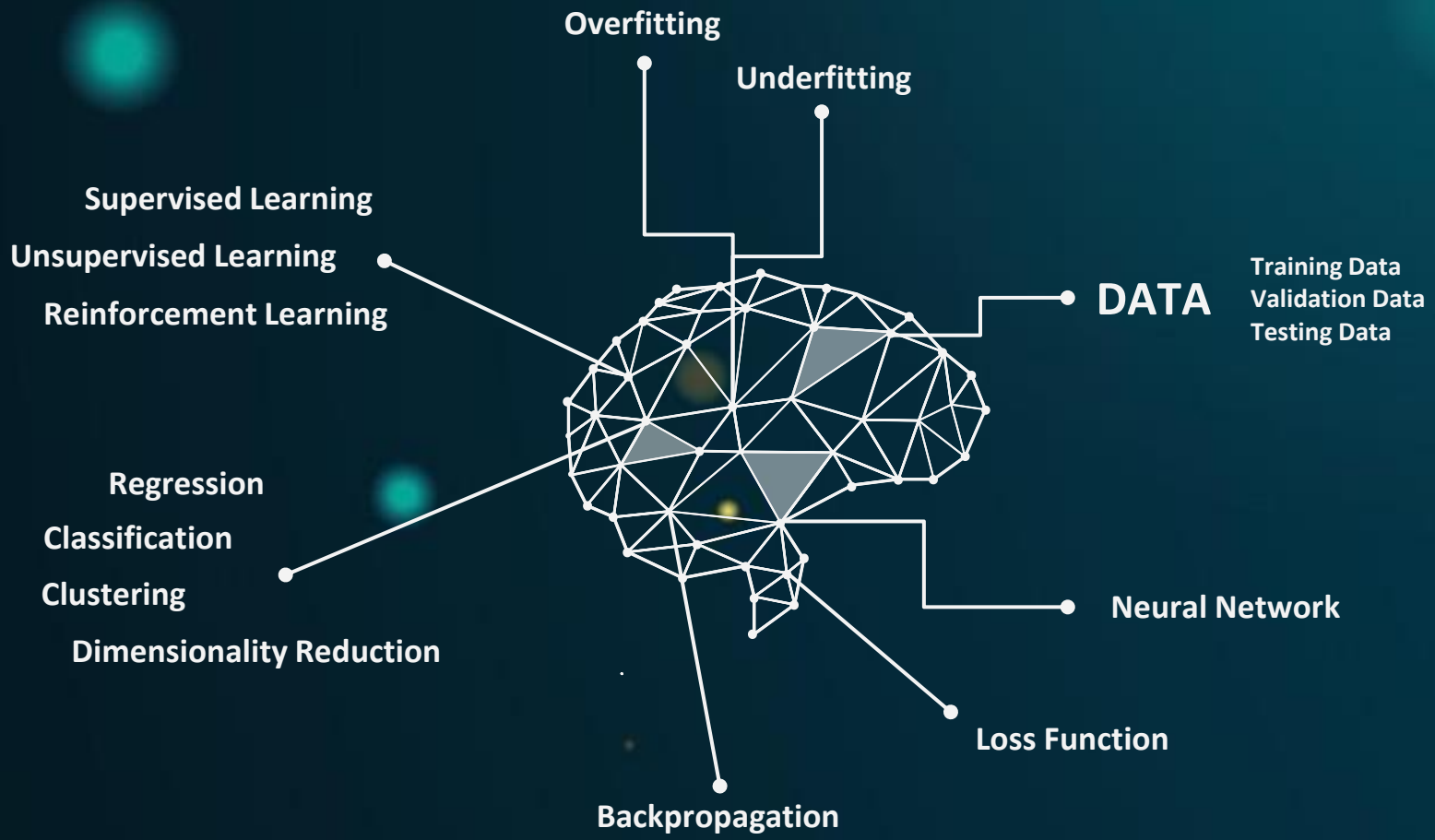
Deep Learning

Tac Pot #7 | 2021


Overfitting
Underfitting
Supervised Learning
Unsupervised Learning
Reinforcement Learning
DATA
Training Data
Validation Data
Testing Data
Regression
Classification
Clustering
Dimensionality Reduction
Neural Network
Loss Function
Backpropagation

01 **Einführung**
AI > ML > Deep Learning

02 **DL Basics**
Fundamentals

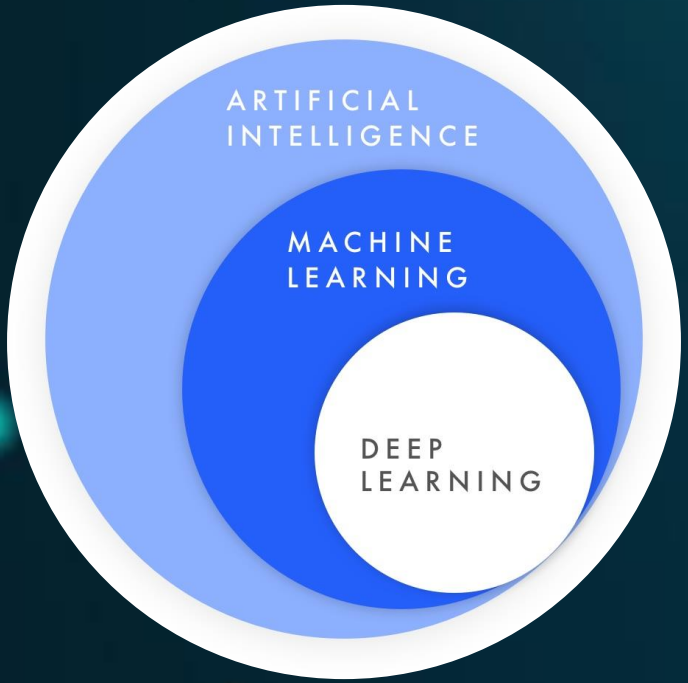
03 **Anwendungsbereiche**
Deep Learning in der Praxis

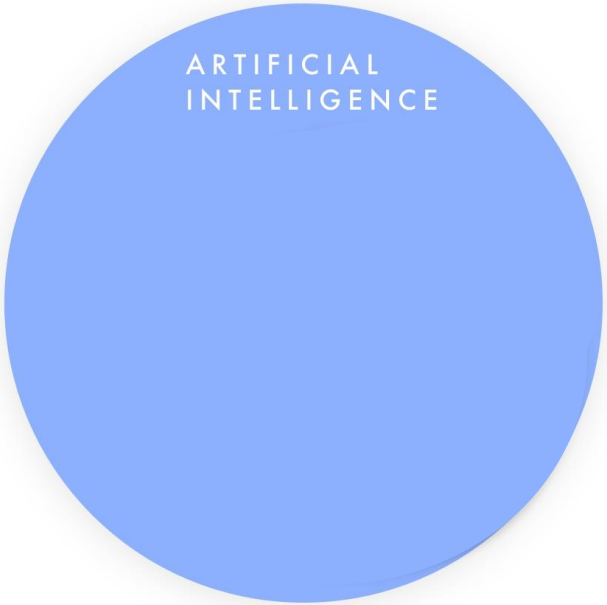
04 **Code Demo**
CIFAR-10 classification mit PyTorch & Wandb

Einführung

01

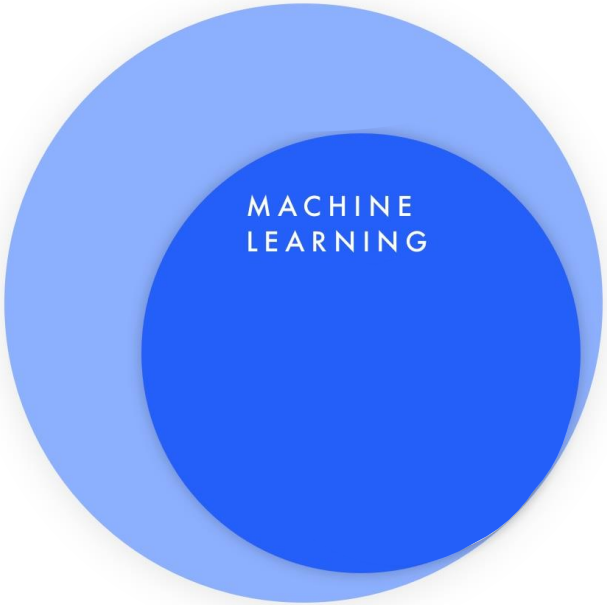
AI > Machine Learning > Deep Learning


ARTIFICIAL
INTELLIGENCE
MACHINE
LEARNING
DEEP
LEARNING



any technique that enables **computers to mimic human behavior**

*machines that **think** and **act** rational / like humans*

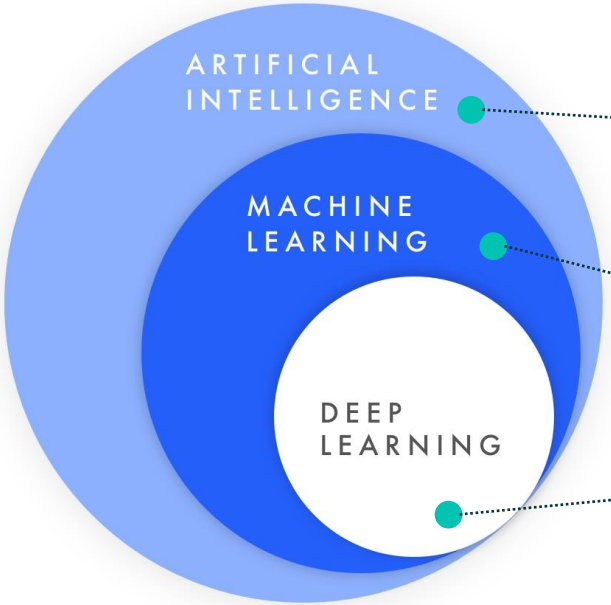


computers with the **ability to learn without being explicitly programmed**

*learn from **data** --> generalize or make predictions*

extract patterns from data using **Neural Networks**

*imitation of the **human brain***



intelligent machines thinking and acting rational

e.g. rule-based chatbot for the website of an insurance company

learning without being explicitly programmed

e.g. regression-based prediction of real estate price based on “size” & “area”

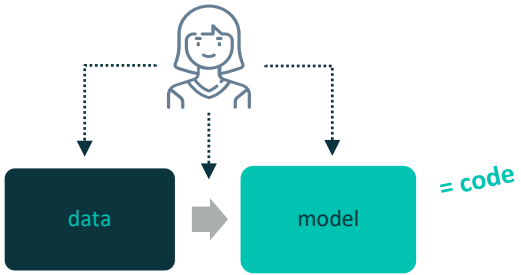
Neural Networks to extract patterns from data

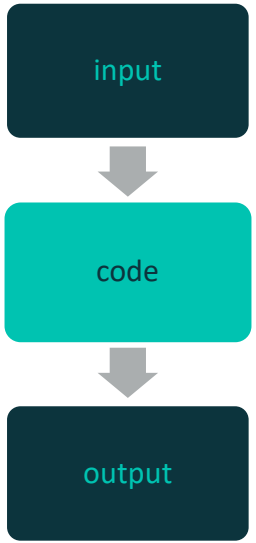
e.g. automatic traffic sign recognition using image classification

Traditional Programming **vs.** Machine Learning **?**

Traditional Programming

Machine Learning





development

operation

Machine Learning

Supervised Learning

Given Data
input **x** and output **y** (label)

Goal
Learn function to map
x -> y

example

This thing is an apple.

