ScaDS.AI
DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS AND
ARTIFICIAL INTELLIGENCE

IOM AI School

TOPIC: Deep Learning – Theory and Practice
SPEAKER: Matthias Täschner

Using materials from Laura Zigutyte, Michaela Unger (EKFZ, TU Dresden), Robert Haase (ScaDS.AI, Leipzig University)

These slides can be reused under the terms of the CC-BY4.0 license.

TECHNISCHE UNIVERSITÄT DRESDEN

UNIVERSITÄT LEIPZIG
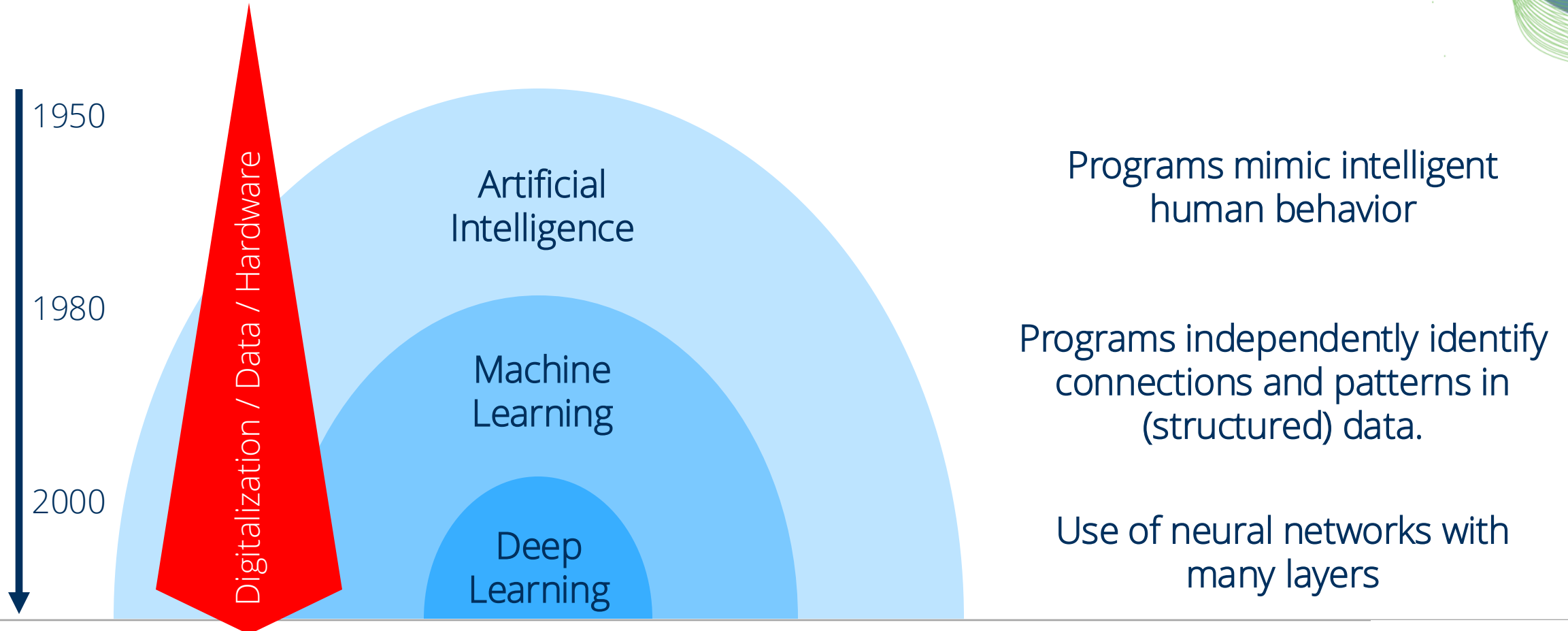
AGENDA

- Areas of Artificial Intelligence (AI)

- Neural Networks and Deep Learning

- Neural Network Architectures

- Practise: Neural Network for Regression
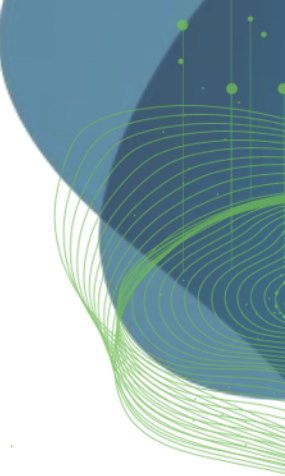
Areas of Artificial Intelligence

Historical phases of AI

1950

1980

2000

Digitalization / Data / Hardware

Artificial Intelligence

Machine Learning

Deep Learning

Programs mimic intelligent human behavior

Programs independently identify connections and patterns in (structured) data.

Use of neural networks with many layers

Areas of Artificial Intelligence

Model Family

Areas, paradigms, model families (not exhaustive)

Artificial Intelligence

Machine Learning

- Linear Models
- Decision Trees
- Kernel-based
- Clustering
- Ensembles
- ...

Neural Networks

- Feed-Forward
- ...

Deep Learning

- Autoencoder
- RNN
- CNN
- Transformer
- ...

Paradigms
- Supervised
- Unsupervised
- Reinforcement
- Semi-Supervised
- Self-Supervised

Problems
- Regression
- Classification
- Clustering
- Dimension reduction
- Anomaly detection
- Sequence-to-sequence
- Recommendations
- Generative modeling
- Time series prediction
- ...

Areas of Artificial Intelligence
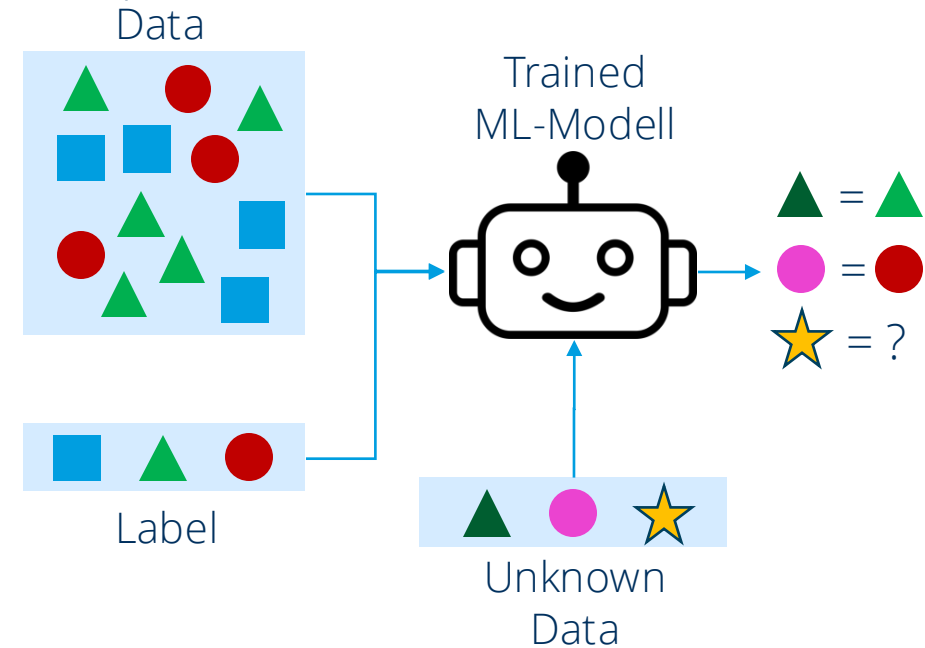## Paradigms - Supervised Learning

