

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](https://creativecommons.org/licenses/by/4.0/) license.

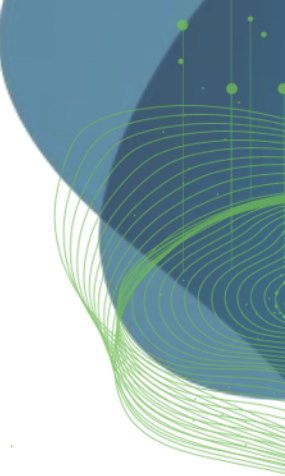


SACHSEN Diese Maßnahme wird gefördert durch die Bundesregierung aufgrund eines Beschlusses des Deutschen Bundestages. Diese Maßnahme wird mitfinanziert durch Steuermittel auf der Grundlage des von den Abgeordneten des Sächsischen Landtags beschlossenen Haushaltes.



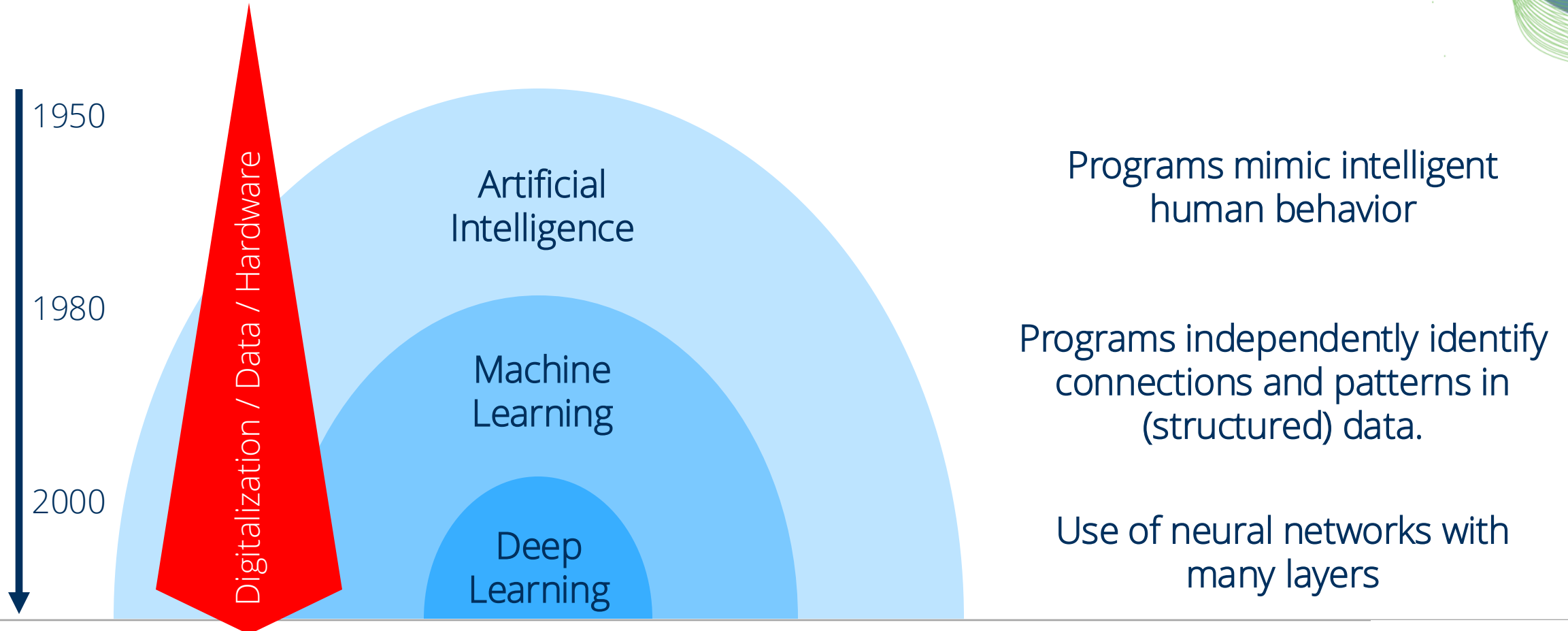
AGENDA

- Areas of Artificial Intelligence (AI)
- Neural Networks and Deep Learning
- Neural Network Architectures
- Practice: Neural Network for Regression



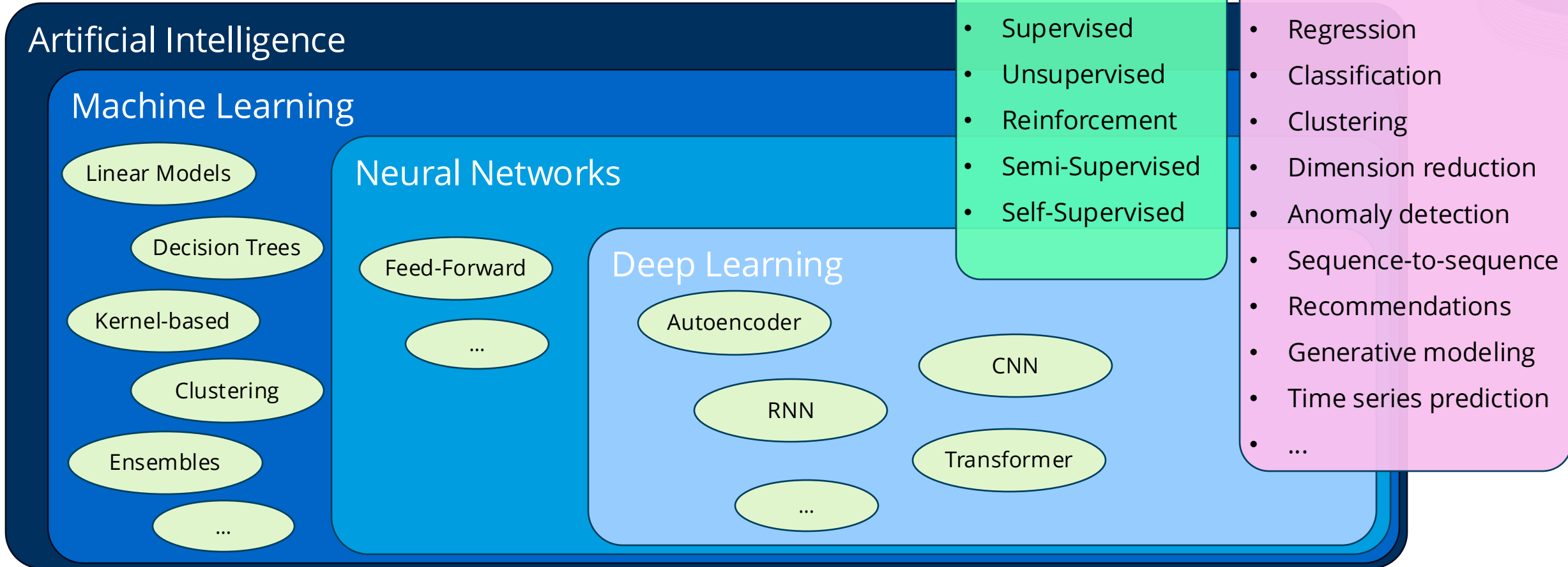
Areas of Artificial Intelligence

Historical phases of AI



Areas of Artificial Intelligence

Areas, paradigms, model families (not exhaustive)



