# Ubiquitous Computing in Business Processes
# Part IV

Prof. Dr. Lutz Heuser
Urban Software Institute

Darmstadt
WS 2023/ 2024

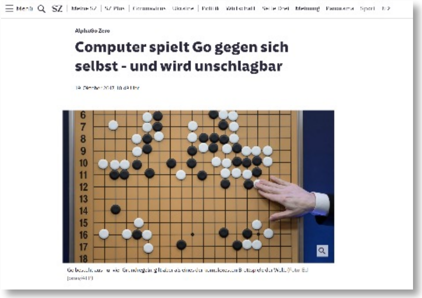
# Outline

1. Analytics

# Outline

# Artificial Intelligence – what is it?

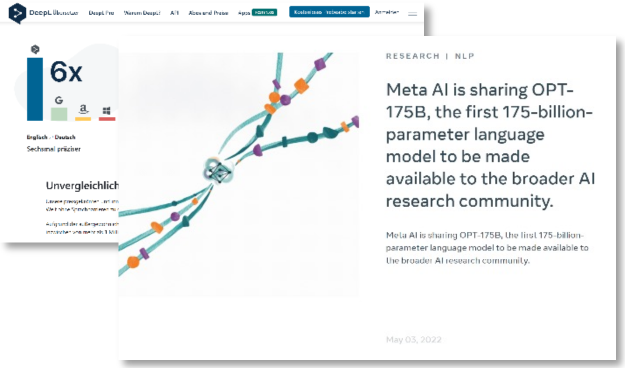
## Games



AlphaGo Zero

**Computer spielt Go gegen sich selbst - und wird unschlagbar**

## Natural Language Processing



Englisch → Deutsch

Sechsmal präziser

Unvergleichlich

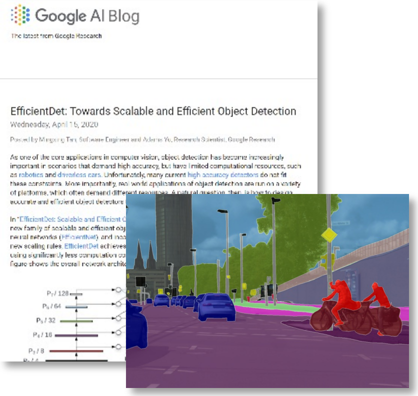
RESEARCH | NLP

Meta AI is sharing OPT-175B, the first 175-billion-parameter language model to be made available to the broader AI research community.

Meta AI is sharing OPT-175B, the first 175-billion-parameter language model to be made available to the broader AI research community.

May 03, 2022

## Computer Vision



Google AI Blog

The latest from Google Research

EfficientDet: Towards Scalable and Efficient Object Detection

Wednesday, April 15, 2020

Posted by Mingxing Tan, Software Engineer and Adams Yu, Research Scientist, Google Research

As one of the core applications in computer vision, object detection has become increasingly important in scenarios that demand high accuracy, but have limited computational resources, such as robotics and driverless cars. Unfortunately, many current high accuracy detectors do not fit these constraints. More importantly, real-world applications of object detection are run on a variety of platforms, which often demand different resources. A natural question, then, is how to design accurate and efficient object detectors

In "EfficientDet: Scalable and Efficient
new family of scalable and efficient obj
neural networks (EfficientNet) and nea
new scaling rules. EfficientDet achieves
using significantly less computation co
figure shows the overall network archite

## Biotechnology

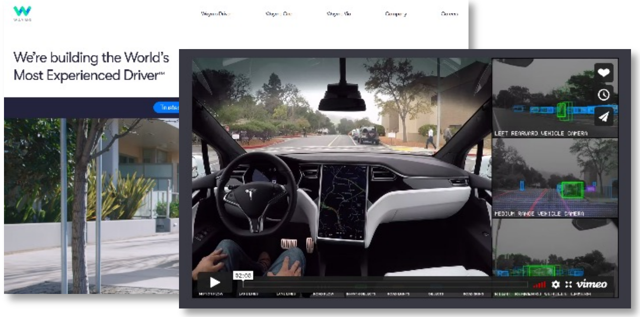
Künstliche Intelligenz

**Turbo für die Biologie**

27. Dezember 2021, 5:00 Uhr | Lesezeit: 5 min

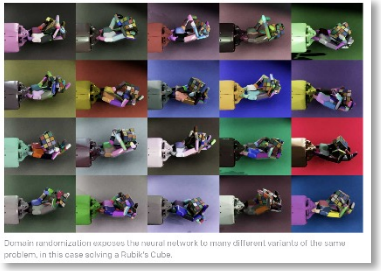
**Die künstliche Intelligenz "Alphafold" ist in der Lage, unbekannte Proteinstrukturen atomgenau vorherzusagen. Warum die Technik einer der wichtigsten Durchbrüche des Jahres ist.**