Procedure

- ML models are trained using pre-labeled data
- Input data and the desired target values (ground truth) are provided for training
- Prediction of target values on new, previously unknown input data of the same format

Application examples

- Classification, Regression
- Anomaly Detection
- Generative modeling

Algorithm / model type examples

- Linear Regression
- Decision Trees & Random Forest (DT & RF)
- Support-Vector-Machines (SVM)
- Neural Networks (NN)

Areas of Artificial Intelligence
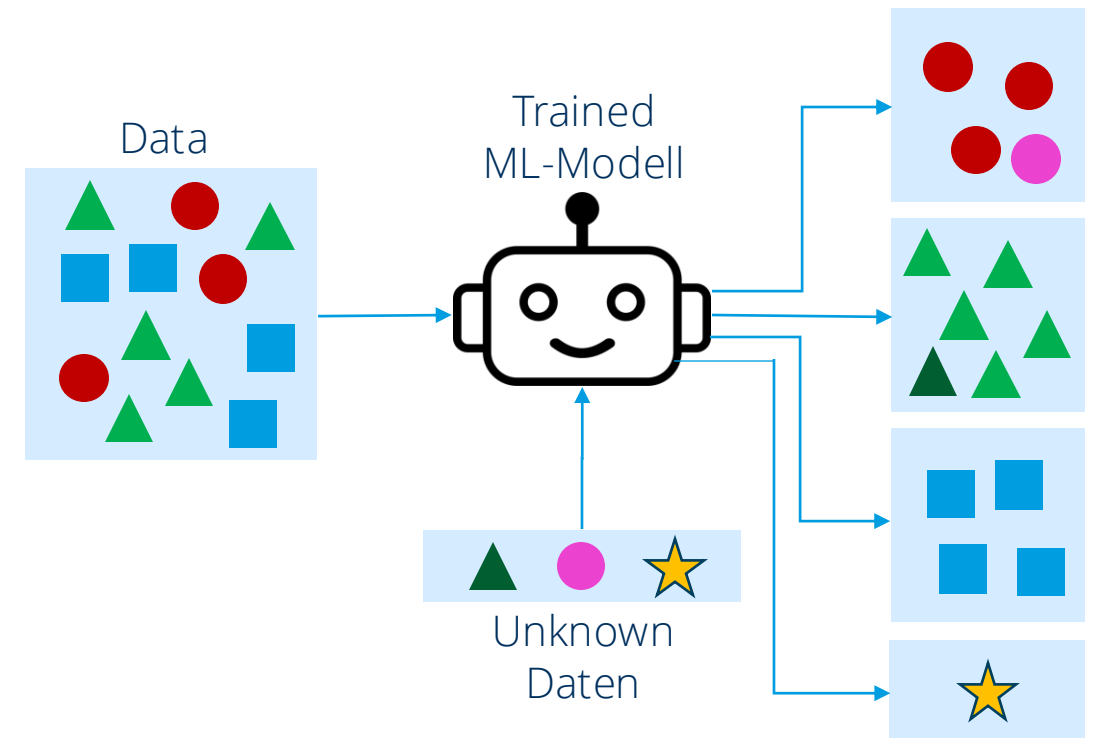## Paradigms - Unsupervised Learning

Procedure

- ML models are trained with unlabeled data

- Models independently recognize patterns, structures, and correlations

Application examples

- Clustering

- Dimensionality Reduction

- Anomaly Detection

Algorithm / model type examples

- K-Means Clustering

- Principal Component Analysis (PCA)

- Autoencoder

Data

Trained
ML-Modell

Unknown
Daten

IOM AI School
Topic: Deep Learning – Theory and Practice
Speaker: Matthias Täschner, Leipzig University, ScaDS.AI

ScaDS.AI DRESDEN LEIPZIG

TECHNISCHE UNIVERSITÄT DRESDEN

UNIVERSITÄT LEIPZIG

Areas of Artificial Intelligence
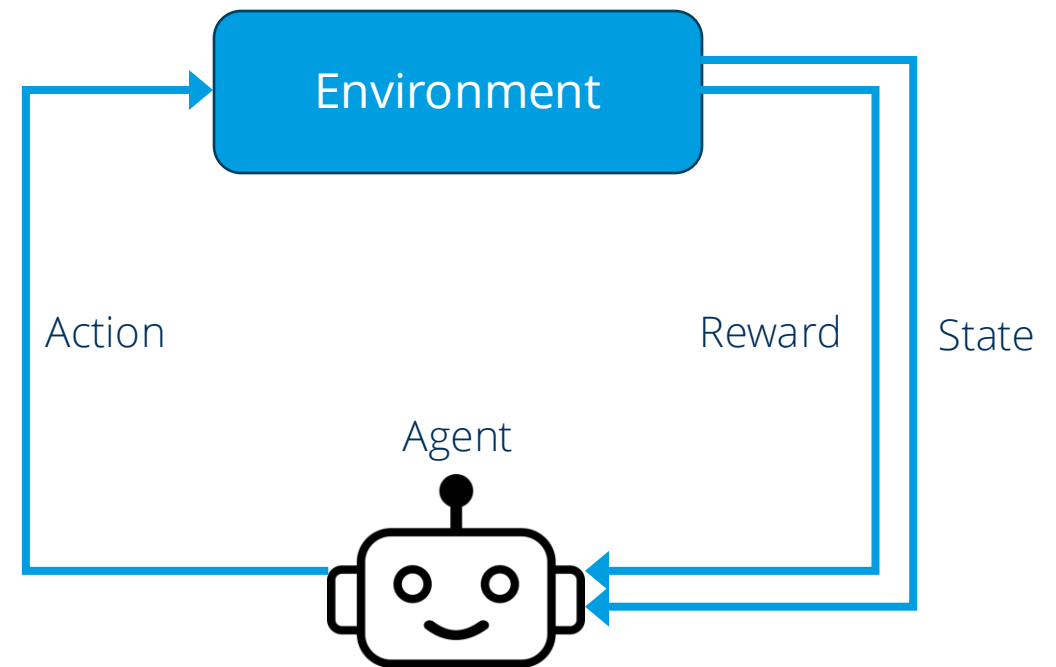## Paradigms - Reinforcement Learning

Procedure

- An agent is trained to maximize a specific reward in an environment through trial and error
- Rules define the possible actions the agent can take
- Rewards (or punishments) influence the agent's behavior

Application examples

- Game agents, e.g., in chess or Go
- Automation systems, robotics
- Simulations

Algorithm / model type examples

- Q-Learning
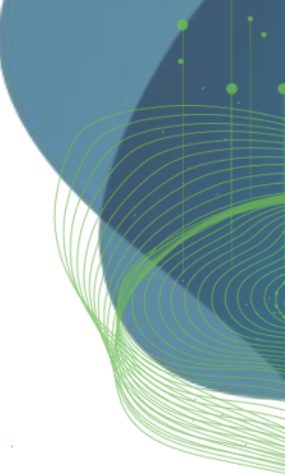- Markov Decision Processes (MDP)
- Monte Carlo Methods