Areas of Artificial Intelligence

Paradigms - Supervised Learning

Procedure

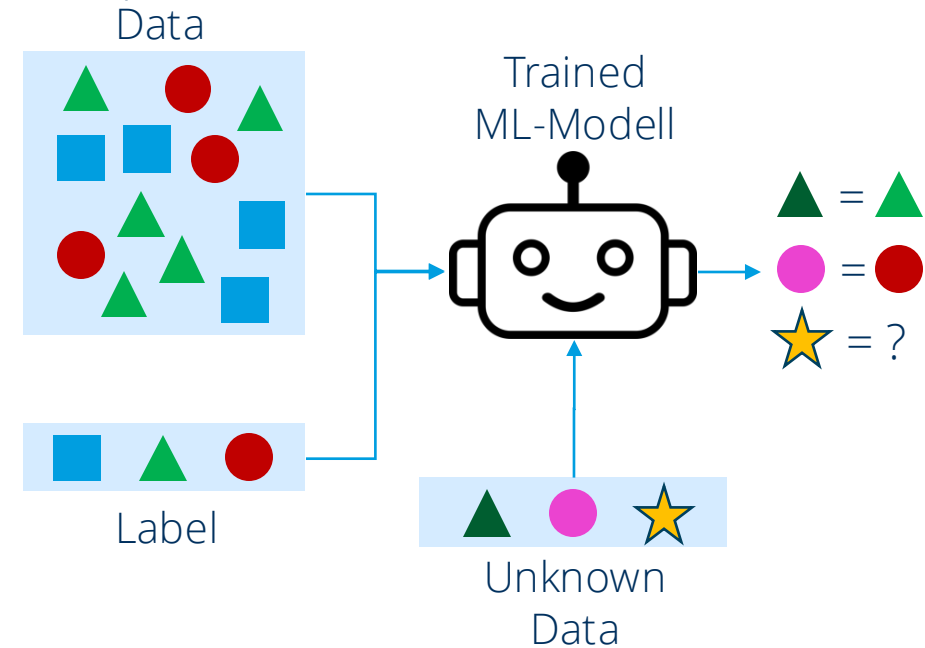
- ML models are trained using pre-labeled data
- Input data and the desired target values (ground truth) are provided for training
- Prediction of target values on new, previously unknown input data of the same format

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

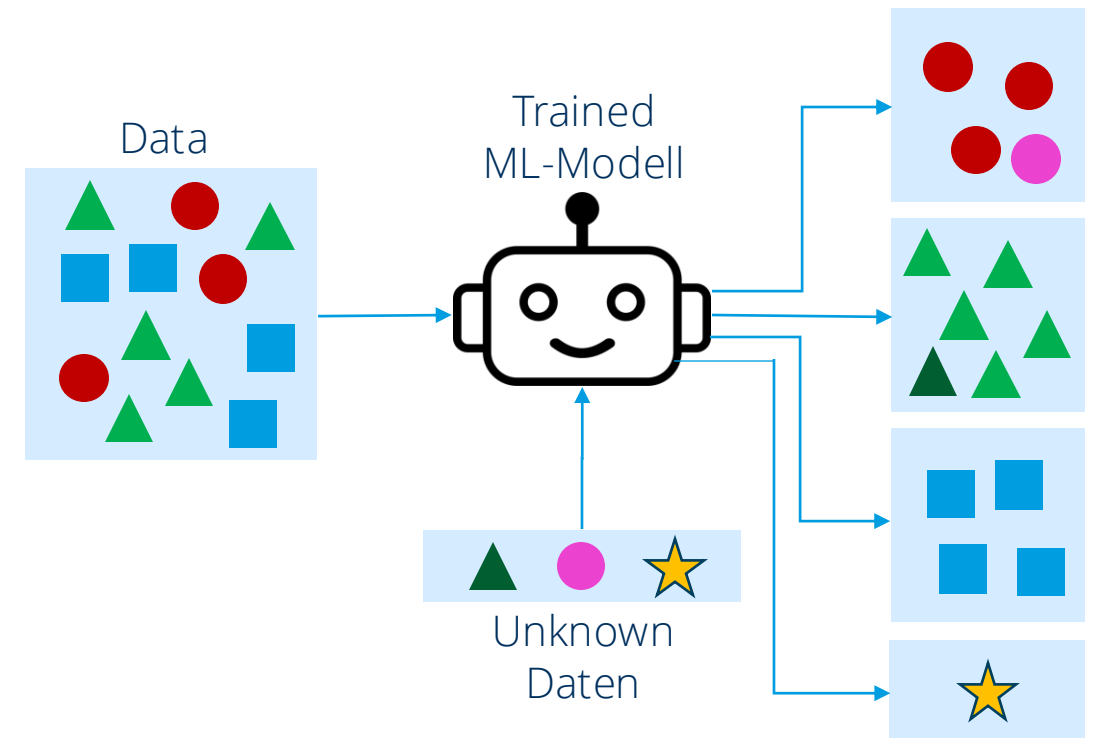
- ML models are trained with unlabeled data
- Models independently recognize patterns, structures, and correlations

Application examples

- Clustering
- Dimensionality Reduction
- Anomaly Detection

Algorithm / model type examples

- K-Means Clustering
- Principal Component Analysis (PCA)
- Autoencoder



Areas of Artificial Intelligence Paradigms - Reinforcement Learning

Procedure

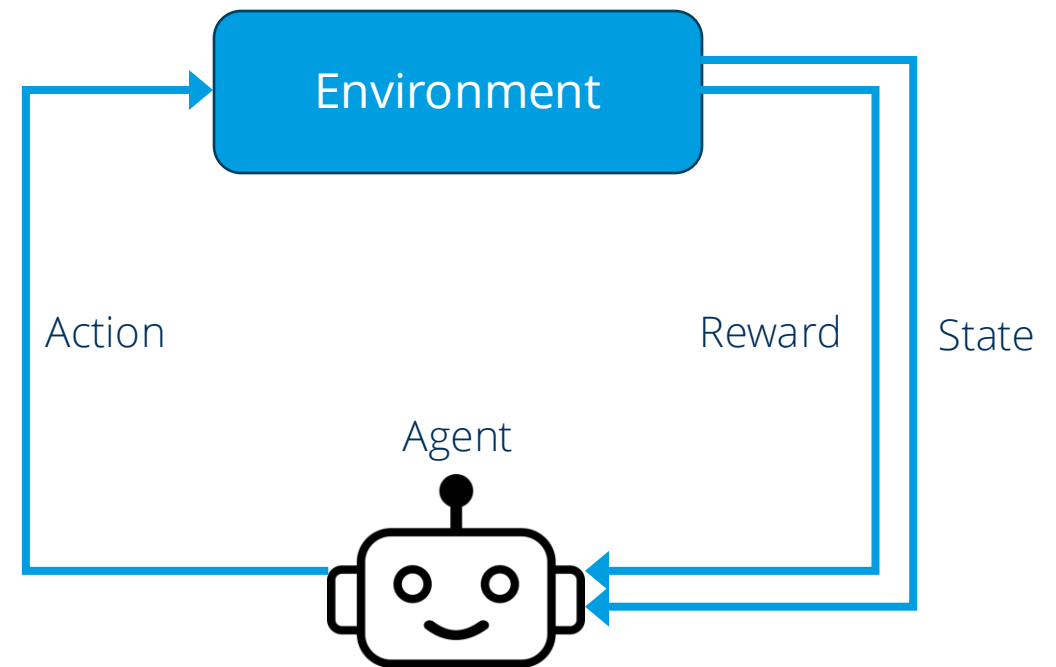
- An agent is trained to maximize a specific reward in an environment through trial and error
- Rules define the possible actions the agent can take
- Rewards (or punishments) influence the agent's behavior