Unsupervised Learning

Given Data
only input **x** (no labels)!

Goal
Learn underlying structure

example

This thing is like the other thing.

Reinforcement Learning

Given Data
state-action pairs

Goal
maximize future rewards over many time steps

example

Eat this thing because it will keep you alive.

Machine Learning

Supervised Learning

Regression

Lineare Regression
Multiple Regression
...

Classification

Logistische Regression
Support Vector Machine (SVM)
Decision-Tree-Klassifikator
Random-Forest-Klassifikator
Boosted Decision-Trees
Neural Networks (Deep Learning)
...

Unsupervised Learning

Clustering

k-means clustering
hierarchical clustering
mean shift clustering
density-based clustering
...

Dimensionality Reduction

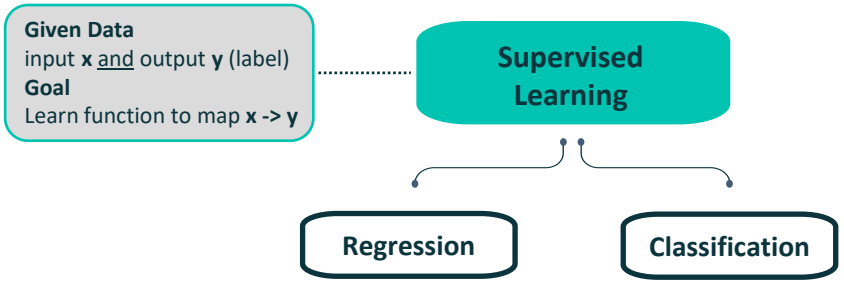
Principial Component Analysis (PCA)
...

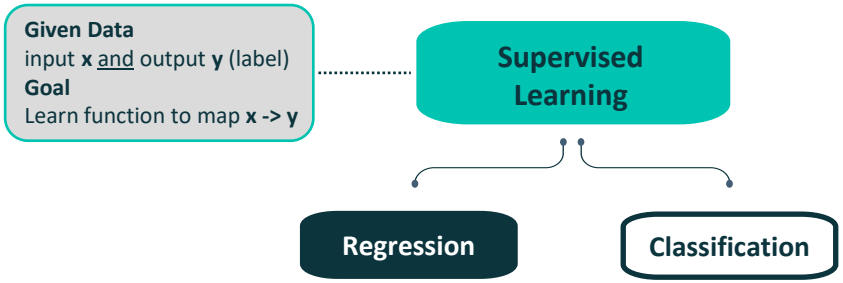
Reinforcement Learning

Model-Based

Model-Free

Value-Based (e.g. DQN)
Policy-Based (e.g. Policy Gradient)

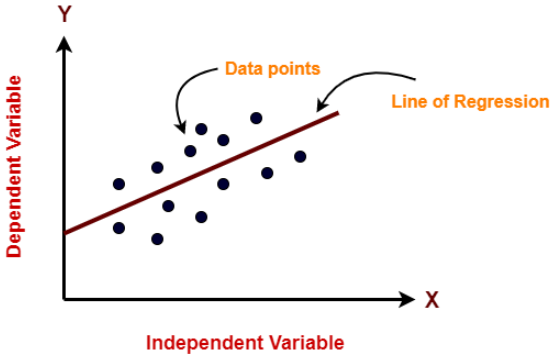

Given Data
input x and output y (label)
Goal
Learn function to map x -> y
Supervised Learning
Regression
Classification

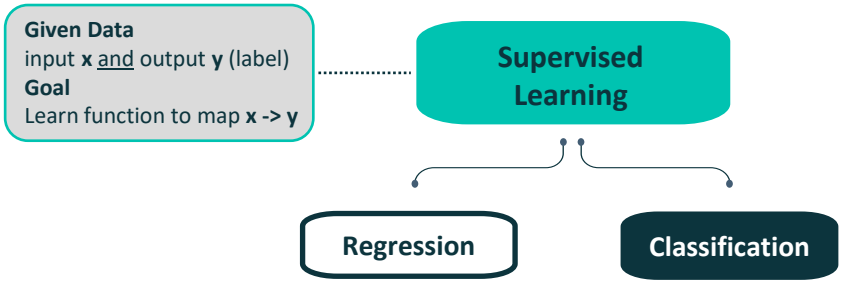


> predict one dependent (target) variable based on one or more independent (predictor) variables

> prediction of a **continuous** value

example:
> linear regression: number of hours learned -> points in the exam
> multiple regression: height + weight + age -> gender

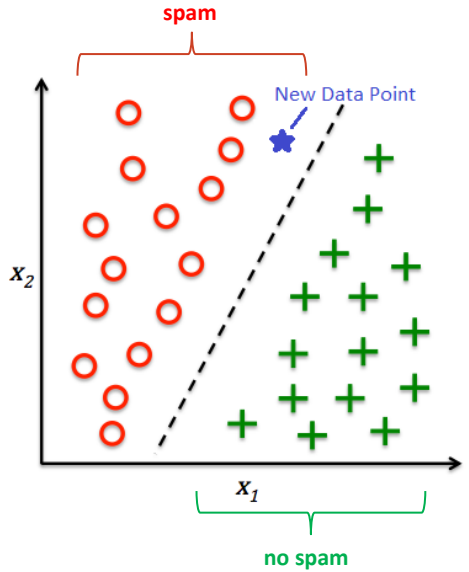


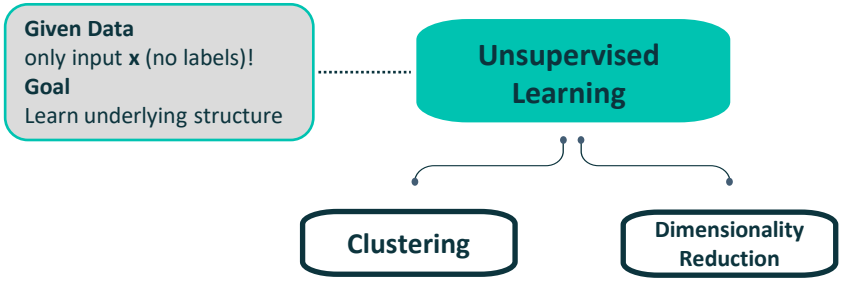


> identify the category (class) of new observations based on pre-classified training data

> prediction of a **categorical** value

examples:
> Yes or no, e.g. spam or not spam?
> Is this object an apple or an orange?

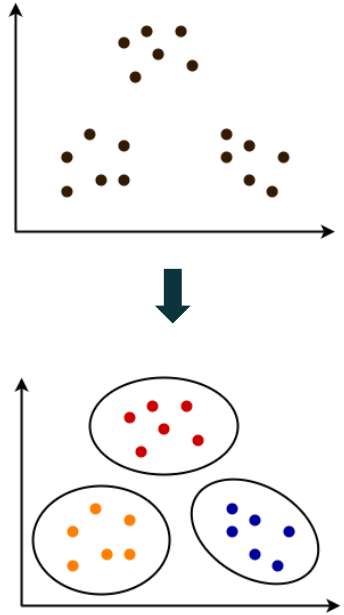



Given Data
only input x (no labels)!
Goal
Learn underlying structure
Unsupervised Learning
Clustering
Dimensionality Reduction

Given Data
only input **x** (no labels)!
Goal
Learn underlying structure

Unsupervised Learning

- **Clustering**
- **Dimensionality Reduction**



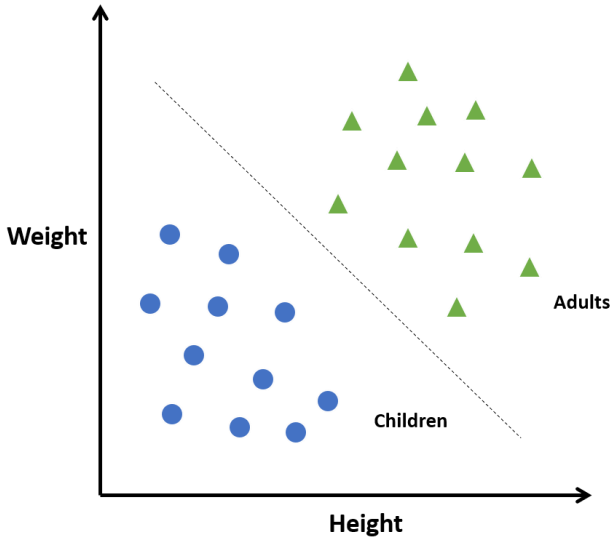
> group a set of data into different clusters based on the underlying structure of multiple attributes

> **find patterns** and draw inferences

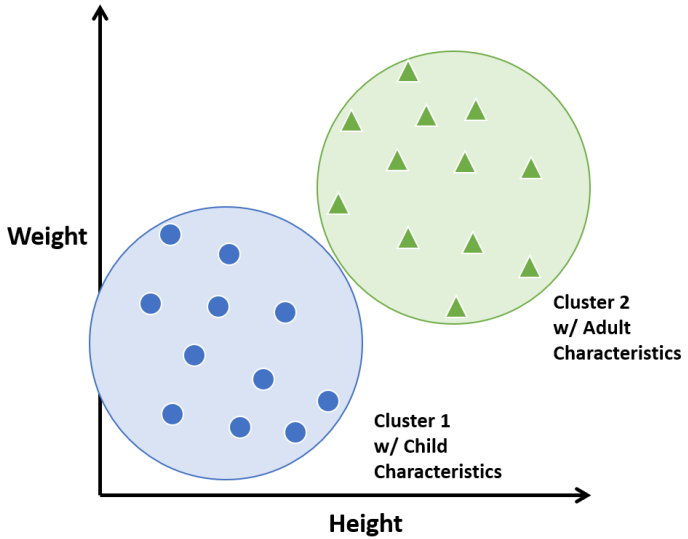
examples:
> *inhabitants of a new planet -> cluster into different species (based on height, # of legs, …)*
> *market research to segment the customers based on their choice and preferences*

Classification vs. Clustering ?

supervised (given labels)



unsupervised (no labels!)



