0, I))$$

where the latent space prediction is made as $h = \mu(x) + \sigma(x)\epsilon$, $\epsilon \sim \mathcal{N}(0, I)$



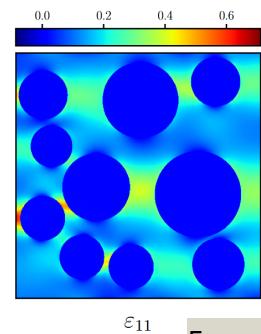
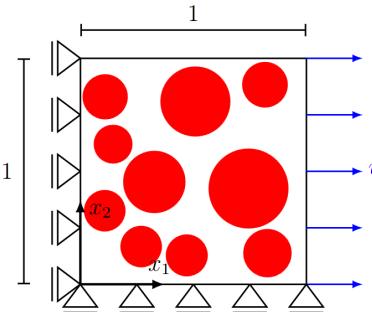
Disentangled and continuous
latent space (with meaningful
clusters centered around zero)

8.4 Autoencoders in Computer Vision: U-Net

Autoencoders for surrogate modeling

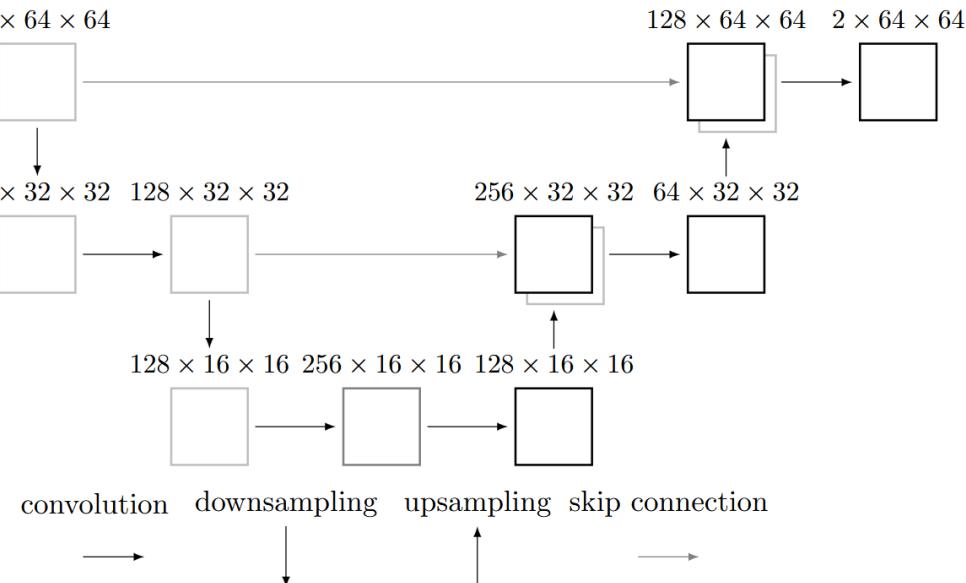
- Extract the essence of the input
- (Potentially) have to reconstruct important features

- **U-Net**
 - Encodes the input to a latent space
 - Adds **skip connections**
 - Features do not have to be relearnt
 - Avoids vanishing/exploding gradients



Example (learning strain distributions) from Chapter 3

*U-Net: Convolutional Networks for Biomedical Image Segmentation,
Ronneberger et al. 2015*

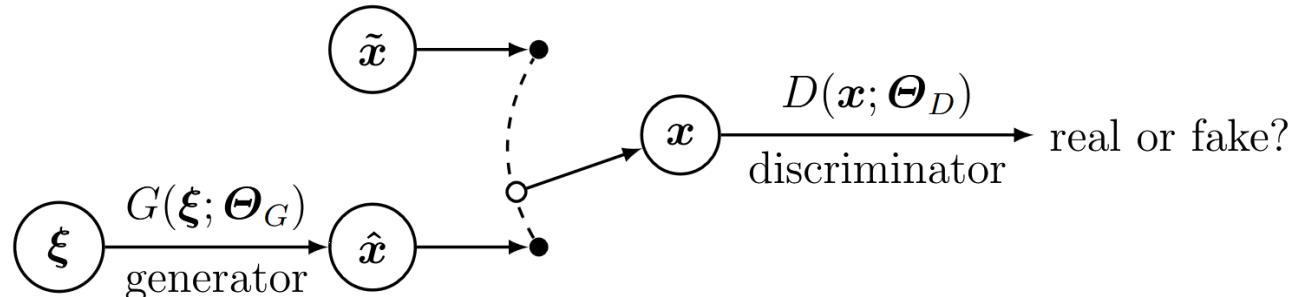


8.3 Generative Adversarial Networks

Generative Adversarial Nets, Goodfellow et al. 2014

Generative adversarial networks (GANs) learn to generate data through a **two-player game**

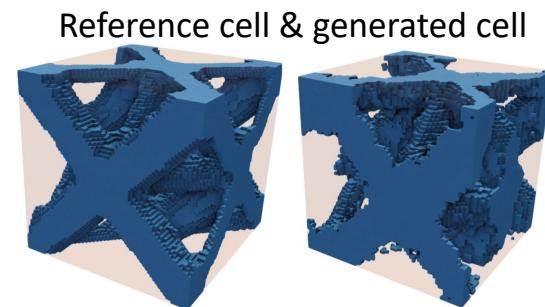
- The **generator** $G_{NN}(\xi; \Theta_G)$ creates fake samples \hat{x} from random noise ξ
- The **discriminator** $D_{NN}(x; \Theta_D)$ assesses if a sample x is **real** or **fake** (i.e., generated by G_{NN})
- This is quantified by the predicted probability score $\hat{p} = D_{NN}(x; \Theta_D)$