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



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

Areas of Artificial Intelligence

Linear models vs. non-linear models

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

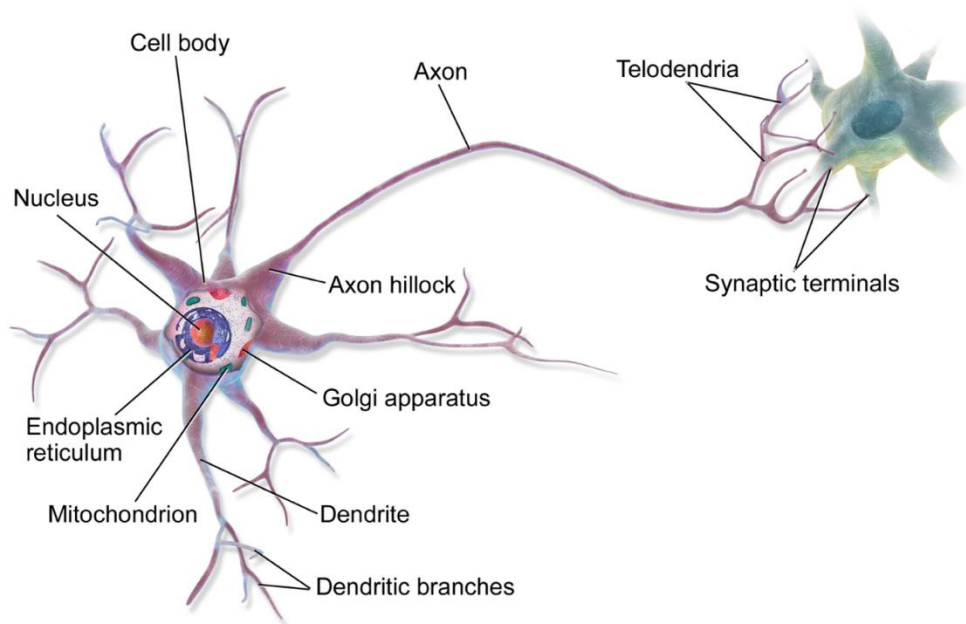
*A model is linear if its prediction is a linear combination of the features.
(e.g., weighted sum of input data)*

- 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.

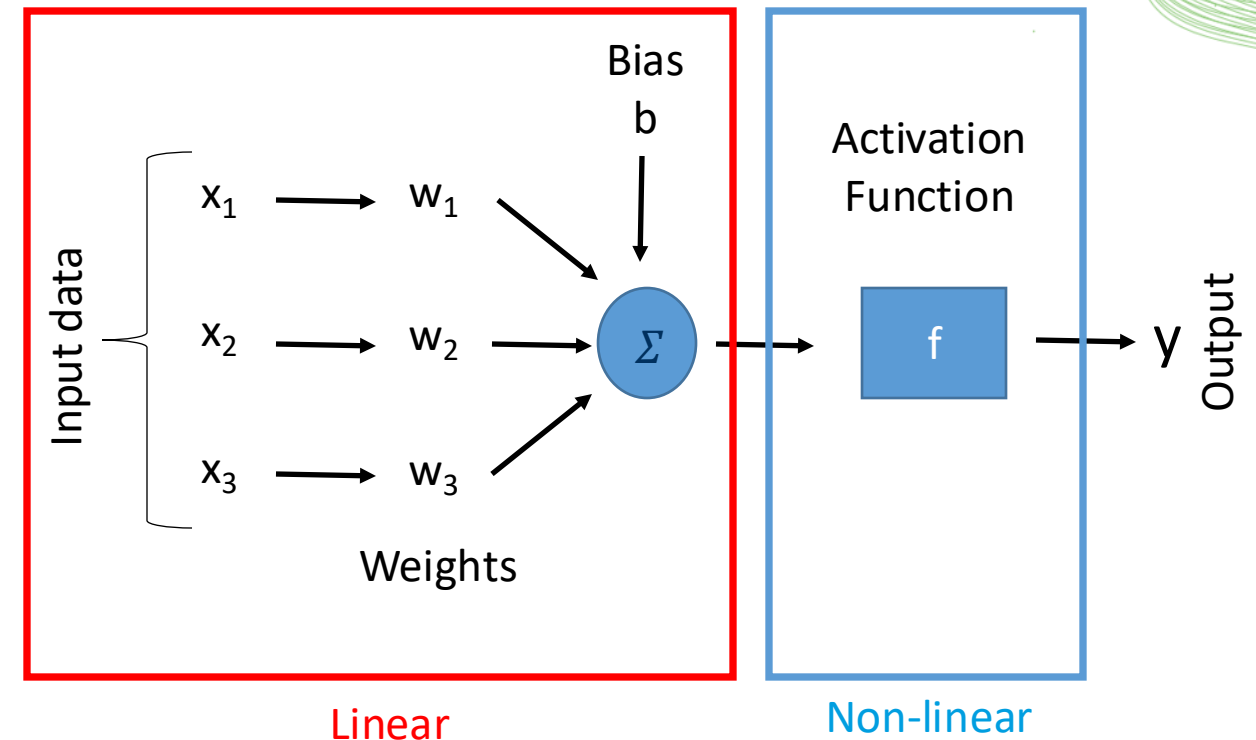
Neural Networks and Deep Learning

Basic Architecture

Neurons from a biological perspective



Neuron in Data Science



$$y = f(w_1x_1 + w_2x_2 + w_3x_3 + b)$$

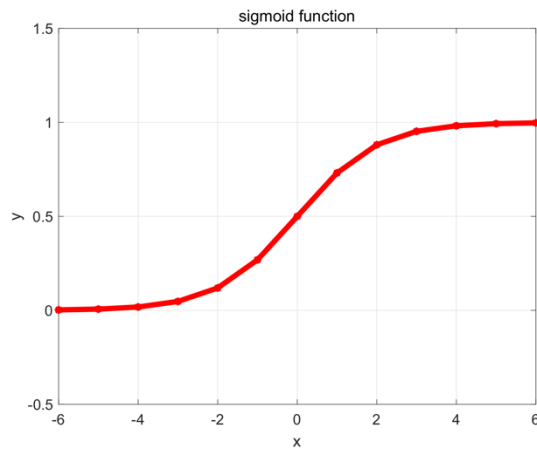
Source: https://commons.wikimedia.org/wiki/File:Blausen_0657_MultipolarNeuron.png
License: [CC-BY 3.0, BruceBlas](#)

Neural Networks and Deep Learning

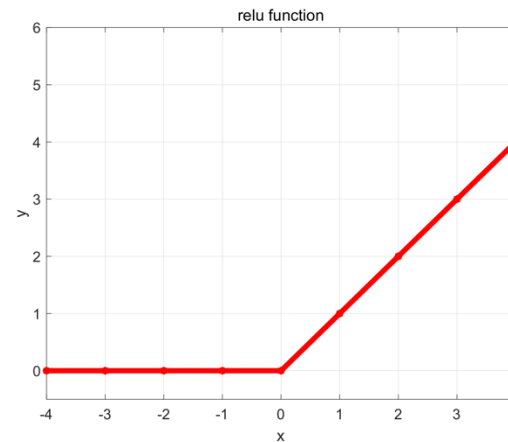
Basic Architecture

Activation-Function

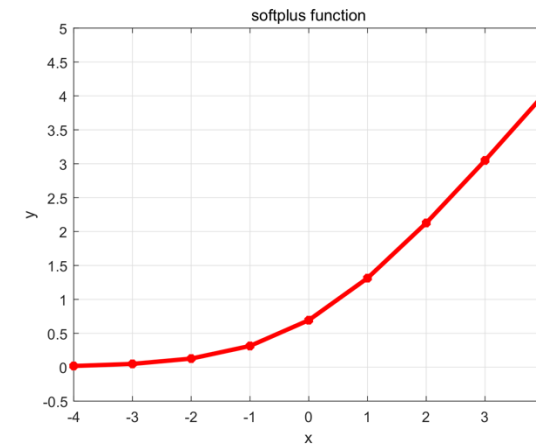
- Non-linear transformation of the linear combination of input features