Given Data
only input **x** (no labels)!
Goal
Learn underlying structure

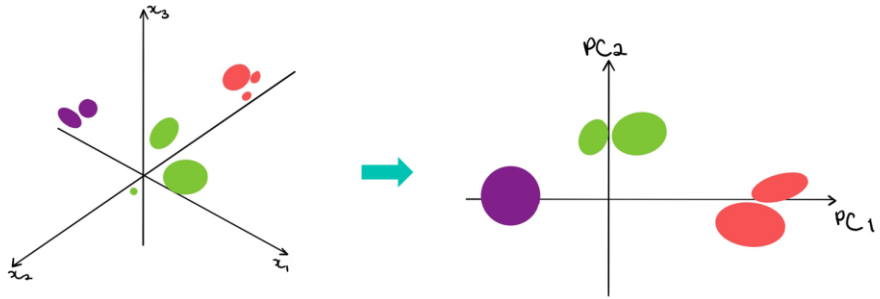
Unsupervised Learning

Clustering

Dimensionality Reduction

> reduce the dimension (= number of variables) of your dataset without loosing relevant information
… by **elimination** of variables
… by **aggregation** of variables

examples:
> eliminate the variable “exam date” for predicting the points in the exam
> aggregate the two variables “length” and “width” by creating one new variable “size”



x_1 = **length**
x_2 = speed
x_3 = **width**

PC_1 = speed
PC_2 = **size**

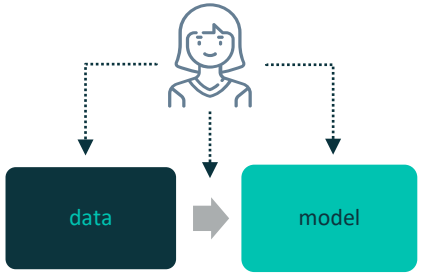
Reinforcement Learning

			Beispiel Schach
State	**s**	State, indem sich der Agent befindet	aktuelle Position aller Figuren
Action	**a**	Aktion, die der Agent ausführen kann	Läufer schlägt Turm
Reward	**r**	Reward, den der Agent erhält, wenn er im State s die Action a ausführt	+5 Bauerneinheiten
Policy	π (a|s)	Funktion: Mapping von States zu Actions	"persönliche Spielstrategie"

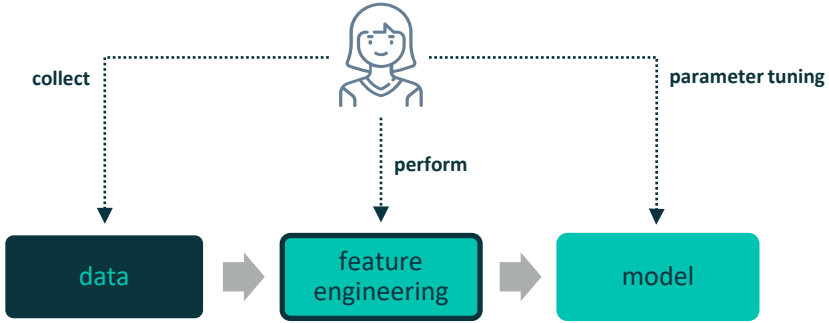
Ziel: maximiere die Summe der zukünftigen (diskontierten) Rewards $R_t = \sum_{i=t}^{\infty} y^i r_i$

Diskontierungsfaktor y: 0 < y < 1
y = 0: $R_{Zukunft} = 0$
y = 1: $R_{Gegenwart} = R_{Zukunft}$

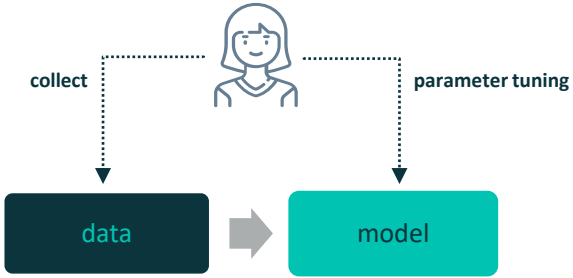
"Traditional Machine Learning" **vs.** Deep Learning **?**



Traditional Machine Learning



Deep Learning



Feature Engineering

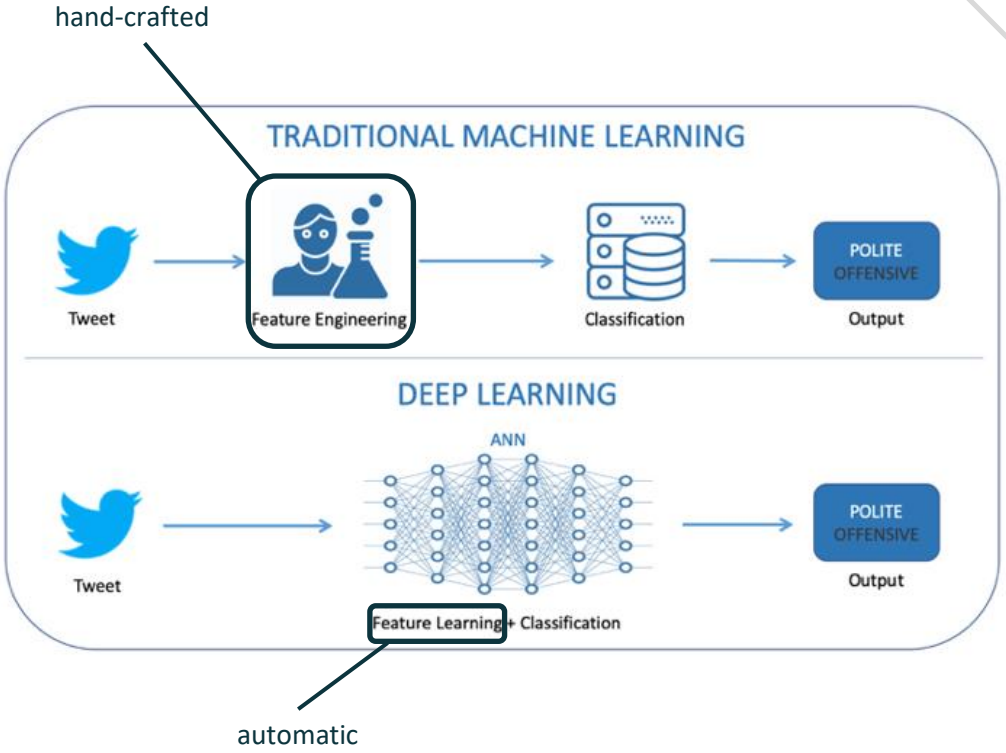
Features
= useful variables representing the most important characteristics / properties of the data

Feature Engineering
= process of using domain knowledge to extract features from raw data

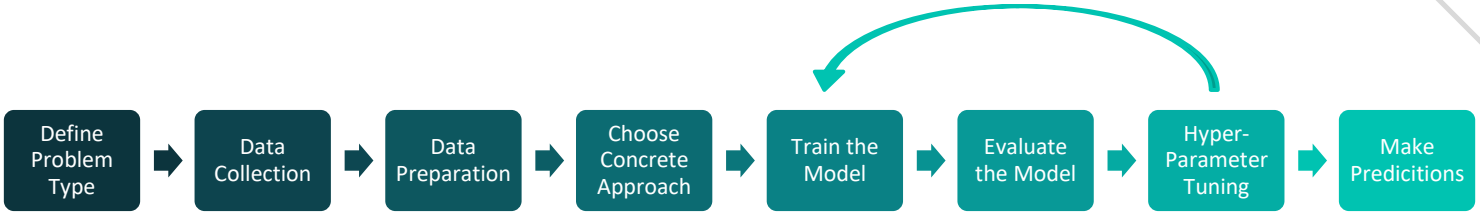
tweet example

raw data: 1.000.000 pre-classified tweets

> **feature 1**: # negative/positive words in the tweet
> **feature 2**: # "**!**" in the tweet
> **feature 3**: # author/tweet has been reported



General Approach: Machine Learning



Traditional Machine Learning	regression classification clustering dim. reduction ...	get as much data as possible	feature engineering **!**	**algorithms** e.g.: SVM, Decision Tree, ...	Training Data Validation Data Testing Data	Underfitting (High Bias) Sweet Spot Overfitting (High Variance)	reduce feature dimension, ...	use your model ☺
Deep Learning		get even more data!	chill ☺	choose an **architecture** for the NN			adjust learning rate / #epochs	

Artificial Intelligence | **intelligent machines** thinking and acting rational

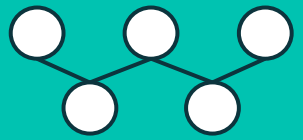
Machine Learning | **learning without being explicitly programmed**

hand-crafted features

- Supervised Learning
  - Regression
  - Classification
- Unsupervised Learning
  - Clustering
  - Dimensionality Reduction
- Reinforcement Learning
  - Model-Based
  - Model-Free

data → model (=code)

Deep Learning | **Neural Networks** to extract patterns from data



automatic feature learning

Define Problem Type → Data Collection → Data Preparation → Choose Concrete Approach → Train the Model → Evaluate the Model → Parameter Tuning → Make Predicitions

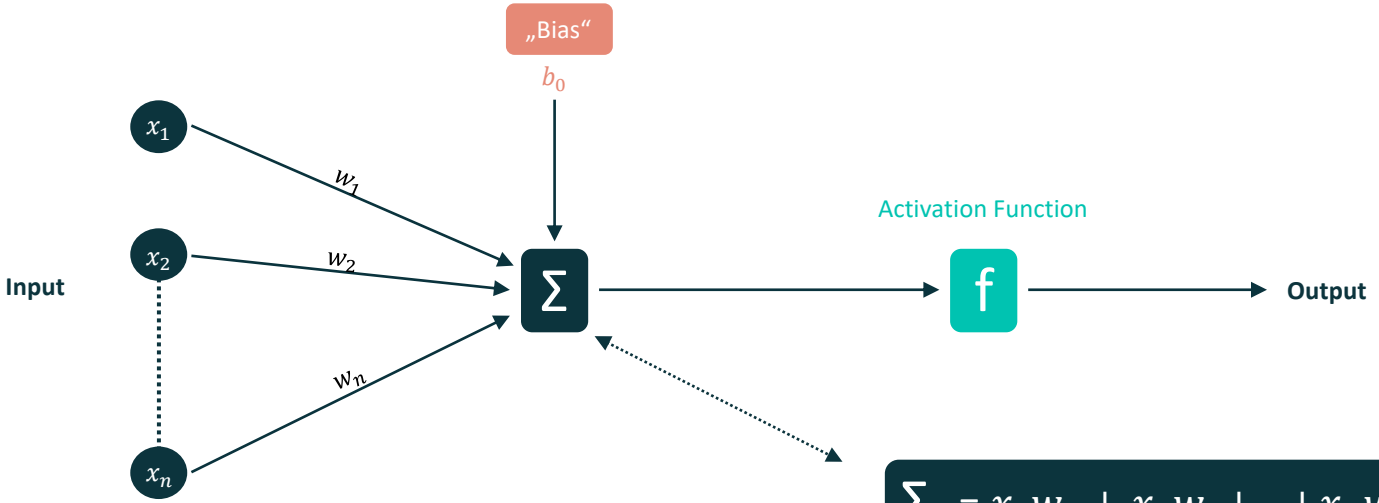
DL Basics

02

Fundamentals

Was macht ein Neuron?