Environment

Action   Reward   State

Agent

IOM AI School
Topic: Deep Learning – Theory and Practice
Speaker: Matthias Täschner, Leipzig University, ScaDS.AI

Areas of Artificial Intelligence
Paradigms - Mixed Forms

Semi-Supervised

- A small dataset with labeled samples
- A larger dataset with unlabeled samples from the same distribution
- Training on labeled data → model assigns "pseudo-labels" to unlabeled data
- Include data with the most reliable "pseudo-labels" in further training (human-in-the-loop)

Self-Supervised

- Data is not labeled; required labels are generated by the model itself
- The model generates its own "practice questions" and answers from the raw data
- In doing so, it learns useful patterns from the data
- Example: Cover words in a sentence and then guess them – no label necessary

IOM AI School
Topic: Deep Learning – Theory and Practice
Speaker: Matthias Täschner, Leipzig University, ScaDS.AI

Areas of Artificial Intelligence
Linear models vs. non-linear models

- Linear model ≠ plotting a straight line / curve
- Linearity refers to model parameters

A model is linear if its prediction is a linear combination of the features.
*(e.g., weighted sum of input data)*

- Linear models
  - Easy to interpret, based on well-understood principles, fast and efficient training
  - Can only model simple, linear relationships in the data
  - Examples: Linear or logistic regression, ARIMA, linear SVM, etc.
- Non-linear models
  - Allows to model complex, non-linear relationships in the data
  - More difficult to interpret, complex training with more parameters, often more data required
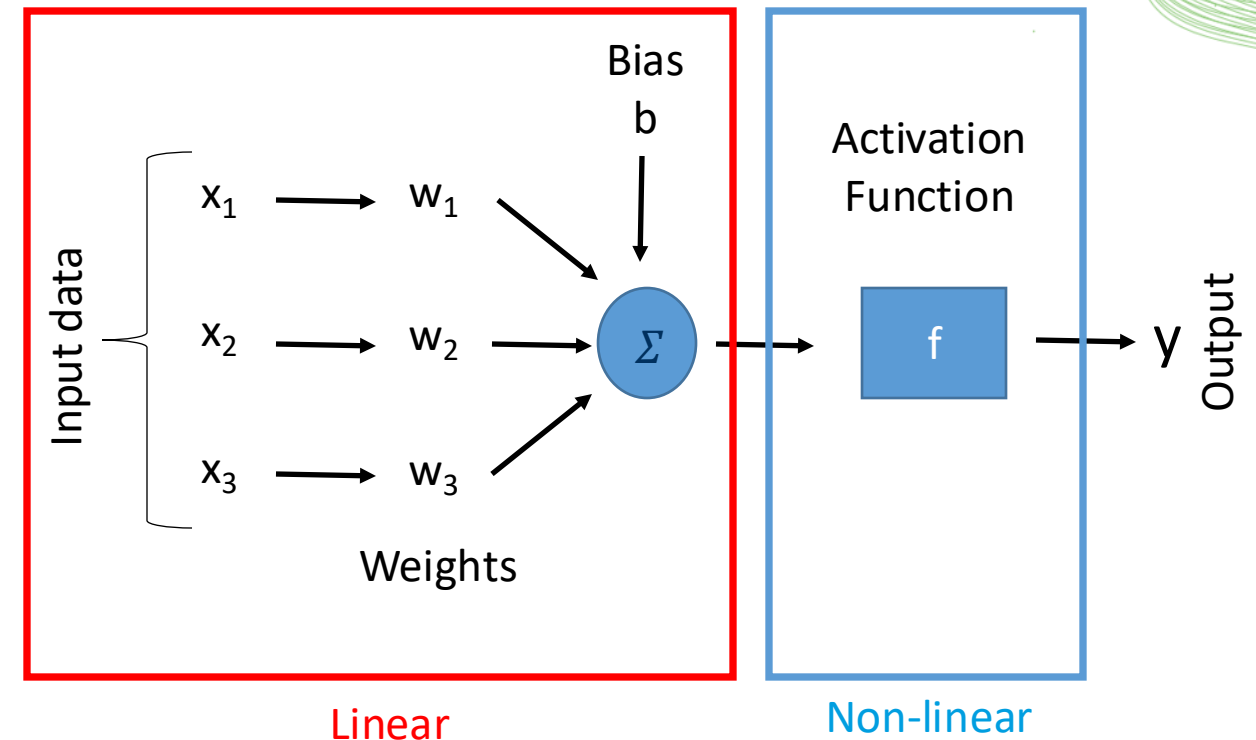 - Examples: Decision tree, kernel SVM, k-NN, neural networks, etc.
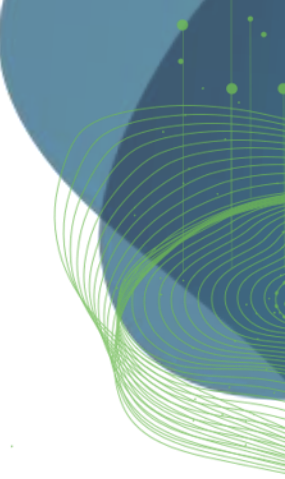
Neural Networks and Deep Learning
## Basic Architecture

Neurons from a biological perspective
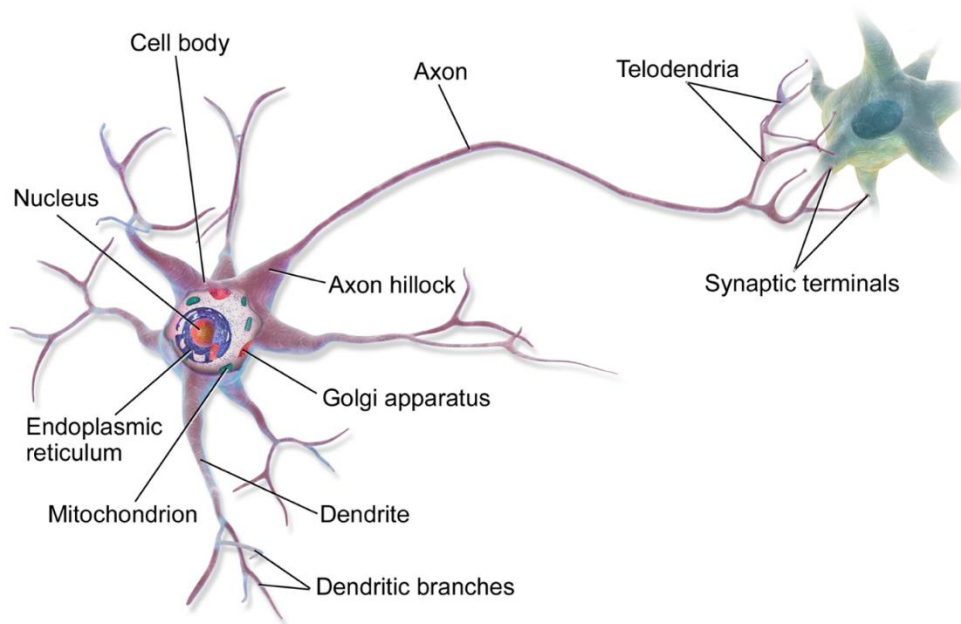
Source: https://commons.wikimedia.org/wiki/File:Blausen_0657_MultipolarNeuron.png
License: CC-BY 3.0, BruceBlaus