(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

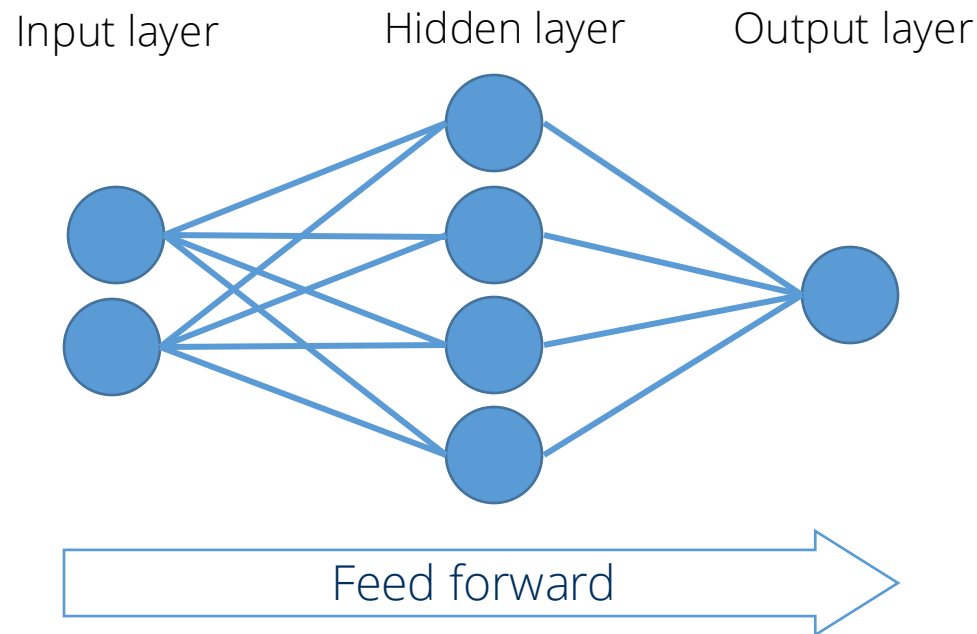
- More examples and explanations: https://en.wikipedia.org/wiki/Activation_function

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)



„Deep Learning“
Neural networks with
many hidden layers

Neural Networks and Deep Learning

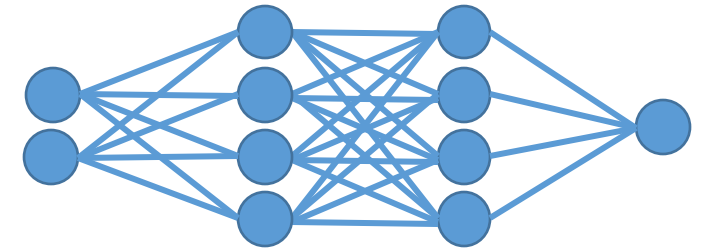
Data Processing

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
...

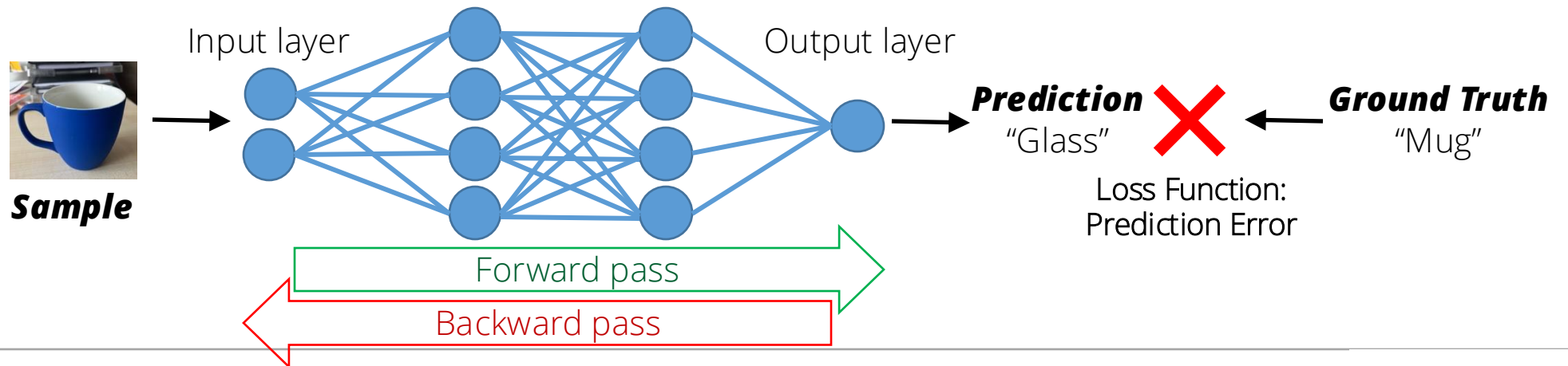


Neural Networks and Deep Learning

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

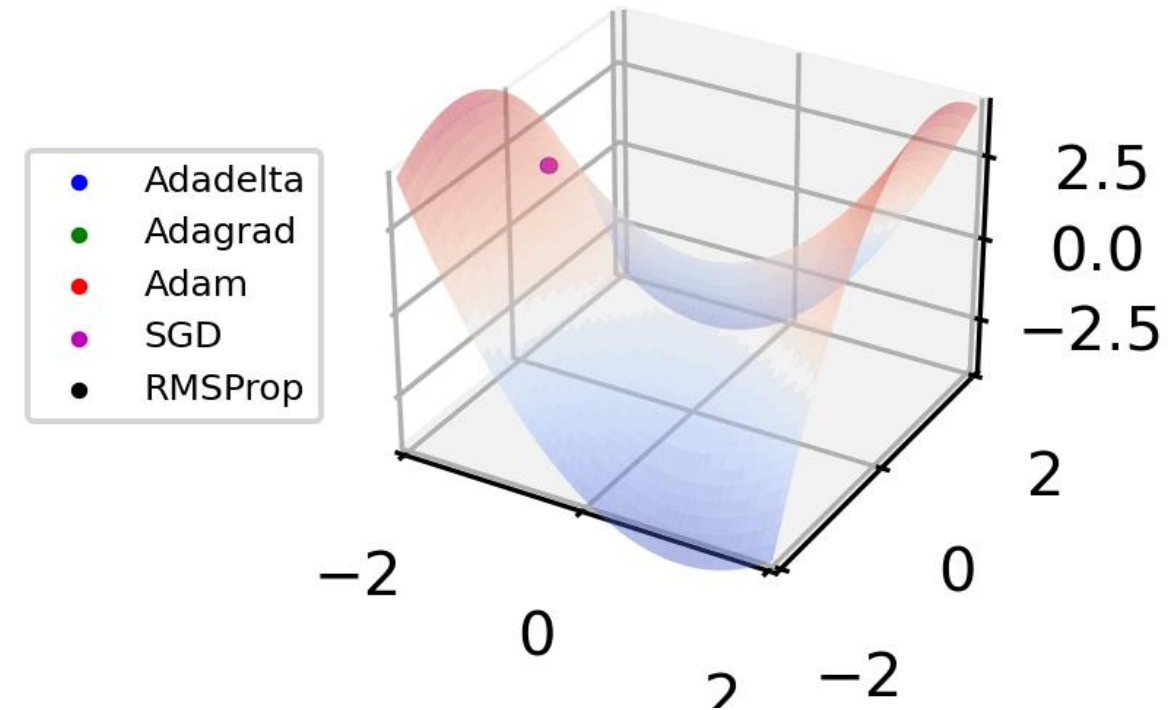


Neural Networks and Deep Learning

Training Process

Weight Adjustment – Optimization via Gradient Descent

- Gradient descent updates weights to minimize an objective function → loss function → error
- Optimizer follows the steepest descent (slope) on the loss function
- 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