Ein künstliches Neuron hat als Eingabe eine Reihe von Werten, aus denen es einen einzelnen neuen Ausgabewert berechnet.



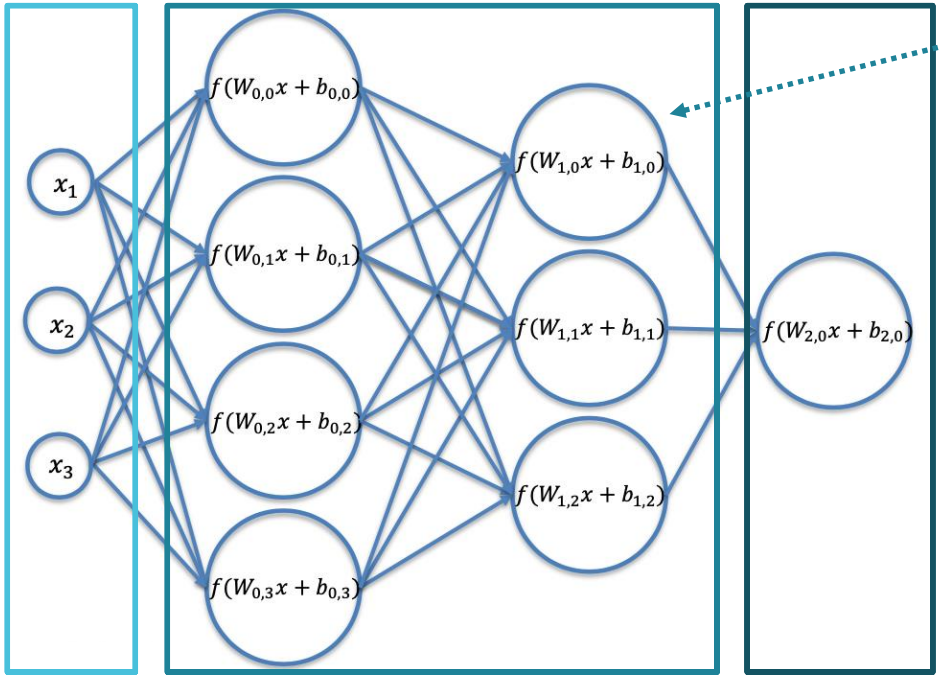
$$\Sigma = x_1 w_1 + x_2 w_2 + .. + x_n w_n + b_0$$

1. Jede Input Variable x_i wird mit einem Gewicht w_i multipliziert
2. Summiere alle Ergebnisse aus 1. und füge ein Bias b hinzu
3. Aktiviere das Ergebnis aus 2. mit einer Activation Function f

code

```
def neuron (x1,x2,x3):
    sum =  w0  +  x1*w1  +  x2*w2  +  x3*w3
    return sigmoid(sum)
```

Was ist ein Neuronales Netz?



$$f_N = f(\boldsymbol{W}_2 * f(\boldsymbol{W}_1 * f(\boldsymbol{W}_0 * \boldsymbol{x} + \boldsymbol{b}_0) + \boldsymbol{b}_1) + \boldsymbol{b}_2)$$

Neuron

$$f(W^T * x + b)$$

W = Weight Vector
x = Feature Vector
b = Bias
f = Activation function

$$f_N^{2-layers} = f(\boldsymbol{W}_1 * f(\boldsymbol{W}_0 * \boldsymbol{x}))$$
$$f_N^{3-layers} = f(\boldsymbol{W}_2 * f(\boldsymbol{W}_1 * f(\boldsymbol{W}_0 * \boldsymbol{x})))$$
$$f_N^{4-layers} = f(\boldsymbol{W}_3 * f(\boldsymbol{W}_2 * f(\boldsymbol{W}_1 * f(\boldsymbol{W}_0 * \boldsymbol{x}))))$$
$$f_N^{5-layers} = f(\boldsymbol{W}_4 * f(\boldsymbol{W}_3 * f(\boldsymbol{W}_2 * f(\boldsymbol{W}_1 * f(\boldsymbol{W}_0 * \boldsymbol{x})))))$$

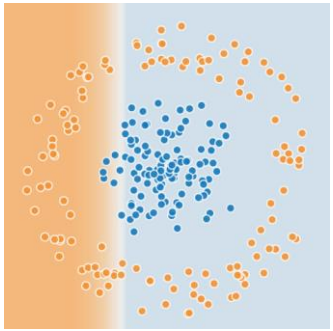
Neuronale Netze sind ineinander geschachtelte Funktionen

Wieso brauchen wir Activation Functions ?

Können wir nicht einfach **linear layers** hintereinanderschalten ?

$$f_N = W_2 * (W_1 * (W_0 * x)))$$
$$\widehat{W} = W_2 * W_1 * W_0$$
$$f_N = \widehat{W} * x$$

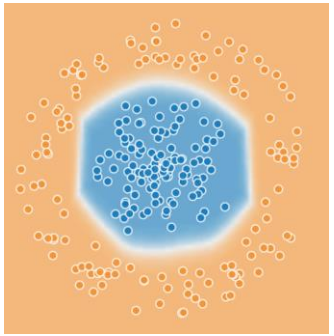
Nur lineare decision boundaries möglich, da immer noch lineare Funktion



Lösung: Füge nicht-Linearität hinzu

$$f_N = max(0, W_2 * max(0, W_1 * max(0, W_0 * x)))$$

Nicht lineare decision boundaries möglich, da Funktion jetzt nicht - lineare Zusammenhänge abbilden kann

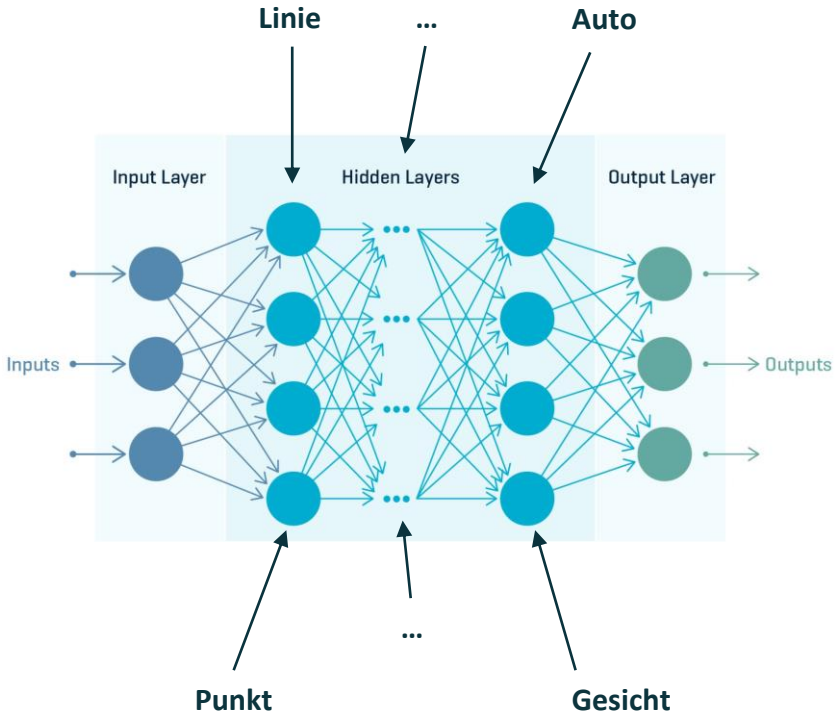


Activation Functions

- **Sigmoid** $\frac{1}{1+e^{-x}}$
- **TanH** $tanh(x)$
- **ReLU** $max(0, x)$

Activation Functions sind essentiell damit ein Neuronales Netwerk nicht lineare Funktionen darstellen kann

Wieso werden Neuronale Netze in Layer geordnet ?



Struktur = Σ lower-level Strukturen

Viele Strukturen in der echten Welt sind eine Komposition von wiederum anderen „lower-level" Strukturen

Beispiel

Gesicht = Σ Auge + Nase + …

Auge = Σ Pupille + Wimpern + …

Pupille = Σ Punkte + Linien + …

Die **hierarchische Struktur** von neuronalen Netzen soll das **Lernen solcher Kompositionen** erleichtern

Wie macht ein Neuronales Netzwerk eine Prediction $(\hat{y})$?

Input (x)

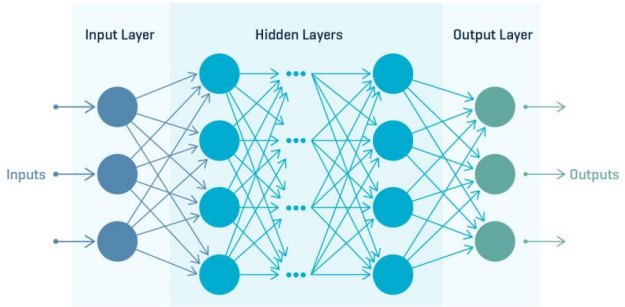
- High level features (z.B. Alter)
- Bilder
- Text
- …

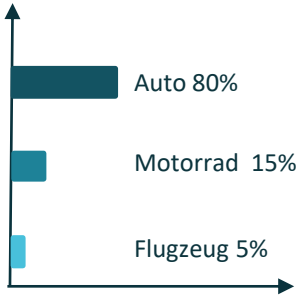
Neuronales Netzwerk

$$f_N(x;\theta) = \hat{y}$$

Prediction $(\hat{y})$

- Wahrscheinlichkeit (Klassifizierung)
- Continuous Variable (Regression)
- Parameter einer Distribution (Generative Models)





$$f_N(x;\theta) = \hat{y}$$

Indem wir die Parameter (θ) ändern erhalten wir für den selben Input (x) eine andere Prediction $(\hat{y})$

Wie wissen wir ob die Prediction $(\hat{y})$ gut ist ?