Neuron in Data Science

Linear · Non-linear

$$y = f(w_1 x_1 + w_2 x_2 + w_3 x_3 + b)$$

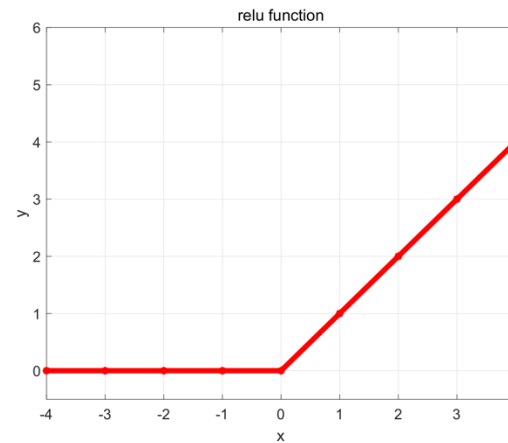# Neural Networks and Deep Learning
Basic Architecture
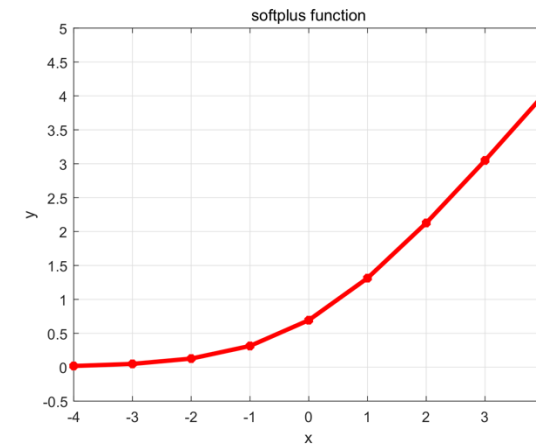
Activation-Function

- Non-linear transformation of the linear combination of input features
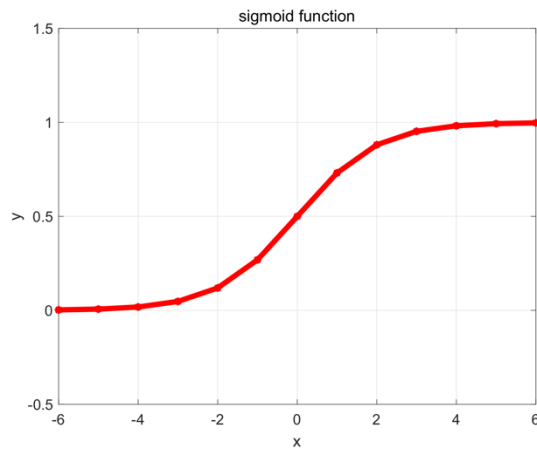
(a) The curve of sigmoid function

(c) The curve of ReLu function

(d) The curve of softplus function

Source: Wang, Y., Li, Y., Song, Y., & Rong, X. (2020). The Influence of the Activation Function in a Convolution Neural Network Model of Facial Expression Recognition. Applied Sciences, 10(5), 1897. https://doi.org/10.3390/app10051897, OpenAccess

- More examples and explanations: https://en.wikipedia.org/wiki/Activation_function

Neural Networks and Deep Learning
Basic Architecture

Multi-Layer Perceptron

- Several layers of neurons: Input → Hidden → Output
- Neurons in one layer are fully connected to those in the next layer (Fully-Connected, Dense)
- Information flows through the network from input to output only (Feed Forward)

Input layer   Hidden layer   Output layer

Feed forward
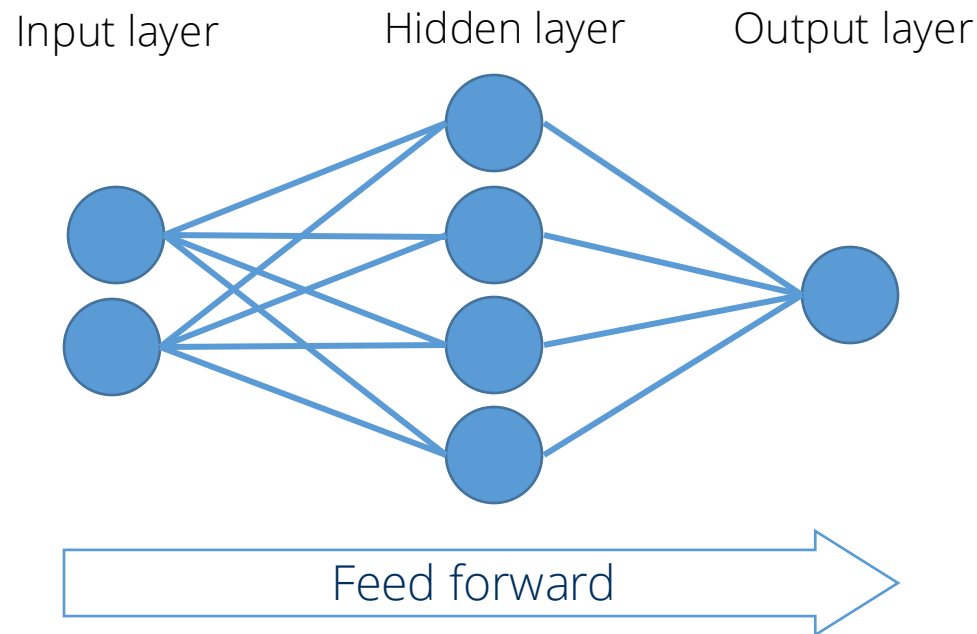
„Deep Learning"
Neural networks with **many** hidden layers
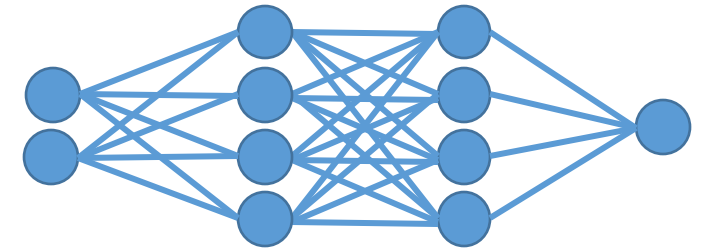
Neural Networks and Deep Learning
Data Processing

Processing and conversion of input data

- Create features that can be used and learned by ML models / neural networks
- Goal: Turn input data into numerical representation so a neural network can process them

Is the data representative?