Source: VirtualVistas - Own work,

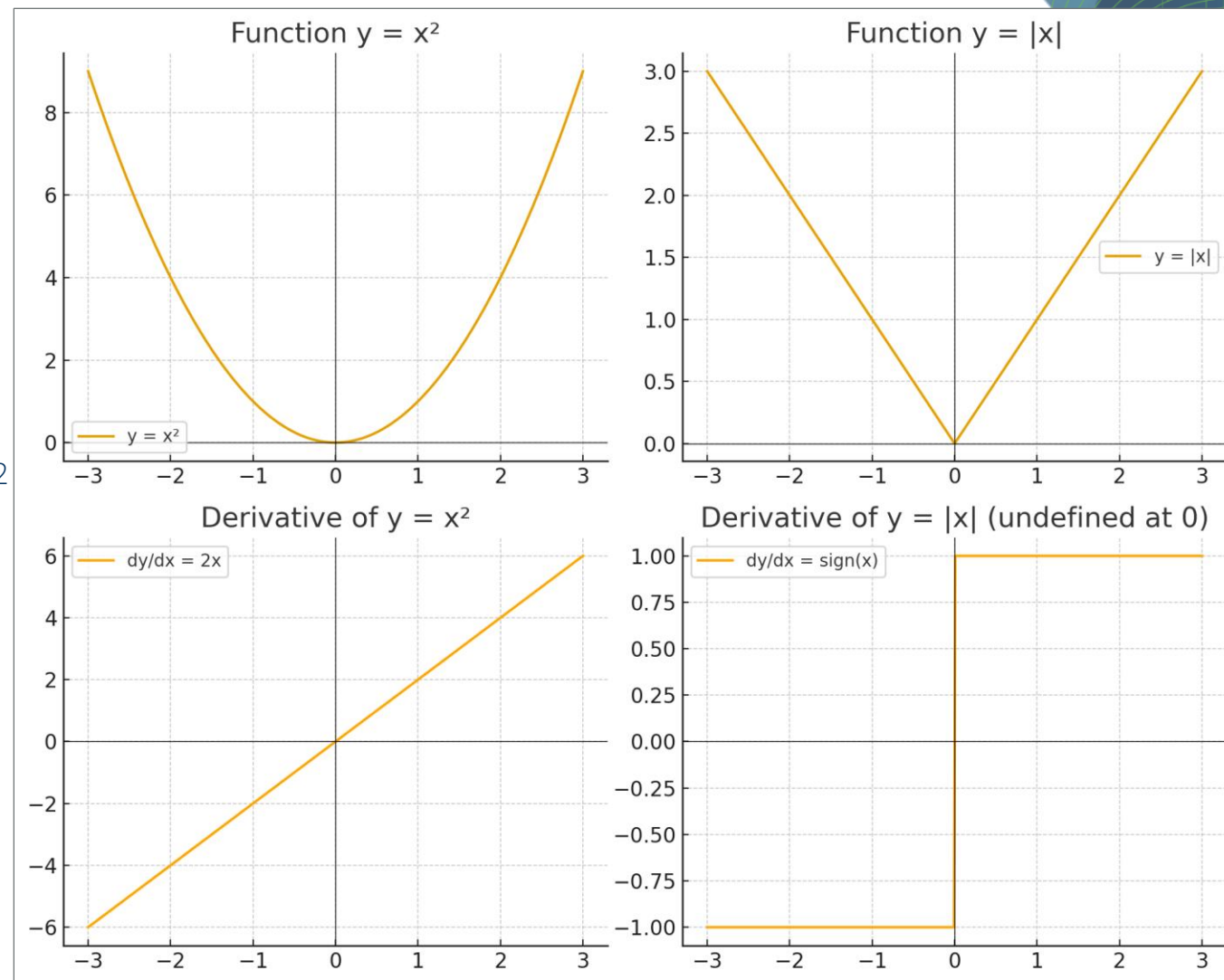
<https://commons.wikimedia.org/w/index.php?curid=139889895>, CC BY-SA 4.0

Neural Networks and Deep Learning

Training Process

Loss Function

- Needs to be a differentiable function to compute a slope
- Derivate must be defined for all points
- Example:
 - Mean Squared Error: $(y_{\text{true}} - y_{\text{predicted}})^2$



Neural Networks and Deep Learning

Training Process

Architecture and Training Parameters (Hyperparameters)

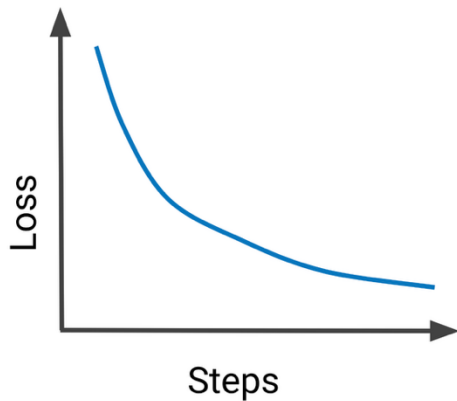
- Model parameters = learned by the neural network during training (weights, biases)
- Model architecture choices
 - Layer types and arrangement
 - Number of neurons per layer
 - Type of activation function
 - ...
- Training parameter
 - Number of samples / batches per epoch
 - Number of epochs
 - Optimization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

Neural Networks and Deep Learning

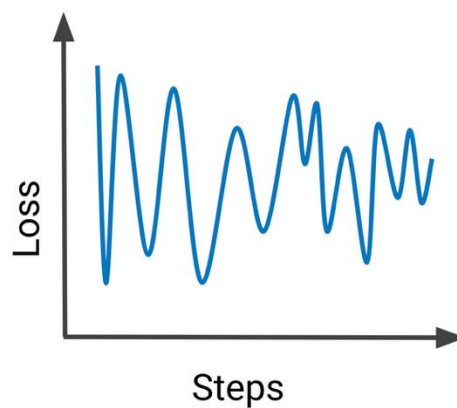
Training Process

But how can we check how well the training process is going?

- Plot the loss curve to see how the loss (error) develops over time (steps / epochs)

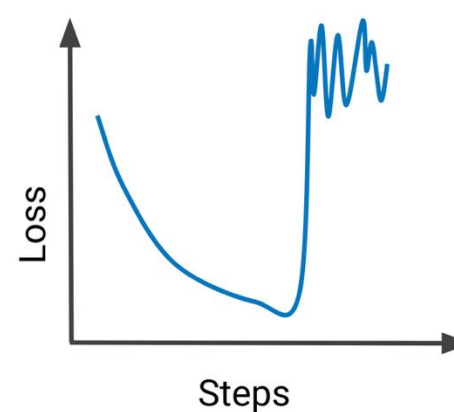


Ideal loss curve



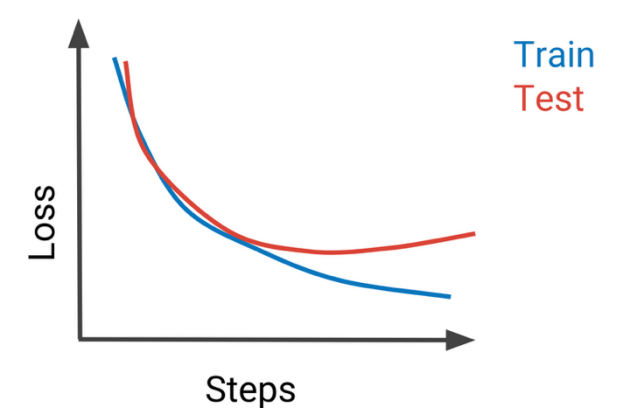
Oscillating training

- Bad training data
- Learning rate too high



Disturbed training

- Outliers in data
- Missing values (NaN)