- Thus: the generator aims to fool the discriminator
- The two-player game is formulated with the cost function

$$C = \frac{1}{m_D} \sum_{i=1}^{m_D} \log(D_{NN}(\tilde{x}_i; \Theta_D)) + \frac{1}{m_G} \sum_{i=1}^{m_G} \log(1 - D_{NN}(G_{NN}(\xi_i; \Theta_G); \Theta_D))$$

- Handled with a **minimax optimization** $\min_{\Theta_G} \max_{\Theta_D} C$



Convergence is reached
when $\hat{p} = 0.5$ for all samples

8.5 Diffusion Models

Diffusion models learn to reverse a gradual noising procedure

- A **forward diffusion process** q (slowly adds random noise to data x)

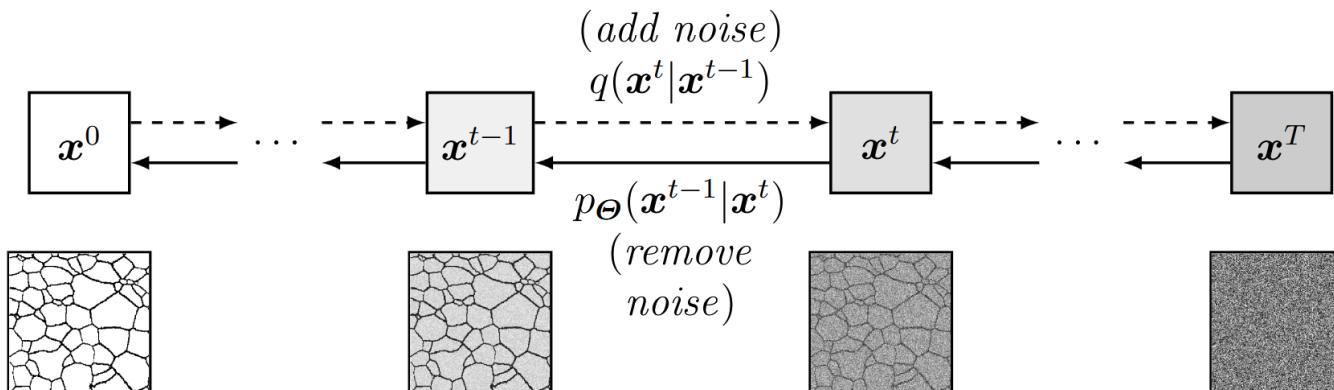
$$q(x^t | x^{t-1}) = \mathcal{N}(x^t; \sqrt{1 - \beta^t} x^{t-1}, \beta^t I)$$

- A **reverse denoising diffusion process** p_Θ (a neural network learns to reverse the diffusion process)

$$p_\Theta(x^{t-1} | x^t) = \mathcal{N}(x^{t-1}; \mu_\Theta(x^t, t), \Sigma_\Theta(x^t, t))$$

$\mu_\Theta, \Sigma_\Theta$ can be described
by neural networks

$$p_\Theta(x^{0:T}) = p(x^T) \prod_{t=1}^T p_\Theta(x^{t-1} | x^t)$$



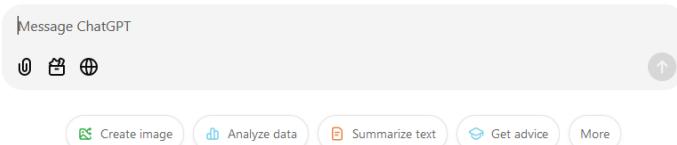
8.6 Transformers

Attention Is All You Need, Vaswani et al. 2017

Transformers operate on sequential data (typically text in natural language processing, e.g., ChatGPT):

- Text classification
- Text generation
- Summarization
- Question Answering
- Machine translation

What can I help with?



Key ingredients

- **Self-attention** (to understand the context in long sequences)
- **Positional encoding** (enables parallelism, despite sequential input)
- **Pretraining** (cheaper applications to new tasks, e.g., GPT: Generative pre-trained transformer)

Let's understand how transformers work on the following translation task

In winter, it snows.

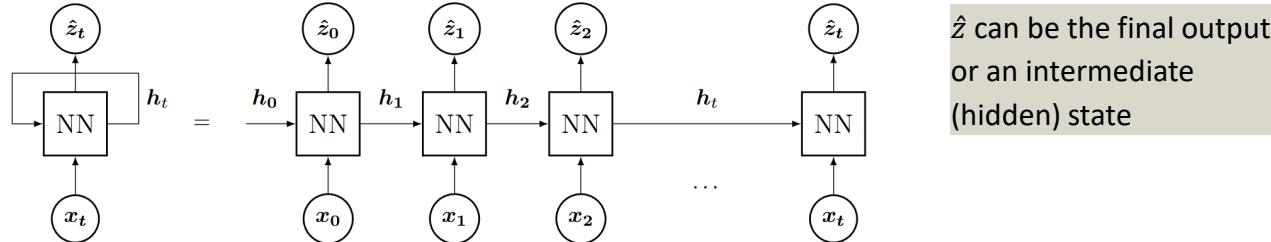
→

Im Winter schneit es.

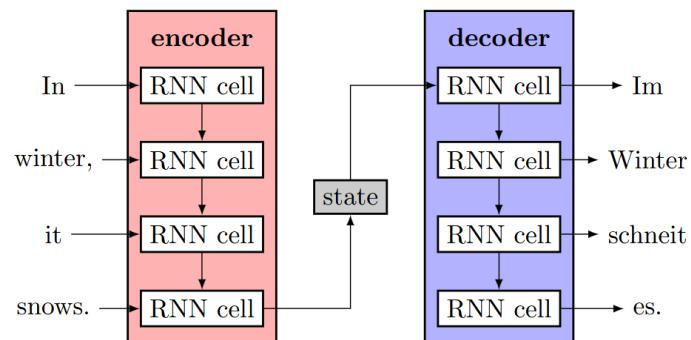
8.6 Transformers

Natural Language Processing with Transformers, Tunstall et al. 2022

In natural language processing, the state-of-the-art before transformers were recurrent neural networks (& variants)