Input (x)

- High level features (z.B. Alter)
- Bilder
- Text
- …

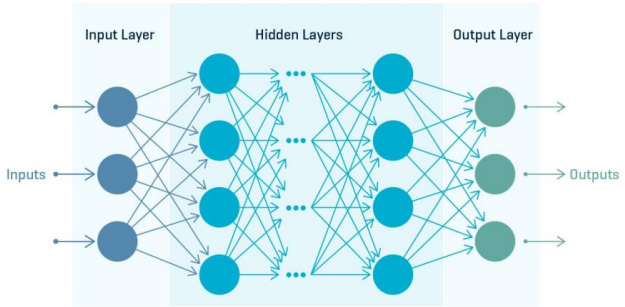
Neuronales Netzwerk

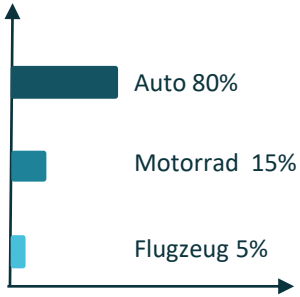
$$f_N(x;\theta) = \hat{y}$$

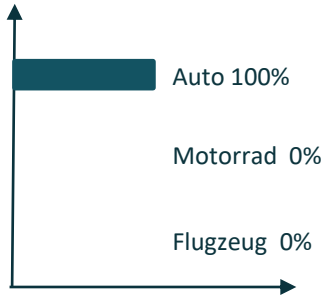
Prediction $(\hat{y})$

- Wahrscheinlichkeit (Klassifizierung)
- Continuous Variable (Regression)
- Parameter einer Distribution (Generative Models)

Target (y)







$\hat{y}$ ⟷ y

Distanz

➡ **Messe die Distanz zwischen der Prediction $(\hat{y})$ und dem Target (y)**

Loss Function

Die Loss function beschreibt mathematisch die Distanz der Prediction $(\hat{y})$ zum Target (y)

Je nach Problem muss eine andere Loss Function gewählt werden:

Regression - Mean squared error loss

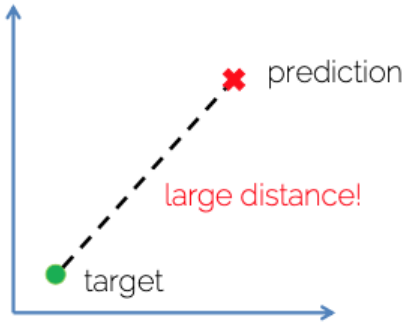
$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Target Prediction

Klassifizierung - Binary cross entropy loss

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} y_i * \log(\hat{y}_i) + (1 - y_i) * \log(1 - \hat{y}_i)$$

Target Prediction



bad prediction

good prediction

Remember

$$f_N(x;\ \theta) = \hat{y}$$

Indem wir (θ)ändern, ändern wir $J(\theta)$

Die Wahl der Loss Function ist extrem wichtig !

Wenn die Loss Function die Distanz zwischen Prediction und Target schlecht oder falsch beschreibt lernt das Netzwerk falsch

Loss Function - Beispiel

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f_N(x_i;\theta))^2$$

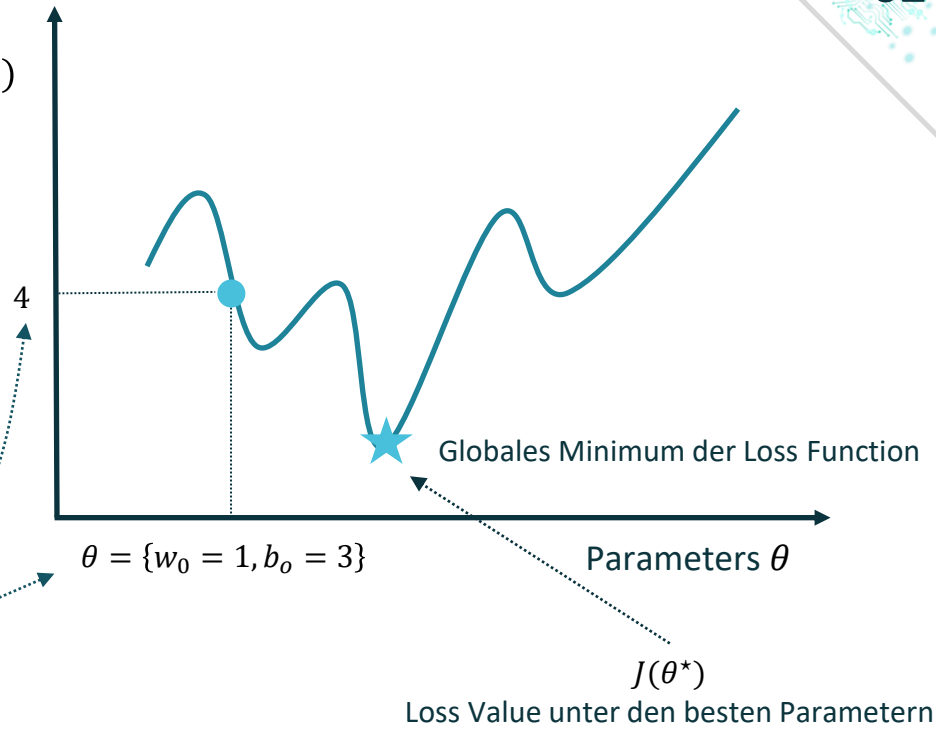
Target — Prediction

Beispiel

$x = 2$
$y = 7$
$\theta = \{w_0, b_o\}$
$w_0 = 1, b_0 = 3$

$f_N(x;\theta) = x * w_0 + b_o$
$= 2 * 1 + 3 = 5$

$J(\theta) = (y - f_N(x;\theta))^2$
$= (7 - 5)^2$
$= 4$



➡ **Ziel ist es die Parameter θ zu finden, für die die Loss Function $J(\theta)$ am kleinsten ist : $\underset{\theta}{\operatorname{argmin}}\, J(\theta)$**

Loss Function

Input (x)

- High level features (z.B. Alter)
- Bilder
- Text
- …

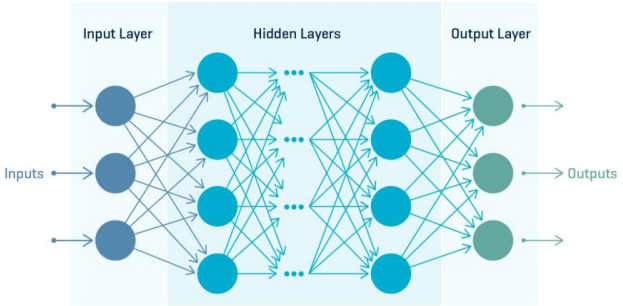
Neuronales Netzwerk

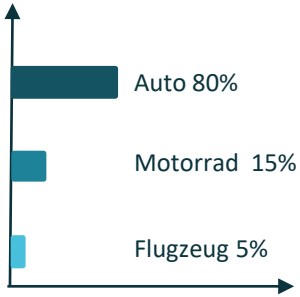
$$f_N(x;\theta) = \hat{y}$$

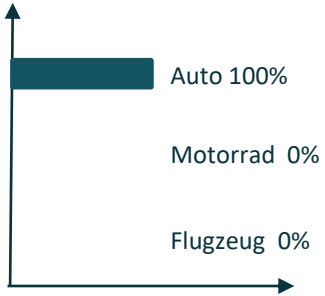
Prediction $(\hat{y})$

- Wahrscheinlichkeit (Klassifizierung)
- Continuous Variable (Regression)
- Parameter einer Distribution (Generative Models)

Target (y)







Wir müssen die Parameter des Netzwerks $\boldsymbol{\theta}$ so ändern, dass die Prediction nächstes mal besser ist, i.e. die Distanz zwischen $\hat{\boldsymbol{y}}$ und $\boldsymbol{y}$ kleiner wird.

$$\underset{\theta}{\operatorname{argmin}}\, J(\theta)$$

$\hat{y}$ ⟷ y

Distanz

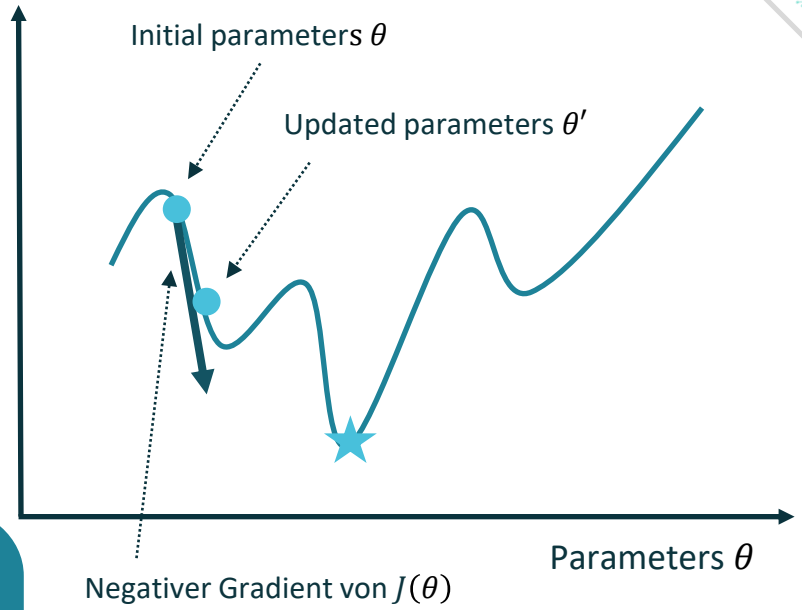
$J(\theta)$

Woher wissen wir wie θ geändert werden muss ?