Overfitting on training data
Bad generalizing for new data

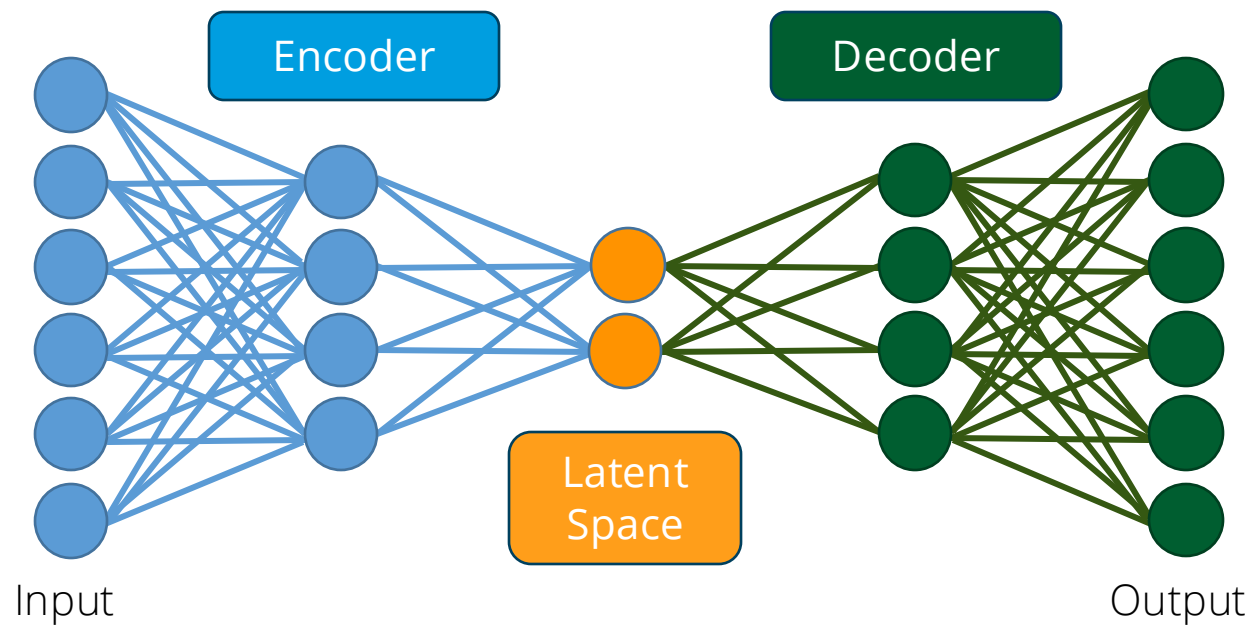
Source: <https://developers.google.com/>

Portions of this page are reproduced from work created and shared by Google and used according to terms described in the [Creative Commons 4.0 Attribution License](#).

Neural Network Architectures

Autoencoder

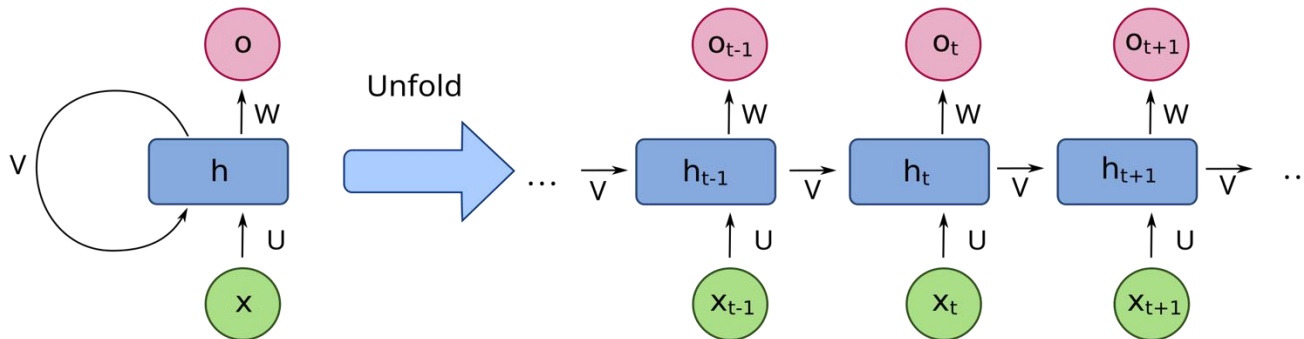
- Neural network for data compression (encoder) and reconstruction (decoder)
- Most important information (properties of features) get embedded in low-dimensional **latent space**
- Training aims to minimize the error during reconstruction
- Applications, e.g., dimensionality reduction or anomaly detection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

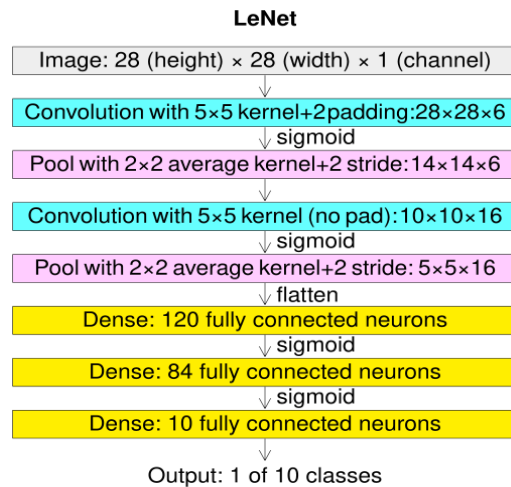
...

Source: fdeloche,
<https://commons.wikimedia.org/w/index.php?curid=60109157>, CC BY-SA 4.0