The translation task could be handled by an **encoder-decoder** architecture (using two RNNs)



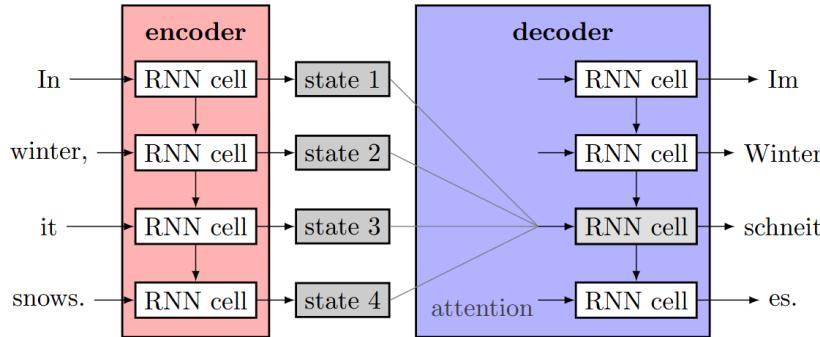
Problems:

- For long inputs, the hidden state cannot capture the meaning of the entire sequence
- The forward propagation cannot be parallelized

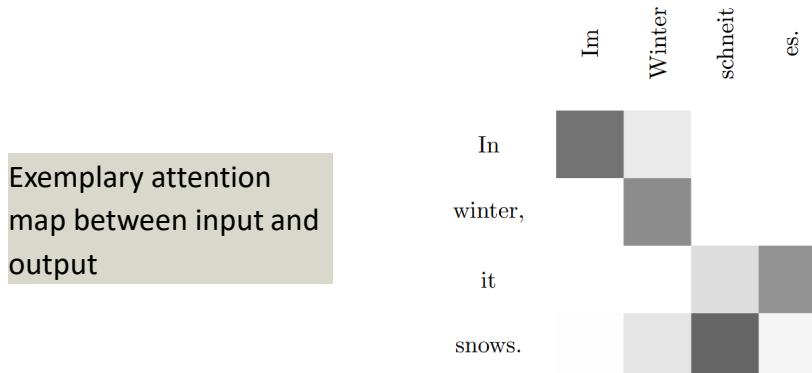
8.6 Transformers

Natural Language Processing with Transformers, Tunstall et al. 2022

To handle long-range dependencies, **attention** is introduced within the decoder



An attention map is applied between every hidden state (each **token**, i.e., word) and each RNN cell of the output



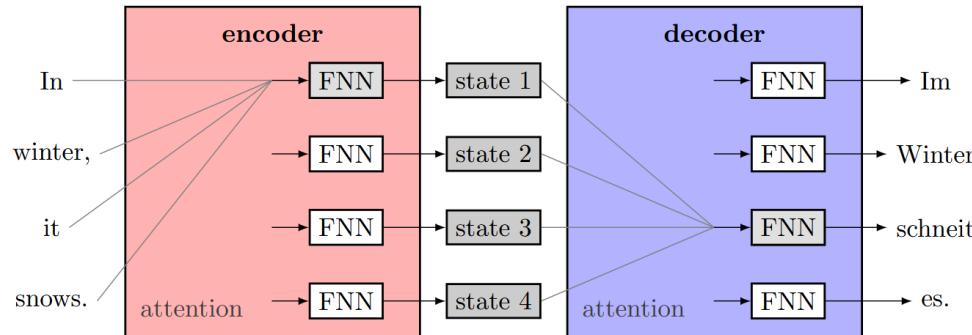
Problem:

- The forward propagation cannot be parallelized

8.6 Transformers

Natural Language Processing with Transformers, Tunstall et al. 2022

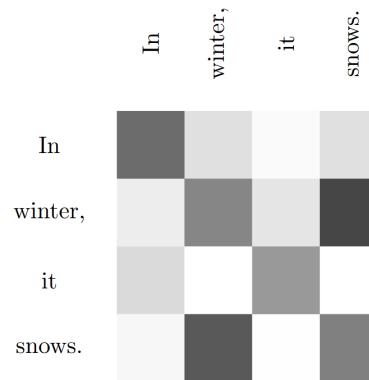
An additional attention mechanism in the encoder enables parallelism



Because each output token (decoder) attends to every input token (encoder), this mechanism is known as **cross-attention**.

But what about **self-attention**?

In self-attention, each token attends to every other token in the same sequence, capturing context and relationships between them.



8.6 Transformers

Implementation of **self-attention** in transformers

- Each input $x_i \in \mathbb{R}^{N_e}$ is queried and compared to a **key**, i.e., other inputs $x_j \in \mathbb{R}^{N_e}$
- The **query** Q and **key** K are obtained through linear transformations on $X \in \mathbb{R}^{N_w \times N_e}$

$$Q = XW^q \in \mathbb{R}^{N_w \times N_s}$$

$$K = XW^k \in \mathbb{R}^{N_w \times N_s}$$

- The **similarity** S describes the self-attention

$$S = QK^T \in \mathbb{R}^{N_w \times N_w}$$

- The similarity is applied to **values** V , i.e., transformed inputs x_i

$$V = XW^v \in \mathbb{R}^{N_w \times N_V}$$

- Through a **normalization**

Without softmax: unbounded

Softmax transforms S into a probability distribution

$$\tilde{z} = \text{softmax}\left(\frac{S}{\sqrt{N_s}}\right)V \in \mathbb{R}^{N_w \times N_V}$$

- Can be applied multiple times, yielding $\tilde{Z} = \{\tilde{z}_0, \tilde{z}_1, \dots, \tilde{z}_h\}$, which can be transformed to

$$z = \tilde{Z}W^z \in \mathbb{R}^{N_w \times N_z}$$

- As the attention mechanism is used multiple times within one layer, it is called **multi-headed attention**
- Learnable parameters are W^q, W^k, W^v, W^z

If key and query act on the same sequence: self-attention
Else: attention

In winter, it snows.