Handling of missing data

Handling of outliers

Feature selection

Data scaling / normalization

Embeddings

Train-test-split
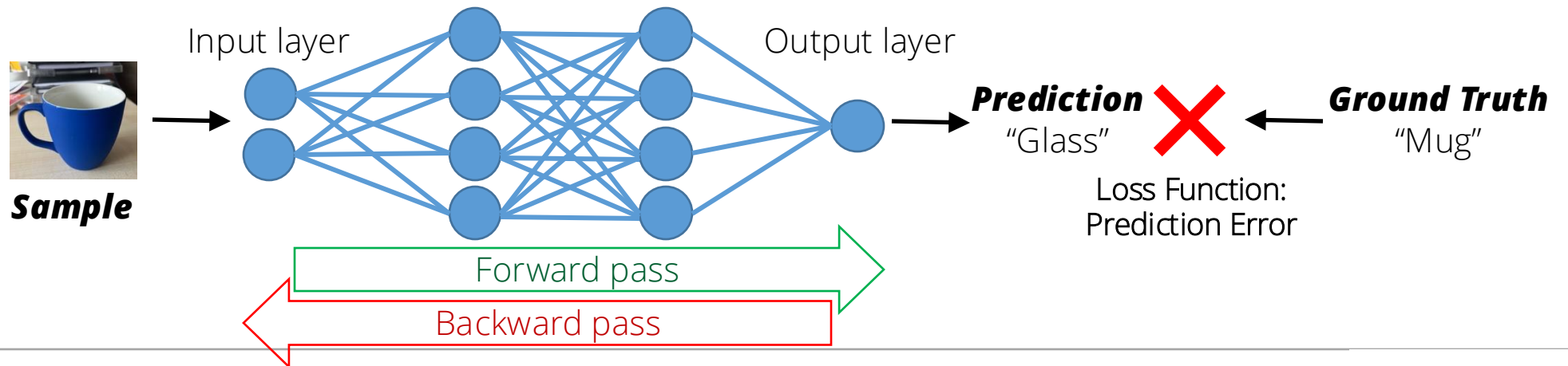
…

Doc1

Doc2

Doc3

The dog chased the cat across the yard.

Neural Networks and Deep Learning
Training Process

Learning through "Back Propagation"

- Step 1: Initialize weights of all neurons with small random values

- Step 2: Send data point (**sample**) through the network for prediction (**forward pass**)

- Step 3: Compare prediction with known label for the sample (supervised learning with **ground truth**), calculate error between prediction and ground truth (via **loss function**)

- Step 4: Starting with the output layer (**backward pass**), adjust the weights in the neurons with the goal to minimize the calculated error (e.g., via **gradient descent**)

- Repeat steps 2-4 for each sample in training data (one **epoch**), running through this for several epochs


Input layer

Output layer

Sample

Prediction
"Glass"

Ground Truth
"Mug"

Loss Function:
Prediction Error

Forward pass

Backward pass

Neural Networks and Deep Learning
Training Process

Weight Adjustment – Optimization via Gradient Descent

- Gradient descent updates weights to minimize an objective function ➔ loss function ➔ error
- Follow the steepest descent
- Learning rate controls the step size
  ➔ too large = overshoot, too small = slow training
- Training may get stuck in local minima instead of reaching best global solution
- Different optimizer algorithms improve training
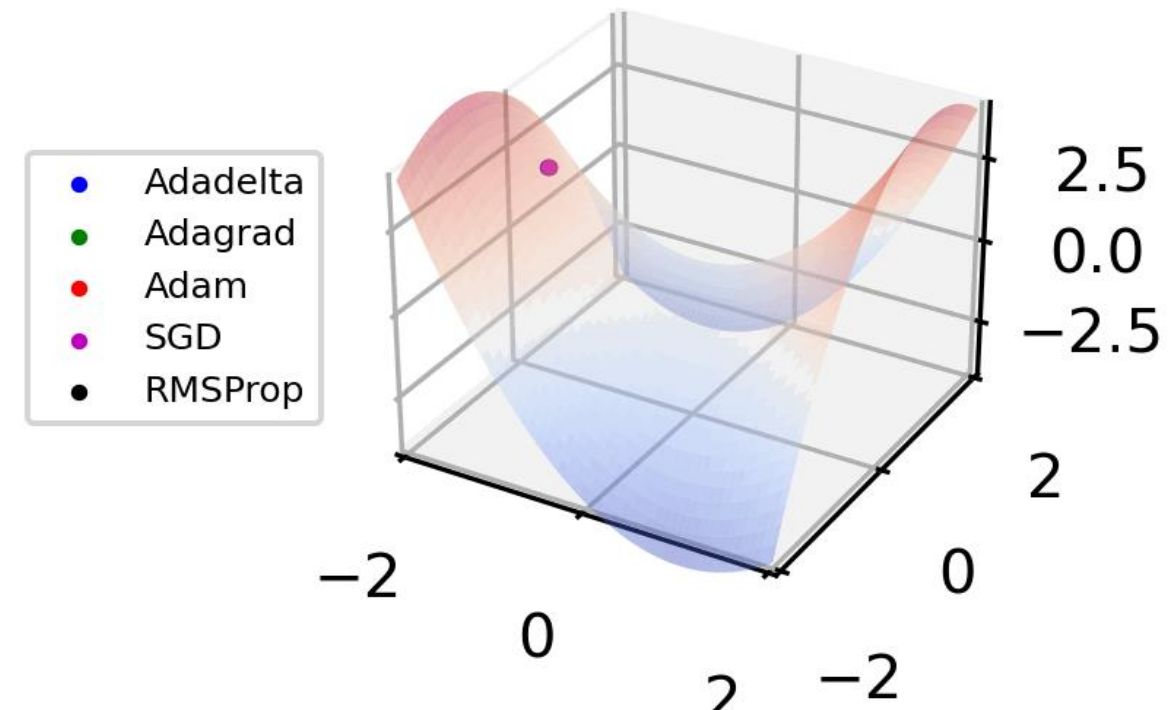
Legend:
- Adadelta
- Adagrad
- Adam
- SGD
- RMSProp

Neural Networks and Deep Learning
## Training Process

Architecture and Training Parameters (Hyperparameters)

- Model parameters = learned by the neural network during training (weights, biases)
- Model architecture choices
  - Layer types and arrangement
  - Number of neurons per layer
  - Type of activation function
  - ...
- Training parameter
  - Number of samples / batches per epoch
  - Number of epochs
  - Learning rate
  - ...
- Hyperparameter optimization → find the best configuration for the architecture and the training
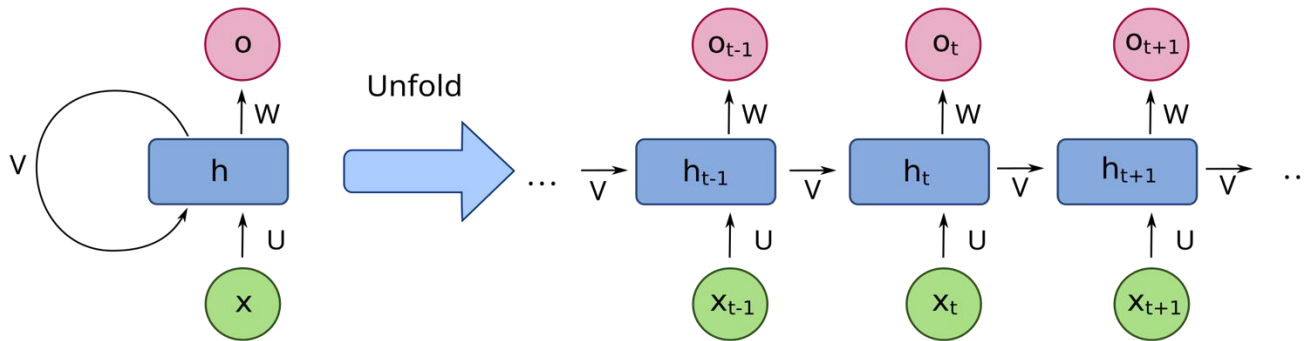
Neural Network Architectures
Autoencoder