Gradient Descent

1. Initialisiere Parameter (θ) zufällig
2. Wiederhole bis Konvergenz
    1. Berechne Gradienten der Loss Function : $\nabla_\theta J$
    2. Update $\theta : \theta \leftarrow \theta - \alpha \nabla_\theta J$

Learning Rate



Für jeden Parameter des Netzwerks „schauen“ wir uns an wie sich $J(\theta)$ ändern würde, wenn wir den Parameter ändern

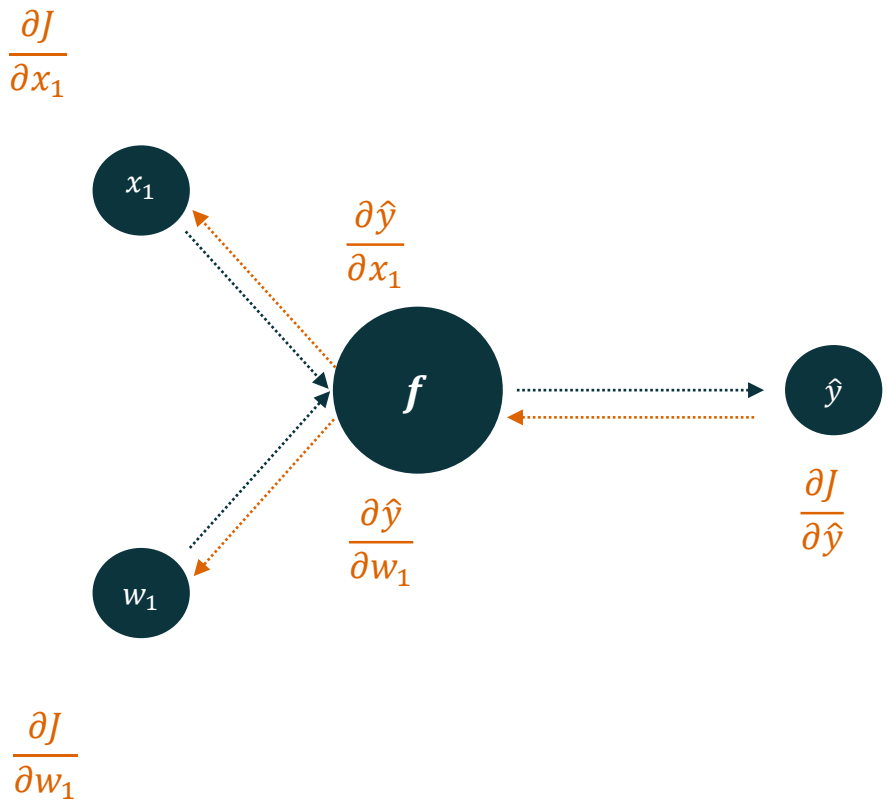
Wie können wir den Gradienten berechnen ?

Backpropagation

$$\nabla_\theta J = \begin{bmatrix} \frac{\partial J}{\partial w_1} \\ \dots \\ \frac{\partial J}{\partial w_n} \end{bmatrix}$$

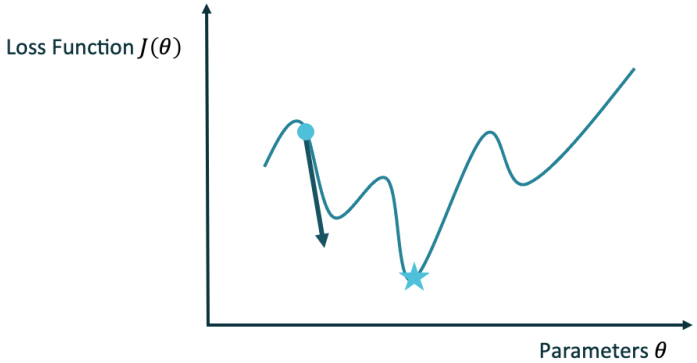
Note : Gradient von der Loss Function w.r.t. allen Parametern, also auch b_i (bias)

Backpropagation



$$J(\theta) = (y_i - \hat{y}_i)^2$$

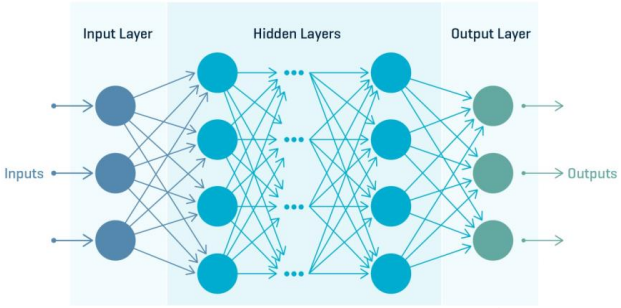
$$\frac{\partial J}{\partial \hat{y}} = -2(y_i - \hat{y}_i)$$



Neuronales Netzwerk Training Cycle

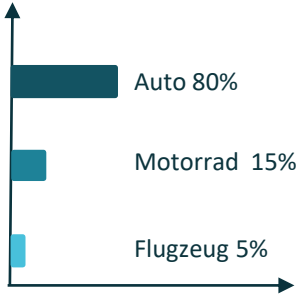
Input (x)

Neuronales Netzwerk

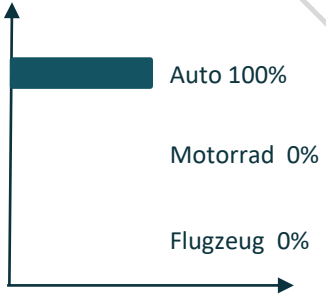


Forward Pass

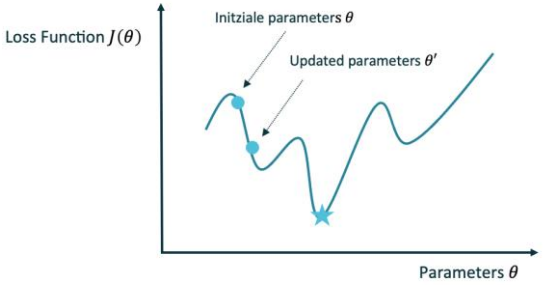
Prediction $(\hat{y})$



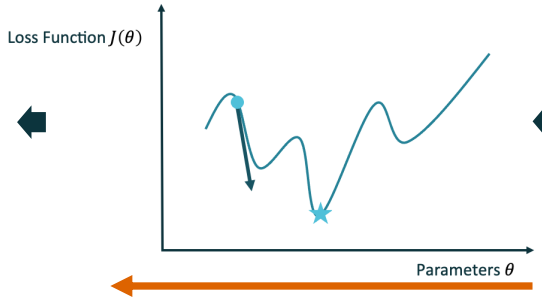
Target (y)



Update Parameter $\boldsymbol{\theta \leftarrow \theta - \alpha \nabla_\theta J}$



Berechne Gradienten $\nabla_\theta J$



Backward Pass

Berechne Loss Function $J(\theta)$

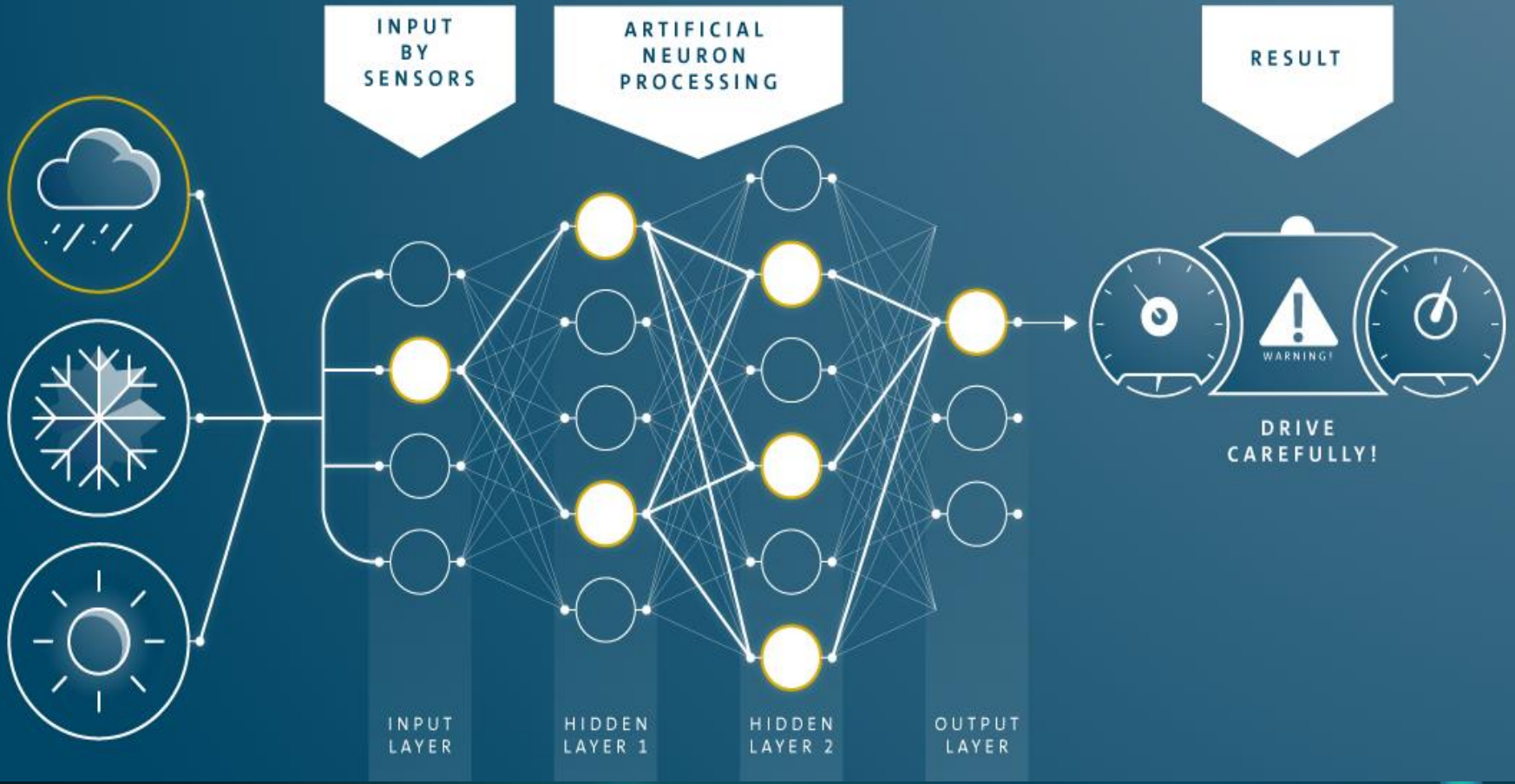
$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Anwendungs-bereiche

03

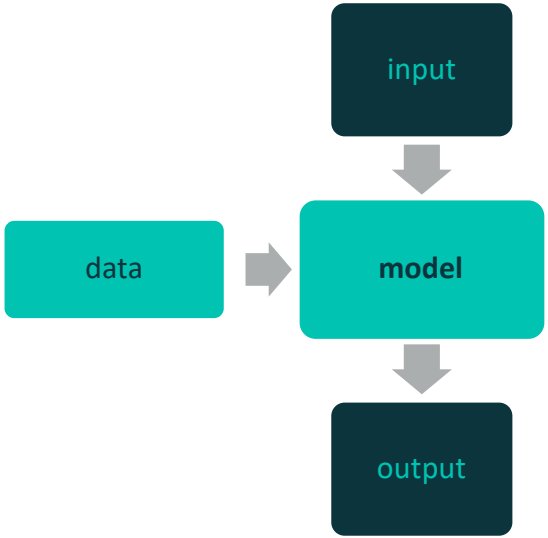
Deep Learning in der Praxis

Simplified example: how slippery is the road?