Similarity matrix S

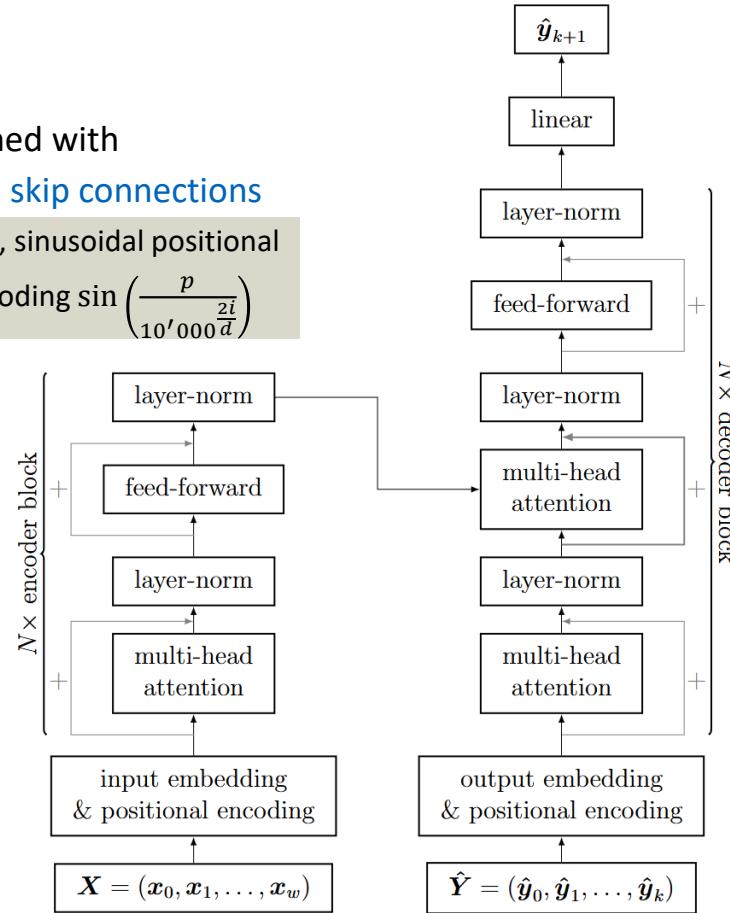
8.6 Transformers

Multi-headed self-attention layers are combined with

- Normalization layers, feed-forward layers, skip connections
- Positional encoding:
 - Extends input/output with unique positional identifier
- Input embedding (how to encode words):
 - Word2Vec, learnable

e.g., sinusoidal positional

$$\text{encoding } \sin\left(\frac{p}{10'000} \frac{2i}{d}\right)$$



In the original transformer formulation an encoder-decoder architecture was proposed:

- Encoder-only: text classification, extractive question answering
- Decoder-only: text generation, autoregressive question answering
- Encoder-decoder: machine translation, speech processing, data to text

8.6 Transformers

https://github.com/meta-llama/llama/blob/main/MODEL_CARD.md

Pretraining: Perform a training on a larger and more general dataset

- e.g., predict next word from previous words (get text from internet)

Domain adaption: Perform a second training on specific dataset

- e.g., (hand-crafted) Q&A texts to build a chatbot, or adaption to another language

Fine-tuning: combination with additional neural networks for specific task

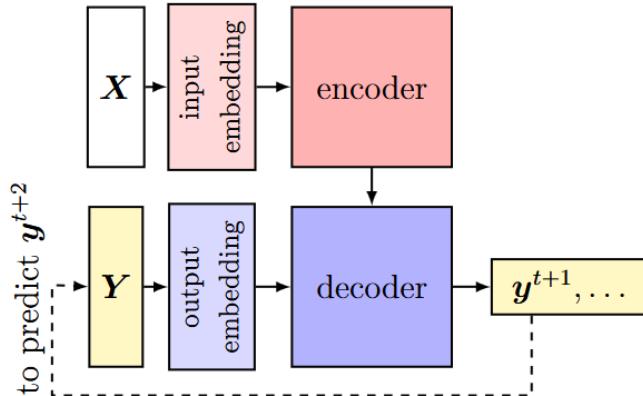
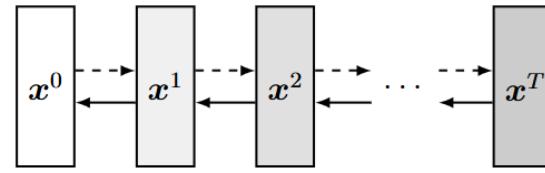
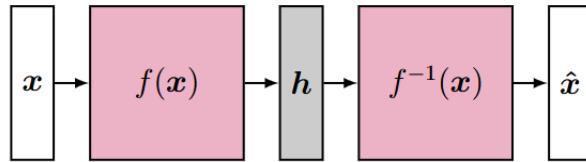
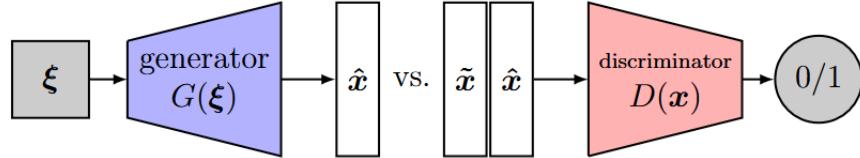
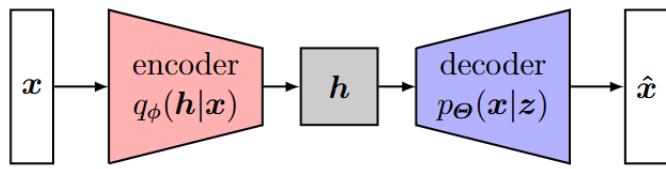
- e.g., classification layer appended to the hidden state, or using only the decoder for chatbots

Pretraining is the most expensive part

	parameters	model size	training time (on 2048 GPUs)	training cost
Llama 7B	7B	28 GB	4 days	0.1 M \$
Llama 13B	13B	52 GB	8 days	0.2 M \$
Llama 70B	70B	280 GB	35 days	1.1 M \$
GPT-3	165B	700 GB	17 days	4.6 M \$
GPT-4	100B-10T?	400 GB-40 TB	?	?