- Neural network for data compression (encoder) and reconstruction (decoder)
- Most important information (properties of features) get embedded in low-dimensional **latent space**
- Training aims to minimize the error during reconstruction
- Applications, e.g., dimensionality reduction or anomaly detection (large error = anomaly)
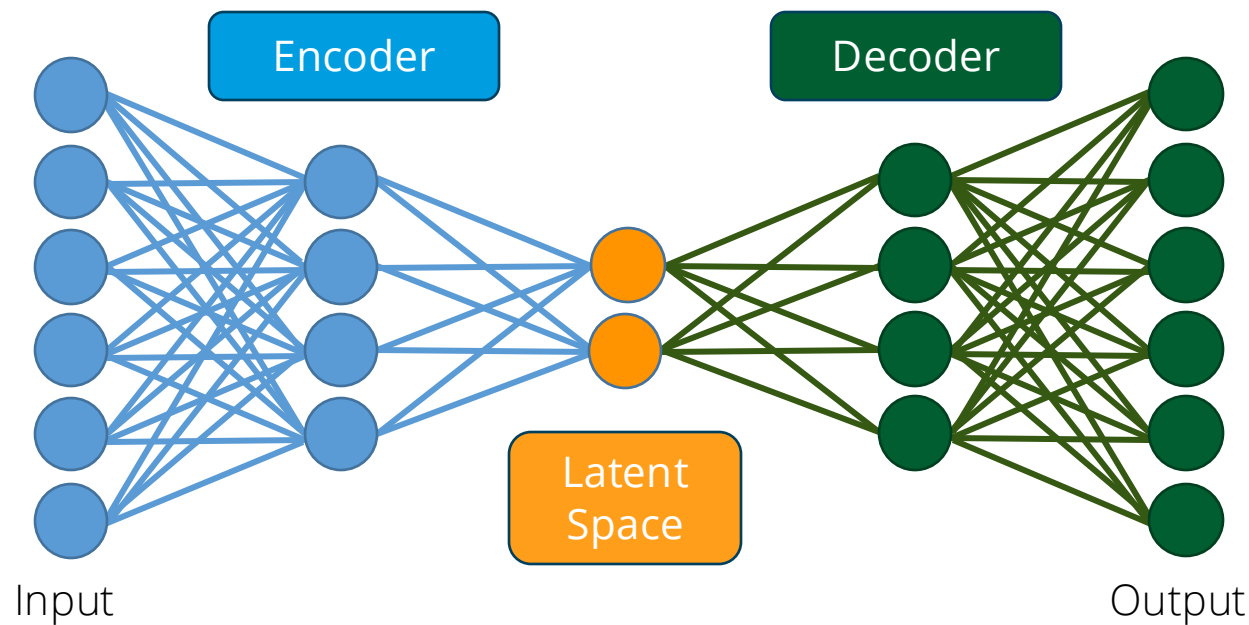
Neural Network Architectures
Recurrent Neural Network (RNN / LSTM / GRU)

- Neural network for modeling sequences
- Implements "memory" with the help of "feedback loops"
- Information no longer flows only from input to output (feed forward), but is also fed back into the neurons of previous hidden layers
- Variants of this enable, among other things, "long short-term memory" (LSTM)

x – Input
h – Hidden (Memory)
o – Output
U, V, W – according Weights

Unfold…"roll up" over time:
At time t, the previous state t-1 is incorporated via V
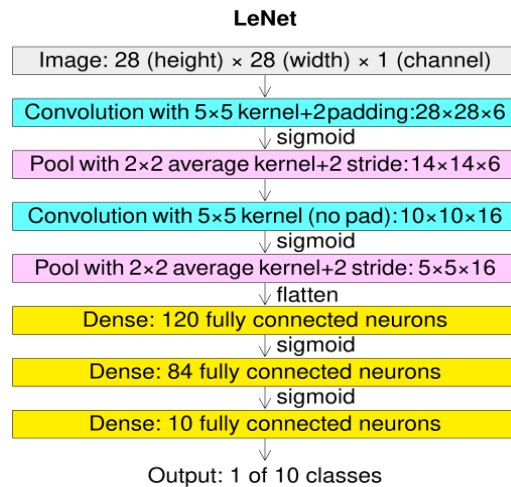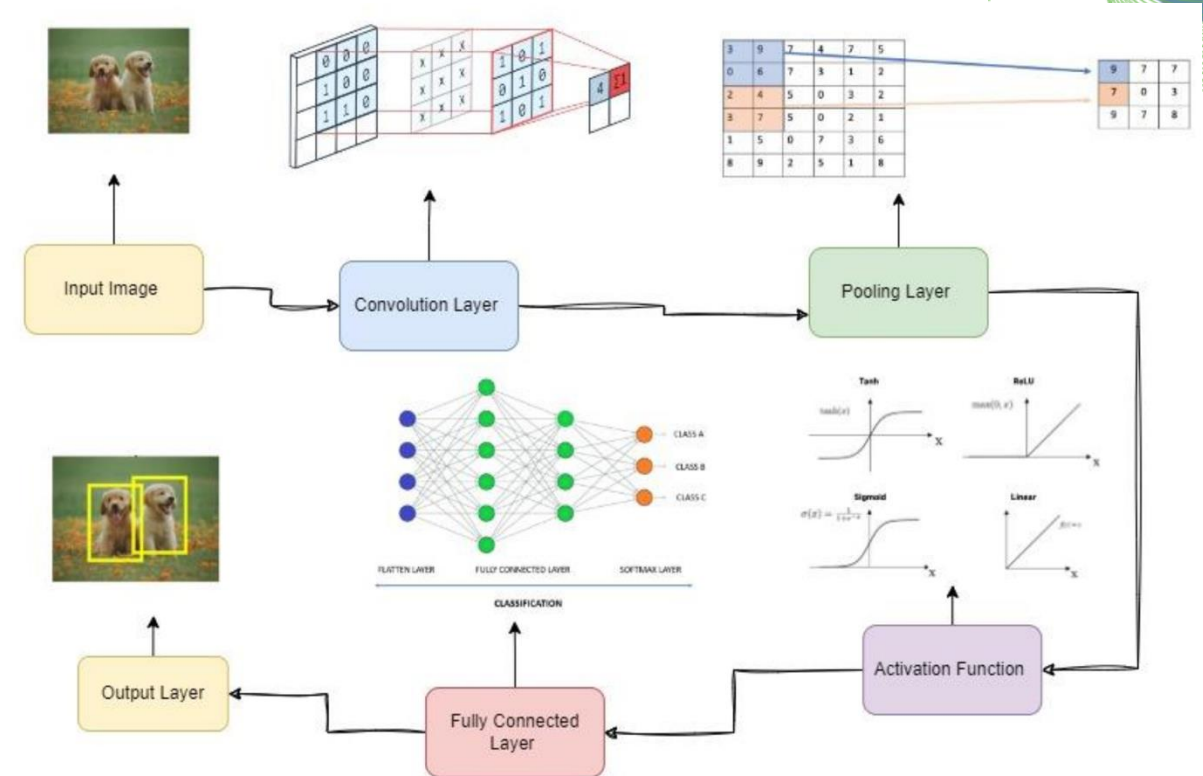At time t+1, the previous state t is incorporated via V
…