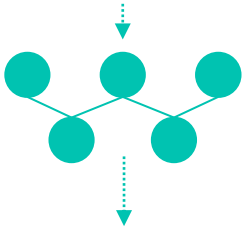


In welchen Bereichen kann man Deep Learning anwenden?

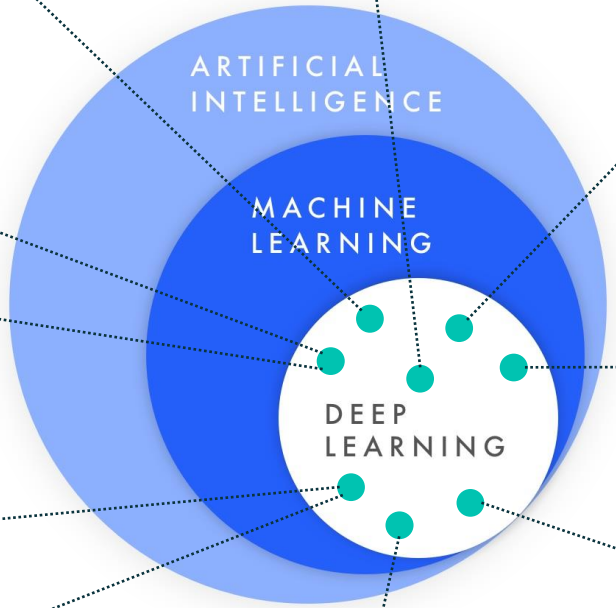
Einfach gesagt in (allen) Bereichen, in denen man mit **Daten** sinnvoll arbeiten kann.



numbers
text
audio
images (videos)



(transformed) numbers / text / audio / images (videos)
predictions
actions



Robotics
Boston Dynamics

Self Driving Cars

Virtual Assistants
Siri, Alexa

Art
Deep Dreaming

Composing Music

Entertainment
Netflix / Spotify recommendations

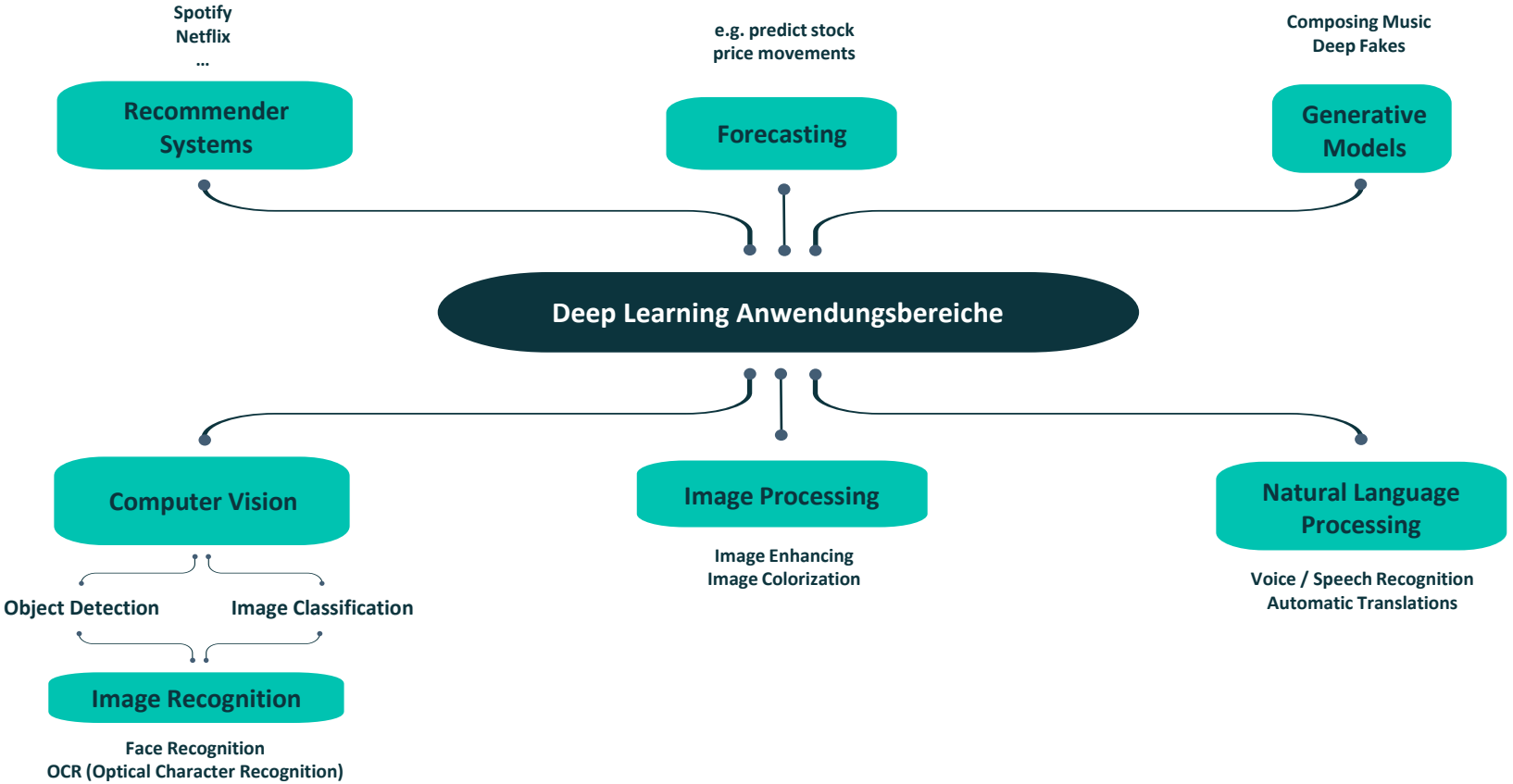
Fraud Detection
spam emails

vs.

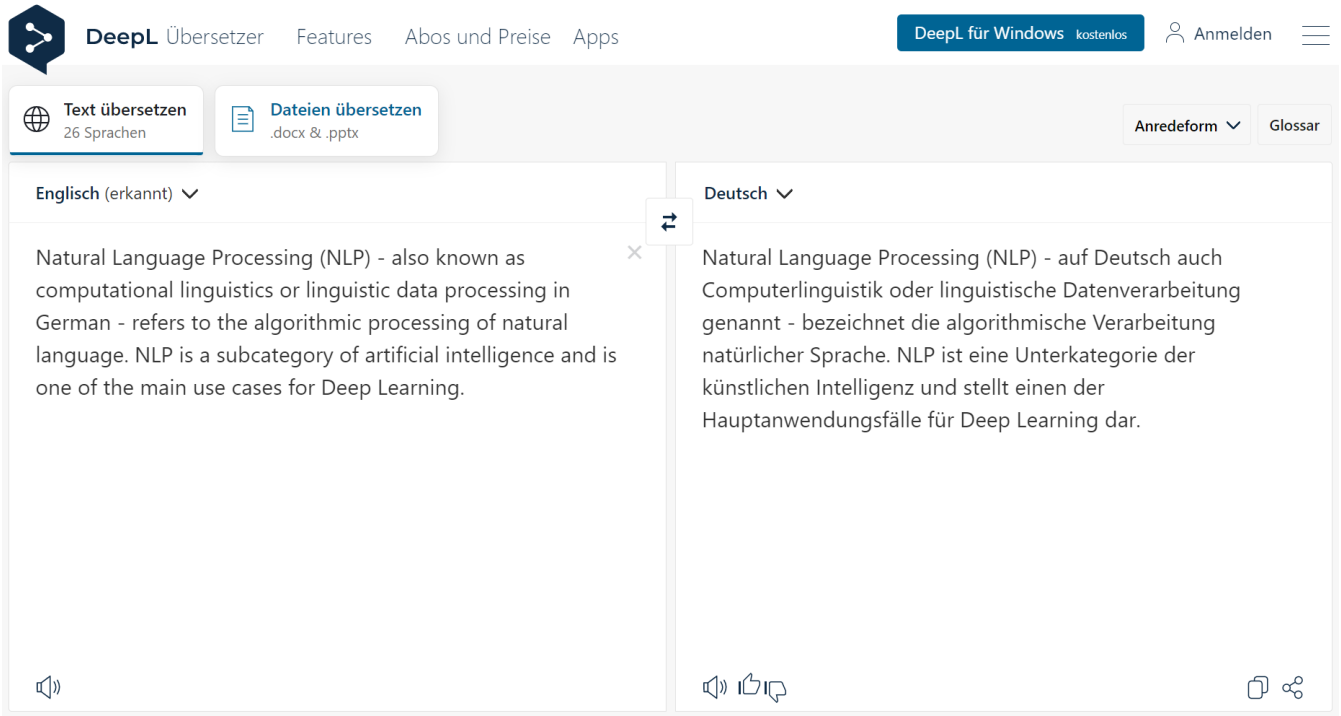
Deep Fakes

Games
Chess, Go, Starcraft

Healthcare
computer-aided disease detection


Spotify
Netflix
...
Recommender Systems
e.g. predict stock price movements
Forecasting
Composing Music
Deep Fakes
Generative Models
Deep Learning Anwendungsbereiche
Computer Vision
Object Detection
Image Classification
Image Recognition
Face Recognition
OCR (Optical Character Recognition)
Image Processing
Image Enhancing
Image Colorization
Natural Language Processing
Voice / Speech Recognition
Automatic Translations

Obama
https://www.youtube.com/watch?v=cQ54GDm1eL0



DeepL
https://www.deepl.com/translator

2 Teams

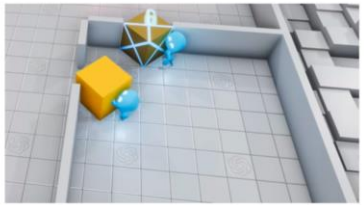
- 2 verstecken sich
- 2 suchen (am Anfang für ein paar Sekunden "eingefroren")

The agents can **move** by setting a force on themselves in the x and y directions as well as rotate along the z-axis.