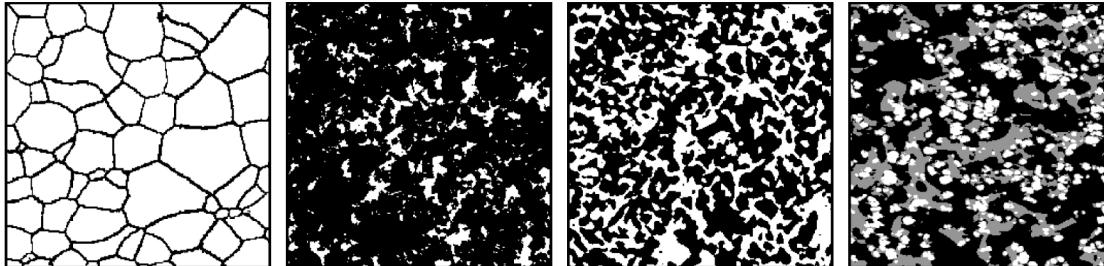
8 Generative Artificial Intelligence

What is what?

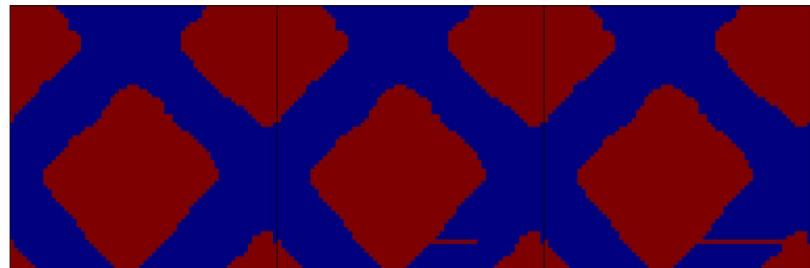


8.7 Applications in Computational Mechanics

- **8.7.1 Data Generation**
 - For surrogate models
 - As surrogate models



- **8.7.2 Generative Design & Design Optimization**
 - Nonlinear dimensionality reduction (optimization on latent vectors)
 - Enforcement of exotic constraints
 - Novelty, aesthetics, manufacturability, creativity, ...
 - Inverse design
- **8.7.3 Anomaly Detection**
 - If generation is not possible, it must be an anomaly!
- **8.7.4 Conditional Generation**
 - Additional inputs to generative models (but still multiple outputs)
- **8.7.5 Surrogate Modeling**
 - Transformer architectures as surrogates
 - U-Net

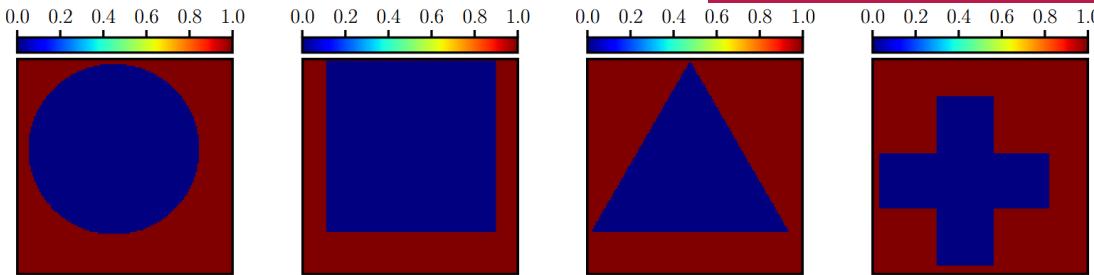


8.7.1 Data Generation

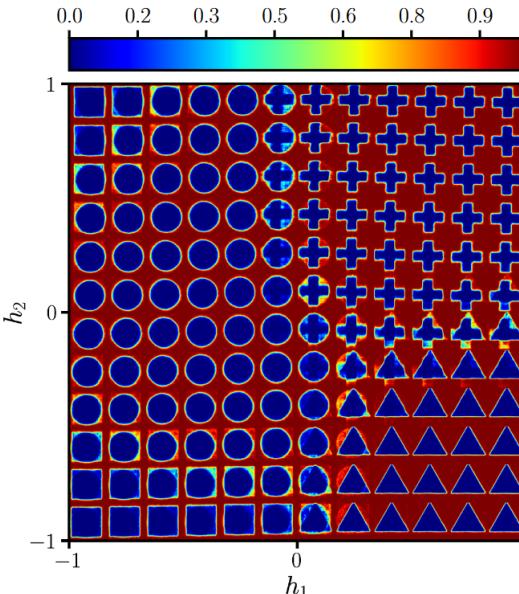
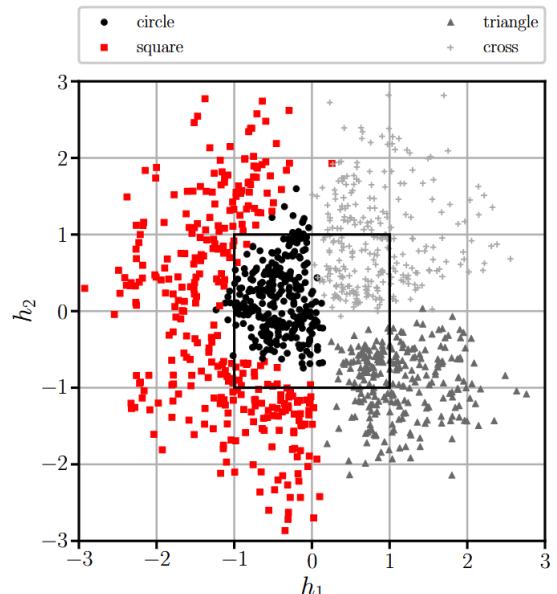
Shape generation with [variational autoencoders](#)