Neural Network Architectures
Convolutional Neural Network (CNN)

- Neural network with different layer types for extracting local to global features from input data
  - Convolutional Layer (Convolution)
  - Pooling Layer (Aggregation)
  - Dropout Layer (Regularization)
  - Fully Connected Layer (Classification)
- Application mostly in image processing (object recognition, classification)

LeNet

Image: 28 (height) × 28 (width) × 1 (channel)
Convolution with 5×5 kernel+2padding:28×28×6

↓ sigmoid

Pool with 2×2 average kernel+2 stride:14×14×6
Convolution with 5×5 kernel (no pad):10×10×16

↓ sigmoid

Pool with 2×2 average kernel+2 stride: 5×5×16

↓ flatten

Dense: 120 fully connected neurons

↓ sigmoid

Dense: 84 fully connected neurons

↓ sigmoid

Dense: 10 fully connected neurons

Output: 1 of 10 classes

Source: Cmglee, https://commons.wikimedia.org/w/index.php?curid=104937230, CC BY-SA 4.0

Source: Taye, M. M. (2023). Theoretical Understanding of Convolutional Neural Network: Concepts, Architectures, Applications, Future Directions. *Computation*, *11*(3), 52. https://doi.org/10.3390/computation11030052, OpenAccess
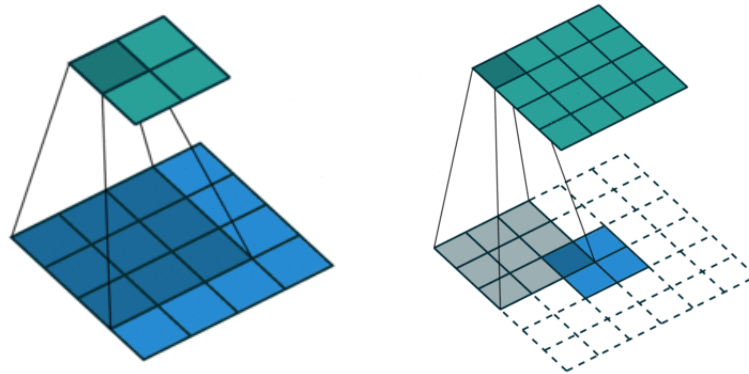
Neural Network Architectures
Convolutional Neural Network (CNN)

- Convolutional Layer – <u>not</u> fully connected
- Kernel moves over input and extracts features by applying element-wise multiplications and sums
  → kernels are small weight matrices that are learned during training

0	2	0
2	1	2
0	2	0

3x3 Kernel

Quellen:
Vincent Dumoulin and Francesco Visin (2018). A guide to convolution arithmetic for deep learning. *arXiv*. https://doi.org/10.48550/arXiv.1603.07285
Vincent Dumoulin, Francesco Visin, https://github.com/vdumoulin/conv_arithmetic, MIT

- Pooling Layer aggregates input (e.g., max, avg, …), introduces robustness to variations in the data
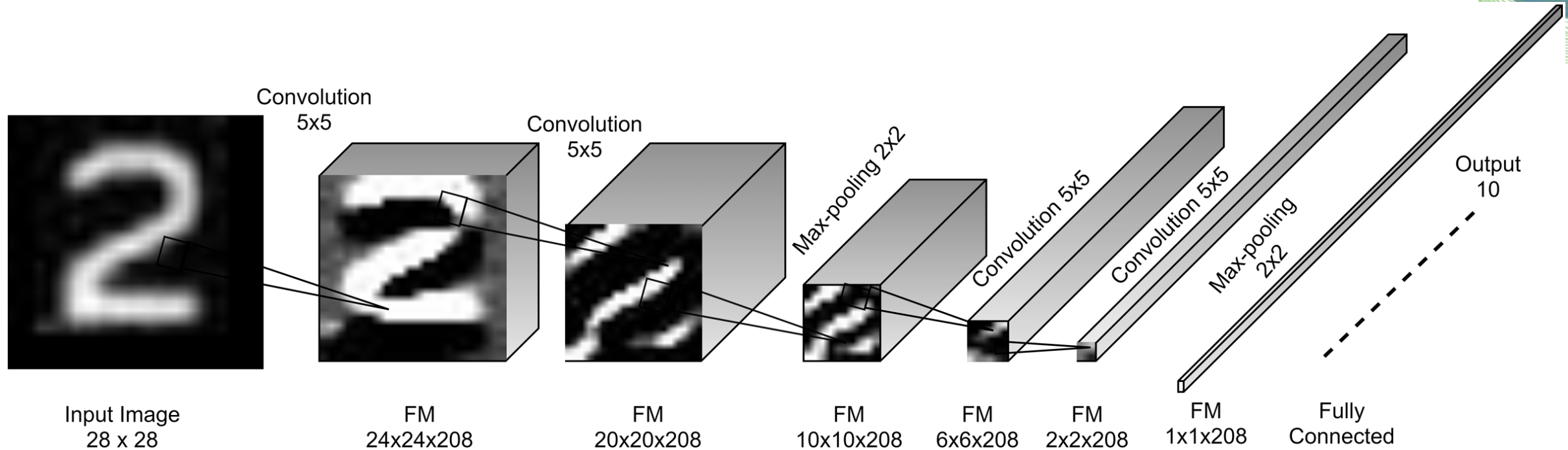- Dropout randomly removes connections during training to prevent overfitting

Max pooling

IOM AI School
Topic: Deep Learning – Theory and Practice
Speaker: Matthias Täschner, Leipzig University, ScaDS.AI

Neural Network Architectures
Convolutional Neural Network (CNN)