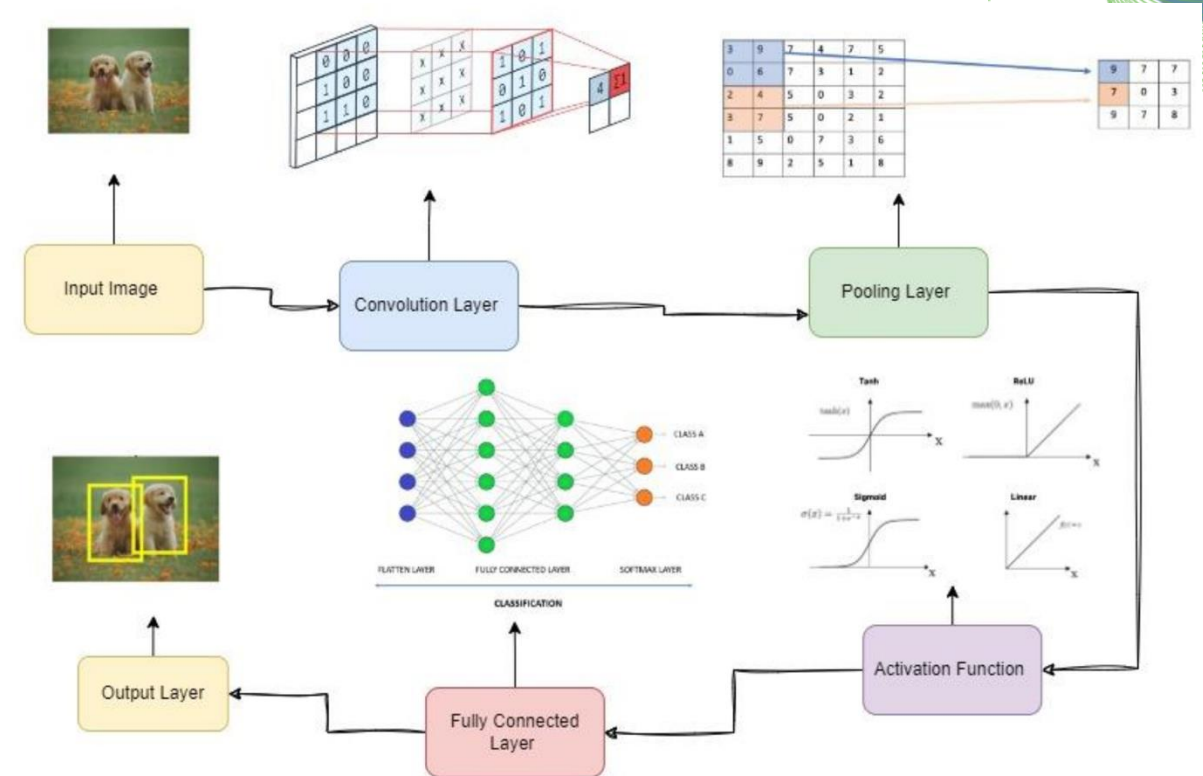
Neural Network Architectures

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)



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

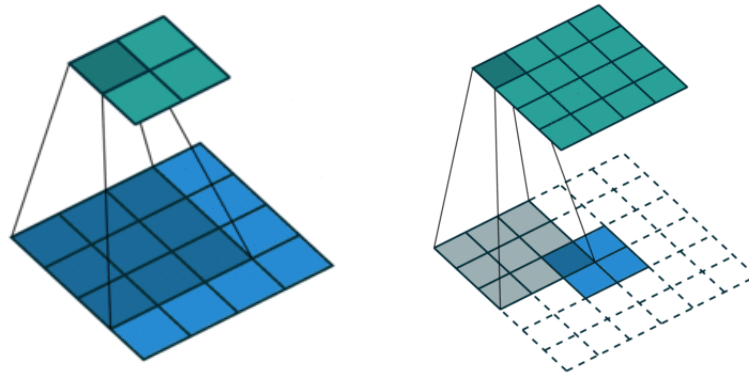
Neural Network Architectures

Convolutional Neural Network (CNN)

- Convolutional Layer – not 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

3	15	1	13
9	7	0	10
11	5	5	3
1	8	9	6

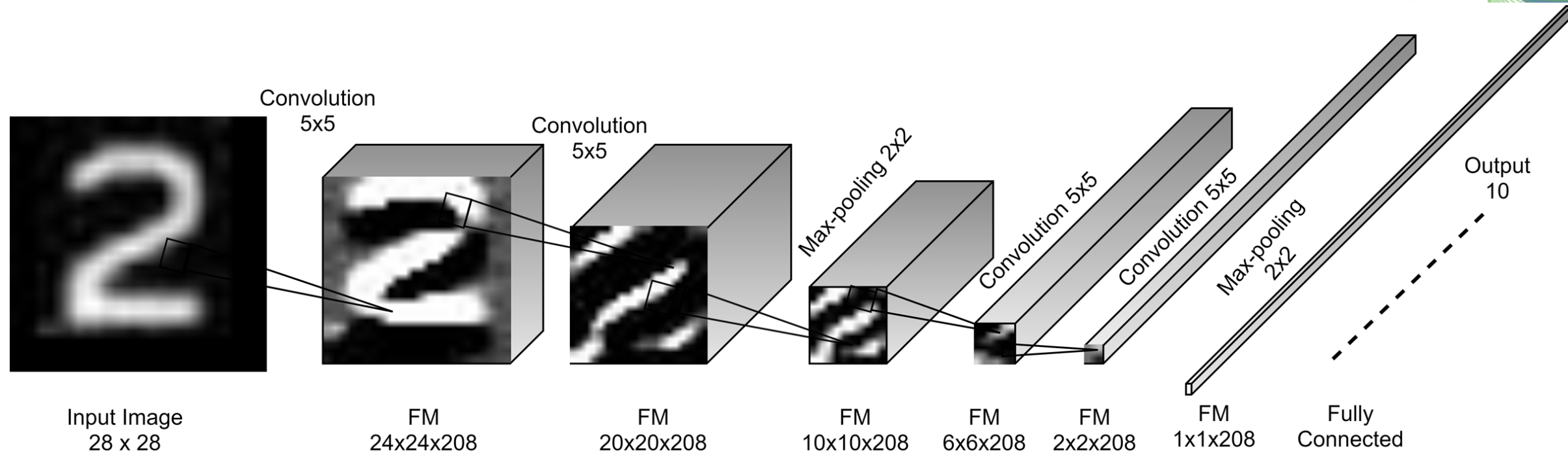
Max pooling

15	13
11	9

Neural Network Architectures

Convolutional Neural Network (CNN)

- Convolutional Neural Network – the full image

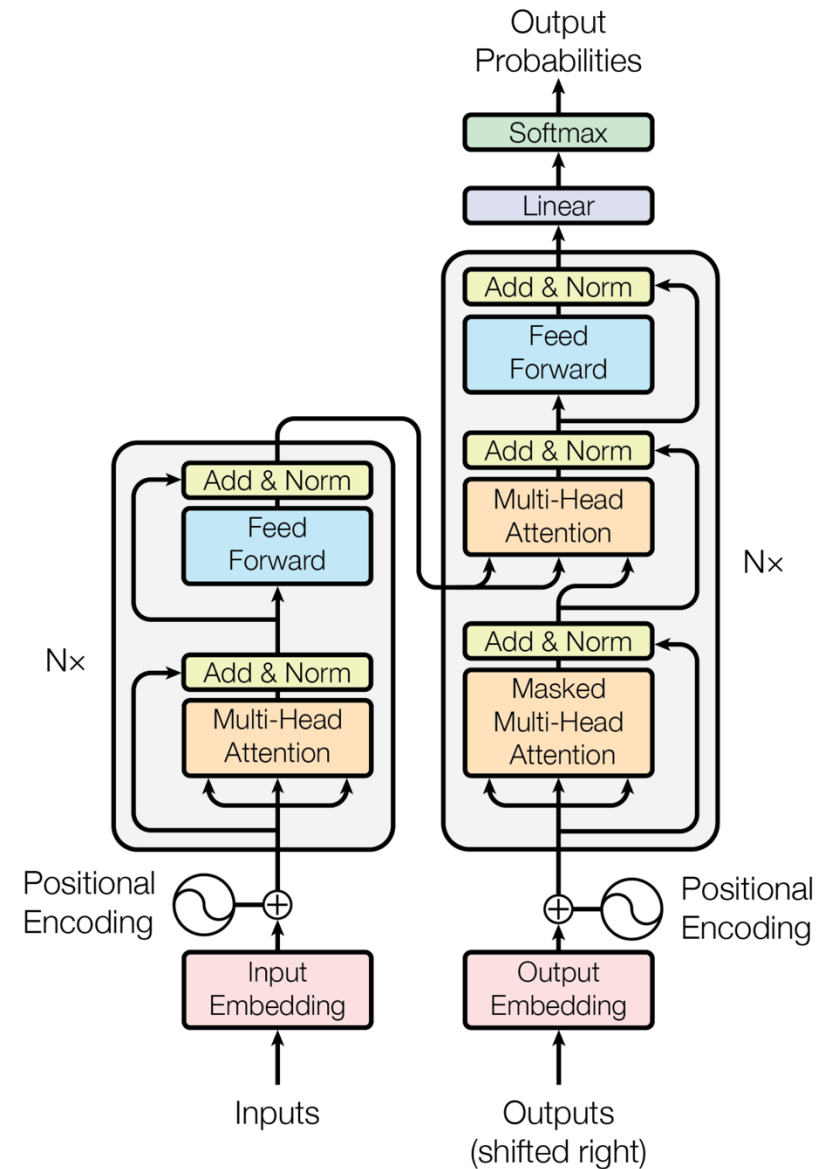


Source: Saleem M, Ertürk S. 2021. Full depth CNN classifier for handwritten and license plate characters recognition. PeerJ Computer Science 7:e576
<https://doi.org/10.7717/peerj-cs.576>, CC-BY4.0