Training with

- Reconstruction loss
- Kullback-Leibler divergence



Latent space



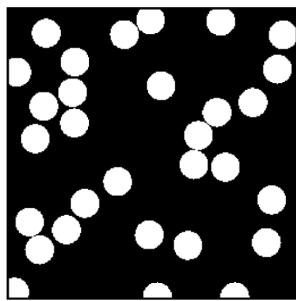
Interpolation of the
latent space

Exercises

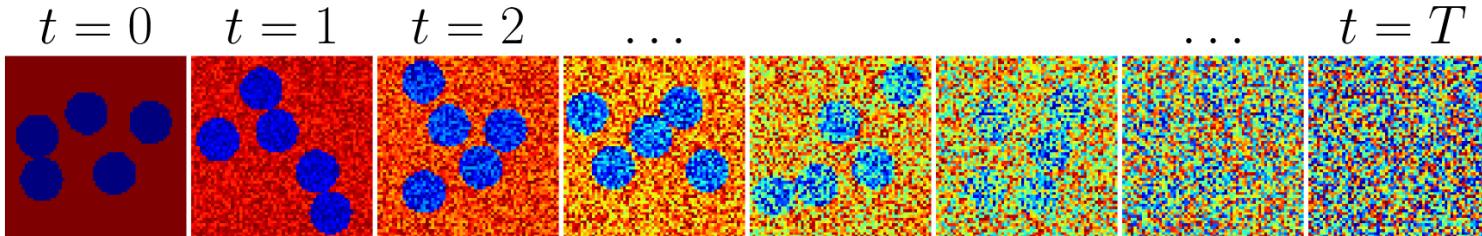
- E.28 Variational Autoencoder (C)
 - Apply a variational autoencoder to a simple data generation task. The task consists of generating basic shapes; circles, triangles, squares and crosses. Explore the latent space distribution and the effect of the Kullback-Leibler divergence.

8.7.1 Data Generation

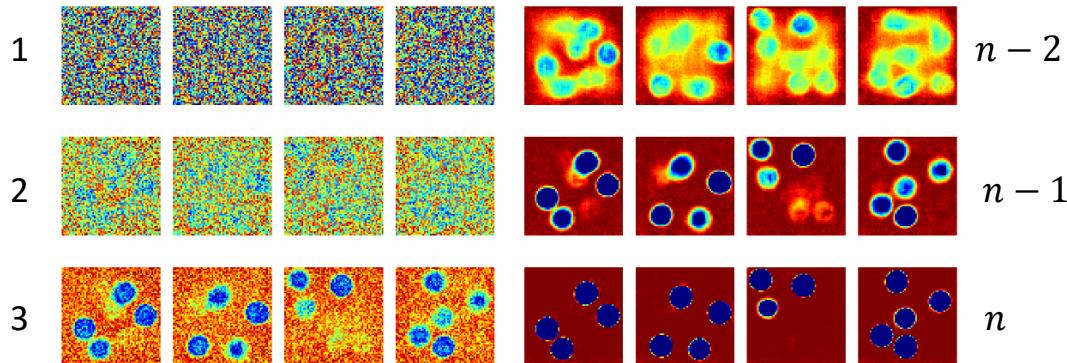
Fiber distribution generation with diffusion model



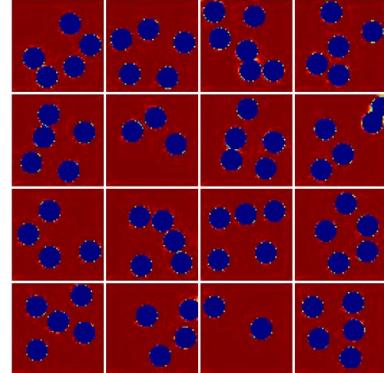
Forward diffusion process



Learned reverse denoising process



After 40 denoising steps

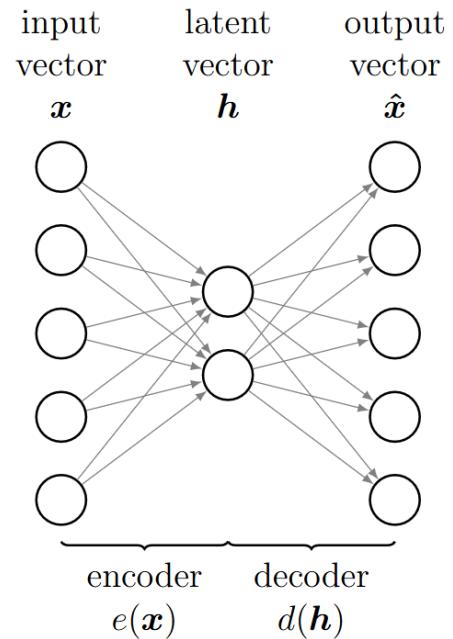
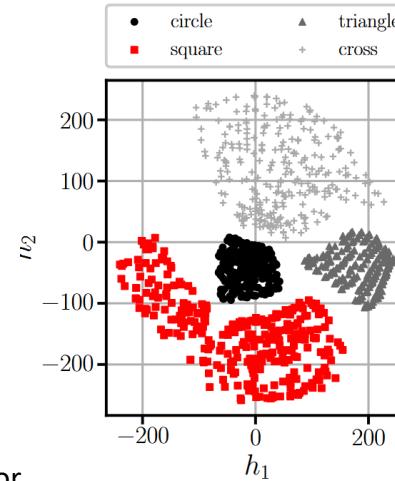
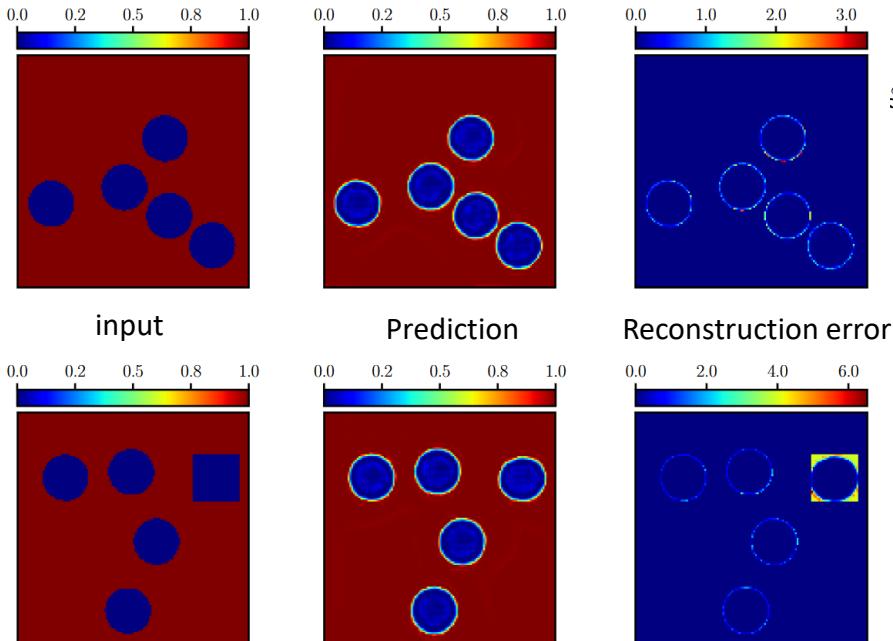