The agents can **see** objects in their line of sight and within a frontal cone.

The agents can **sense** distance to objects, walls, and other agents around them using a lidar-like sensor.

The agents can **grab and move** objects in front of them.



The agents can **lock** objects in place. Only the team that locked an object can unlock it.

OpenAI | Hide and Seek

https://www.youtube.com/watch?v=Lu56xVlZ40M

start: 00:45min

nvidia
celebrity
challenge

Boston Dynamics | Spot

https://www.youtube.com/watch?v=wIkCQXHEgjA (spot launch)
https://www.youtube.com/watch?v=6Zbhvaac68Y (what can spot do)
https://www.youtube.com/watch?v=kHBcVlqpvZ8 (uptown-dance)

MarI/O

https://www.youtube.com/watch?v=qv6UVOQ0F44

Bachelor Thesis

Problem:
Differenzierung zwischen gutartigen (Lipoma) und bösartigen (ALT) Weichteiltumoren

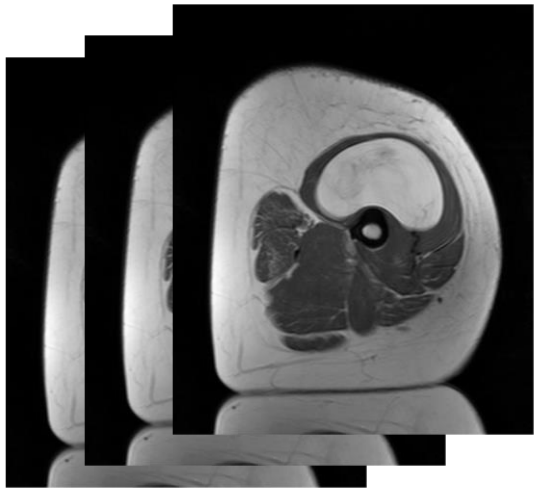
Relevanz:
Selbst für erfahrene Radiologen schwer zu unterscheiden, aufgrund von sehr ähnlichem Aussehen. Die Differenzierung ist jedoch wichtig, da je nach Tumor der operative Ansatz und die Behandlung anders ist

Ziel der Thesis:
Entwicklung eines Deep Learning Ansatzes zur Differenzierung zwischen Lipomas und ALTs

Classification Problem (Binary)

Computer Vision

Daten:
- MRI (Magnet Resonance Imaging) Scans von 108 Patienten
- Information ob der Patient ein ALT (bösartig) oder ein Lipoma (gutartig) hatte



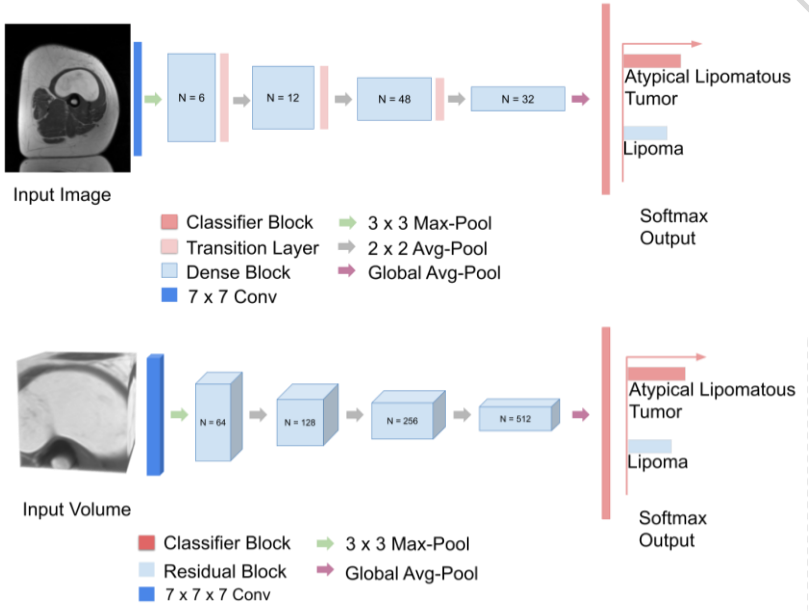
Bachelor Thesis

Input (x)
- MRI Bild (2d)
- MRI Volumen (3d)

Netzwerk $(f_N(x;\theta) = \hat{y})$
- Convolutional Neural Network

Prediction $(\hat{y})$
- Wahrscheinlichkeit Lipoma
- Wahrscheinlichkeit ALT

Target (y)
- 0 - Lipoma
- 1 - ALT



Bestes Model erreichte eine Balanced Accuracy von 85,6 %

Performance deutlich besser als von radiologischen Assistenzärzten und Fachärzten; Vergleichbar mit auf Weichteiltumor spezialisierten Radiologen

Code Demo

04

CIFAR-10 classification mit PyTorch & Wandb

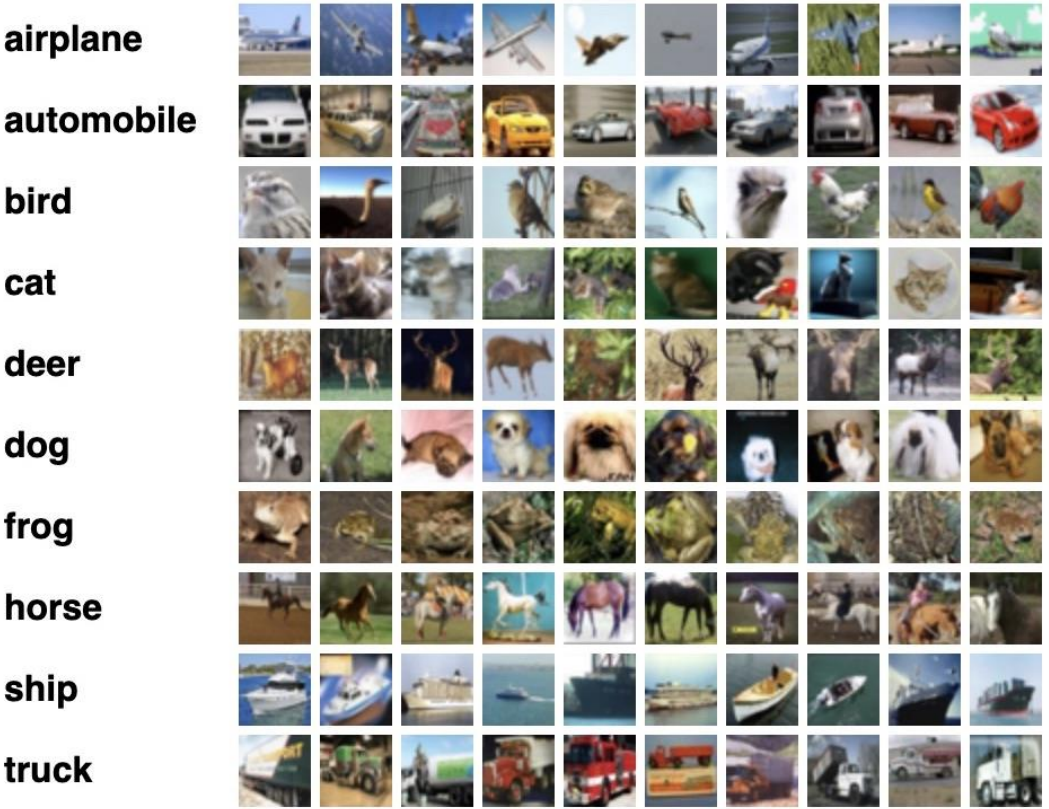
Code Demo - Problem

Dataset:

Das **CIFAR-10** (Canadian Institute for Advanced Research, 10 Klassen) ist ein Dataset welches aus 60000 32x32 farbigen Bildern besteht. Die Bilder sind labeled mit einer von 10 Klassen.

Classification Problem (Multi-label)

Computer Vision



Code Demo - Setup

Netzwerk ($f_N(x;\theta) = \hat{y}$)

```python
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

        self.pool = nn.MaxPool2d(2, 2)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x

net = Net()
```

5 Layer

Activation Function (ReLU)

Hier werden die verschiedenen Arten der Layer definiert (Convolutional Layer, Fully Connected Layer, Pooling Layer)

Hier wird die Reinfolge der Layer definiert, i.e. wie der Input (x) durch das Netzwerk „fließt“ . Diese Methode wird aufgerufen wenn wir eine Prediction machen wollen.

Loss Function $J(\theta)$

```python
criterion = nn.CrossEntropyLoss()
```

Optimizer

```python
optimizer = optim.Adam(net.parameters(), lr=0.001)
```

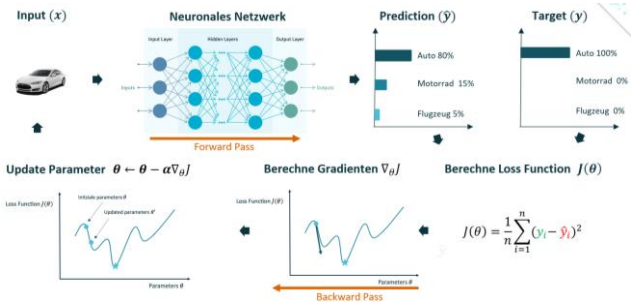
Code Demo – Training Cycle


```python
for i, data in enumerate(trainloader, 0):
    # get the inputs; data is a list of [inputs, labels]
    inputs, labels = data

    # zero the parameter gradients
    optimizer.zero_grad()

    # forward + backward + optimize
    outputs = net(inputs)
    loss = criterion(outputs, labels)
    loss.backward()
    optimizer.step()
```

1. Input (x), Target (y)
2. Forward pass : Create Prediction $(\hat{y})$
3. Berechne Loss Function $J(\theta)$
4. Backward pass: Berechne Gradienten $\nabla_\theta J$
5. Update Parameters: $\theta \leftarrow \theta - \alpha \nabla_\theta J$



Deep Learning

Tac Pot #7 | 2021