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

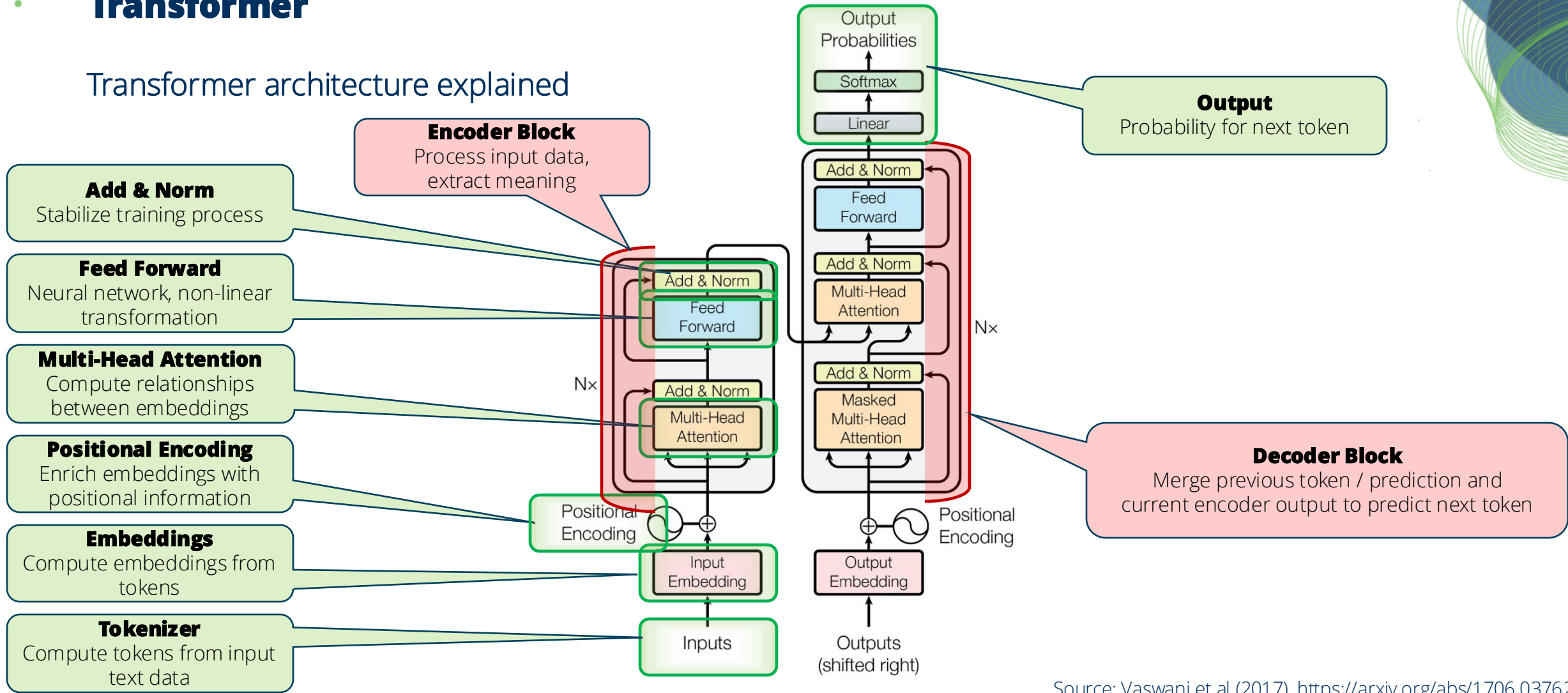


Quelle: Vaswani et al (2017), <https://arxiv.org/abs/1706.03762>

Neural Network Architectures

Transformer

Transformer architecture explained



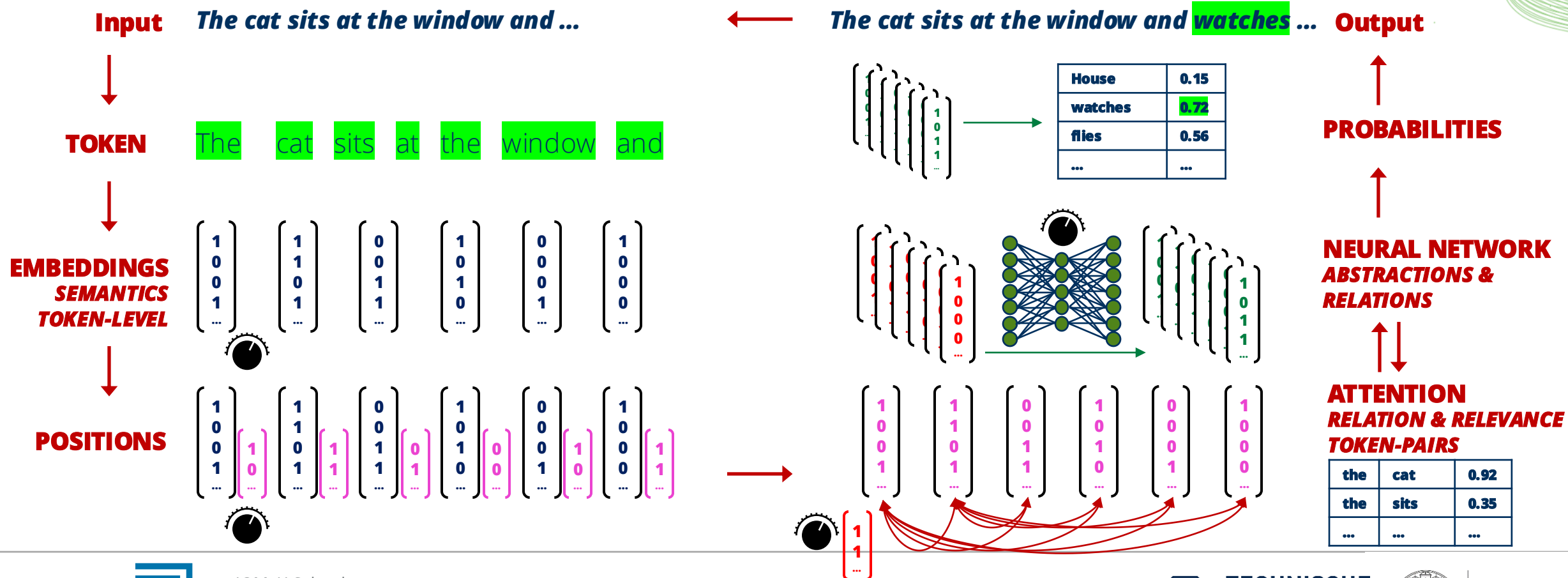
Source: Vaswani et al (2017), <https://arxiv.org/abs/1706.03762>

Neural Network Architectures

Transformer

Transformer – High level view on the process

Weights for almost all operations
"Adjustment screws"



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
 - [optuna](#) – open-source framework for hyperparameter optimization