Exercises

- E.30 Diffusion models (C)
 - Apply a diffusion model to a simple data generation task. The task consists of generating simple fiber matrix microstructures.

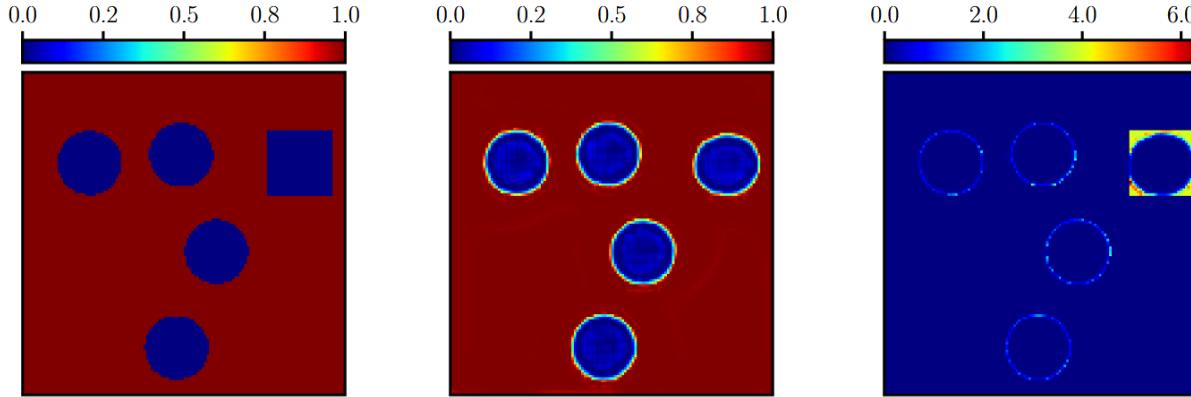
8.7.3.1 Autoencoders for Anomaly Detection

- Training of [autoencoder](#) on "normal data", i.e., without anomalies
- Detection of outliers
 - In latent space
 - By inability to reconstruct the input image