- Convolutional Neural Network – the full image

Convolution 5x5

Convolution 5x5
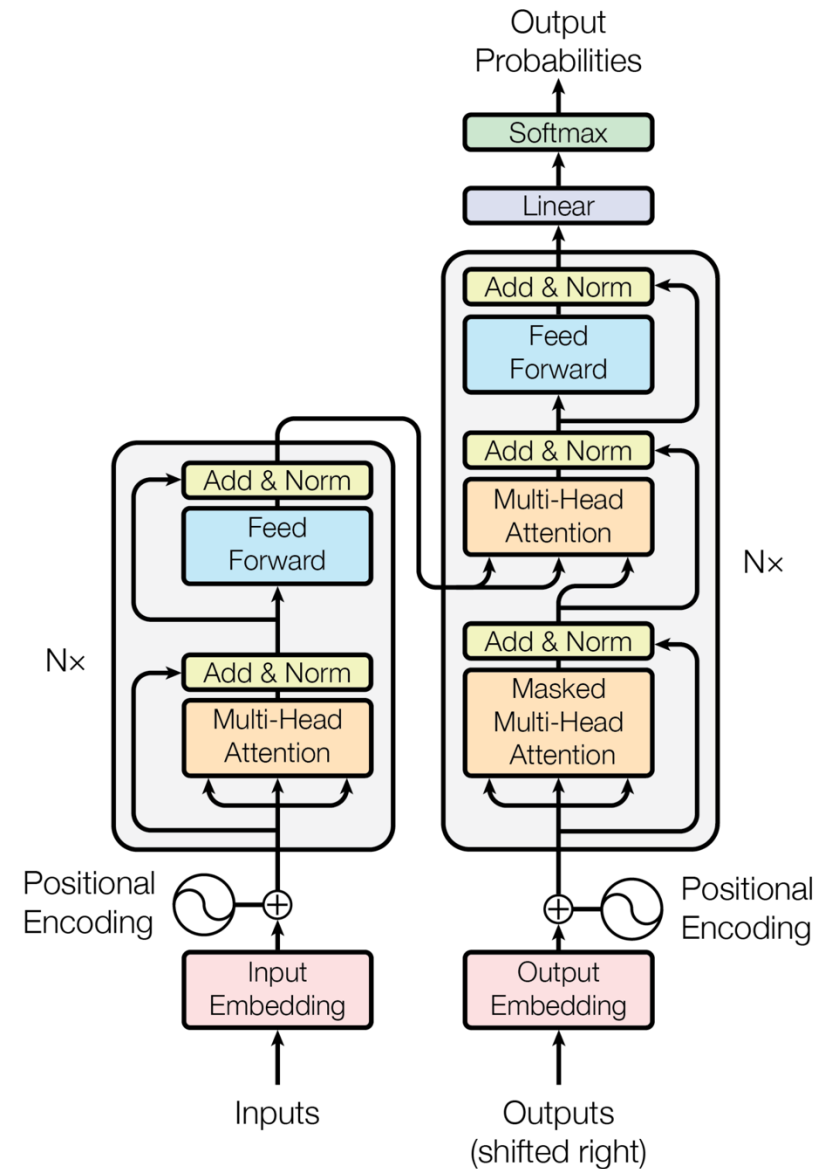
Max-pooling 2x2

Convolution 5x5

Convolution 5x5

Max-pooling 2x2

Output 10

Input Image 28 x 28

FM 24x24x208

FM 20x20x208

FM 10x10x208

FM 6x6x208

FM 2x2x208

FM 1x1x208

Fully Connected

Neural Network Architectures
Transformer

- Complex, multi-component, multi-layer deep learning based model for generative AI, sequence modelling...
- Consists of various components, grouped into encoders and decoders
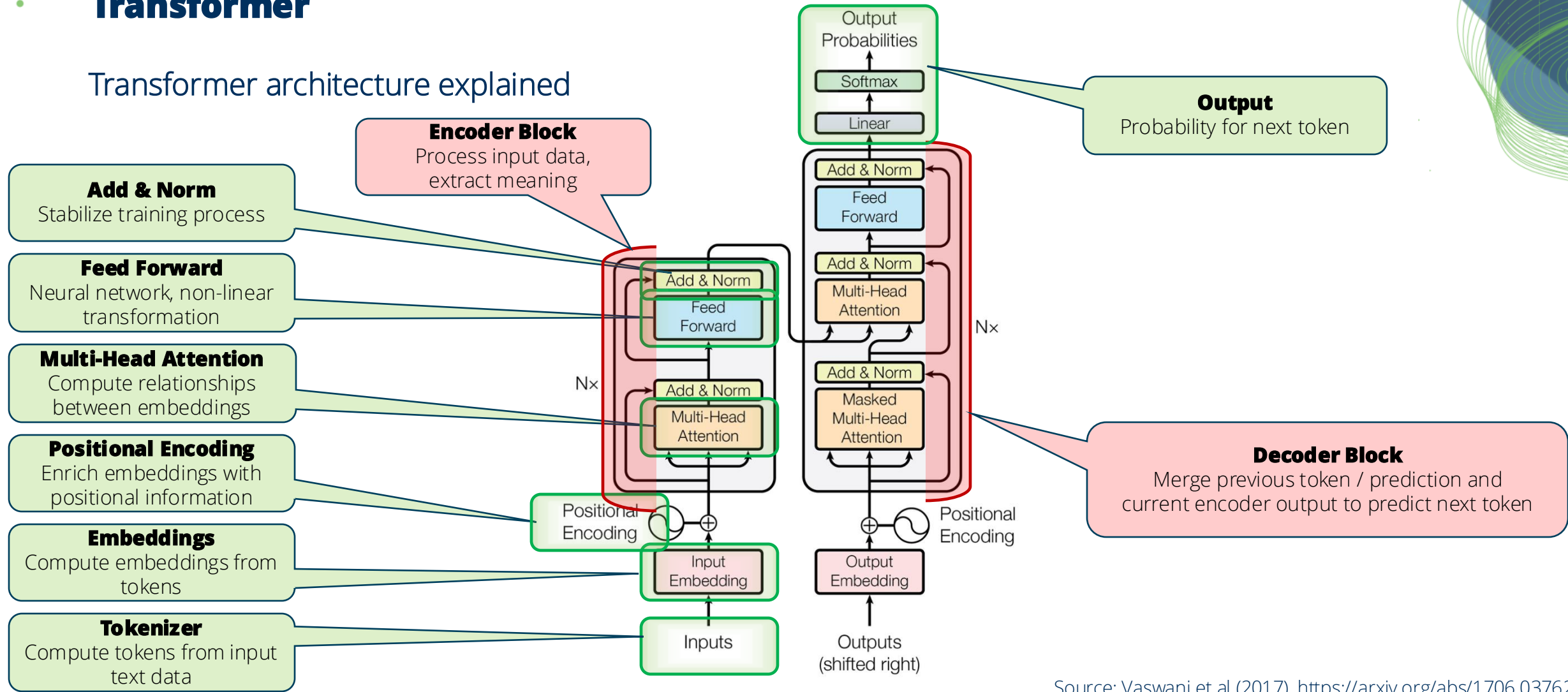

Quelle: Vaswani et al (2017), https://arxiv.org/abs/1706.03762

Neural Network Architectures
Transformer

Transformer architecture explained

Encoder Block
Process input data, extract meaning

Add & Norm
Stabilize training process

Feed Forward
Neural network, non-linear transformation

Multi-Head Attention
Compute relationships between embeddings

Positional Encoding
Enrich embeddings with positional information

Embeddings
Compute embeddings from tokens

Tokenizer
Compute tokens from input text data

Output
Probability for next token

Decoder Block
Merge previous token / prediction and current encoder output to predict next token

Source: Vaswani et al (2017), https://arxiv.org/abs/1706.03762

Practical exercise

- Build and train a neural network for a regression problem
- Covering model training, hyperparameter optimization, monitoring the training process, model evaluation
- Leveraging the Python libraries
  - pytorch – open-source deep learning framework
  - ray – open-source framework for model training and optimization