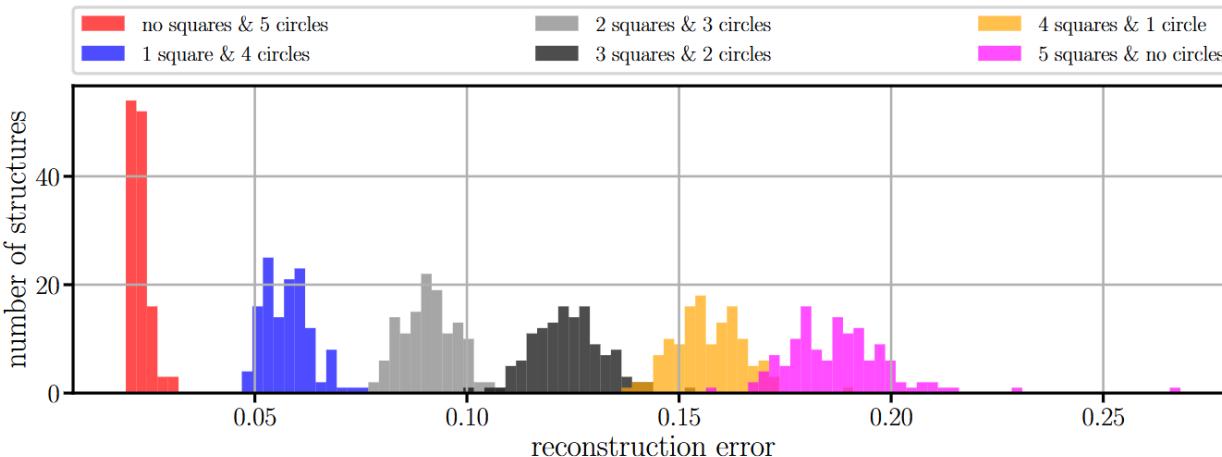


Deep Learning for Anomaly Detection: A Review, Pang et al. 2021

8.7.3.1 Autoencoders for Anomaly Detection



Statistical evaluation



Exercises

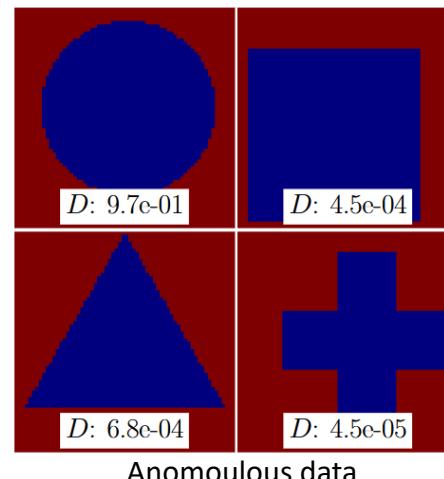
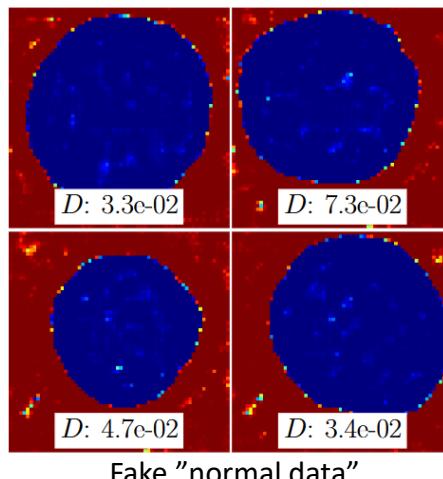
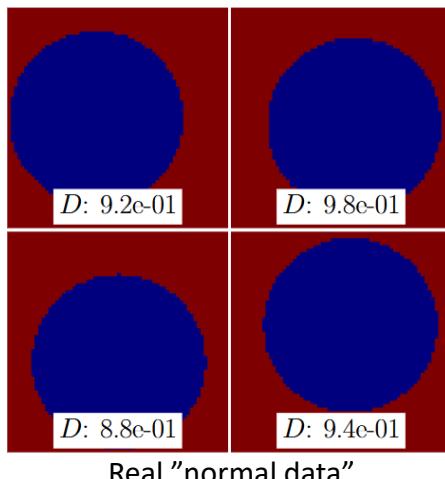
- E.27 Autoencoder (C)
 - Apply an autoencoder as an anomaly detector. First train the autoencoder on normal data (using a dataset consisting of randomly distributed circles). Next apply the autoencoder to anomalous data (using a dataset consisting of randomly distributed circles and squares). Compare the reconstruction errors with those obtained with normal data.

8.7.3.2 Generative Adversarial Networks for Anomaly

Detection

- Training of **generative adversarial network** on "normal data", i.e., without anomalies
- Detection of outliers
 - By inability to reconstruct the input image from noise (requires optimization with respect to input)
 - Through the discriminator
 - Discriminator cannot distinguish between real & fake "normal data"
 - Anomalies outside of "normal data" distribution are detectable

Diffusion models can be used similarly

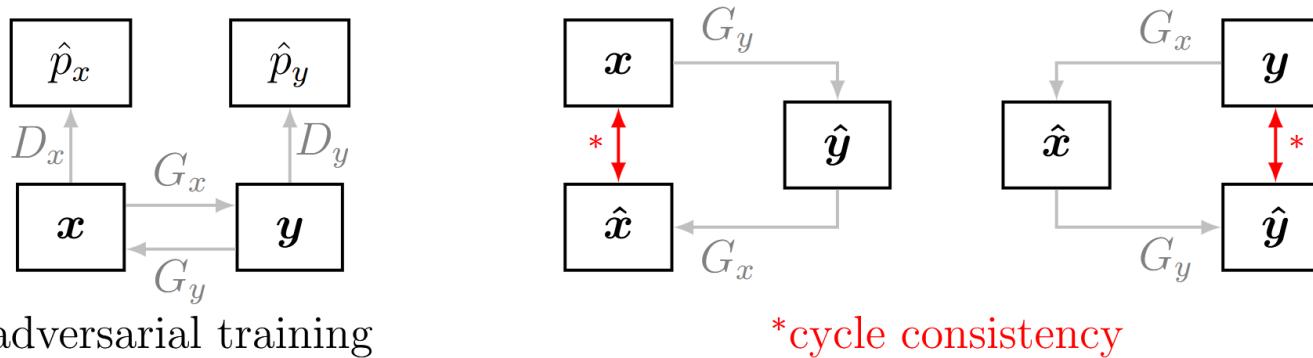


Exercises

- E.29 Generative Adversarial Network (C)
 - Apply a generative adversarial network as an anomaly detector. First train the generative adversarial network on normal data (using a dataset consisting of circles). Next apply the trained discriminator to anomalous data (using a dataset consisting of squares, triangles, and crosses). Compare the discriminator score to the scores obtained with normal data.

8.7.4 Conditional Generation

- Conditional adversarial networks
 - Include inputs & labeled outputs (supervised)
- InfoGANs
 - Include latent code as input & disentanglement loss term (unsupervised)
- Variational autoencoder generative adversarial networks
 - Autoencoder architecture for the GAN generator
- CycleGANs



- tempoGAN (for temporal data, e.g., video)
 - Temporal coherence is assessed by second discriminator

Contents

- [7 Material Modeling with Neural Networks](#)
- [8.1 Autoencoders](#)
- [8.2 Variational Autoencoder](#)
- [8.4 Autoencoders in Computer Vision: U-Net](#)
- [8.3 Generative Adversarial Networks](#)
- [8.5 Diffusion Models](#)
- [8.6 Transformers](#)
- [8.7 Applications in Computational Mechanics](#)
 - [8.7.1 Data Generation](#)
 - [8.7.3 Anomaly Detection](#)
 - [8.7.4 Conditional Generation](#)
- [9 Inverse Problems & Deep Learning](#)

8 Generative Artificial Intelligence

Leon Herrmann

Stefan Kollmannsberger

Chair of Data Engineering in Construction

Bauhaus-Universität Weimar

Paris, February 2025

Deep Learning in Computational Mechanics – an introductory course,

Herrmann et al. 2025



website



book