## Autonomous Driving



We're building the World's Most Experienced Driver™

## Robotics

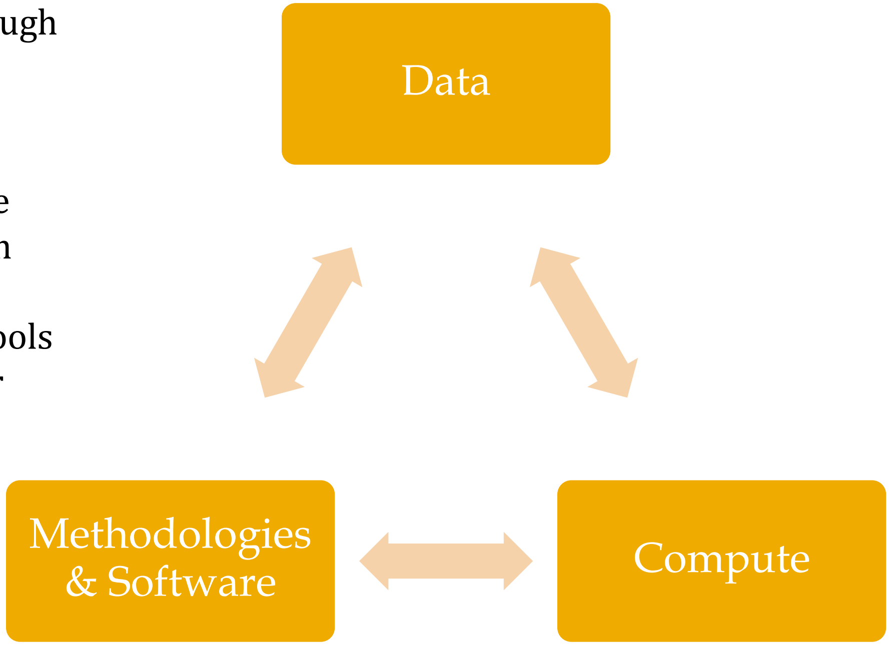


Domain randomization exposes the neural network to many different variants of the same problem, in this case solving a Rubik's Cube.

References:
[2, 3, 4, 5, 6, 7, 8, 9, 55]

# Why these breakthroughs in recent years?

- Structured recording and creation of **large data records**
- (Partly) Freely **available** data records
- Easier **storage and availability** of data through cloud services



- **Improved methods and concepts** for finding patterns in data through (applied) research worldwide
- **Powerful software frameworks** that are freely available (open source).
- **Transferability** of tools and models (transfer learning)

- Comprehensive availability of the necessary **hardware** via the cloud, e.g. GPUs
- **Scalability** of computing capacity
- **Economically attractive** framework conditions thanks to the cloud

References: [12]

# AI and ML

There is no single definition for artificial intelligence. We therefore approach the necessary understanding via the following terms.

## Artificial Intelligence (AI)

**Specialisation**
Can predefined goals be achieved in a given environment?
→ Artificial Narrow Intelligence (ANI)

Status Quo

**Generalisation**
Can poorly defined goals be achieved in an uncertain environment?
→ Artificial General Intelligence (AGI)

## Machine Learning (ML)

„Software 1.0“

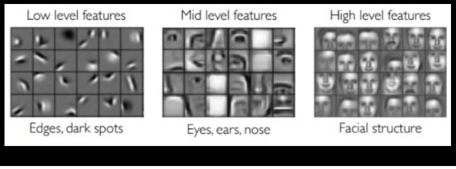
↓

„Software 2.0“

Systems learn to perform tasks independently without being explicitly programmed to do so.

## Deep Learning (DL)

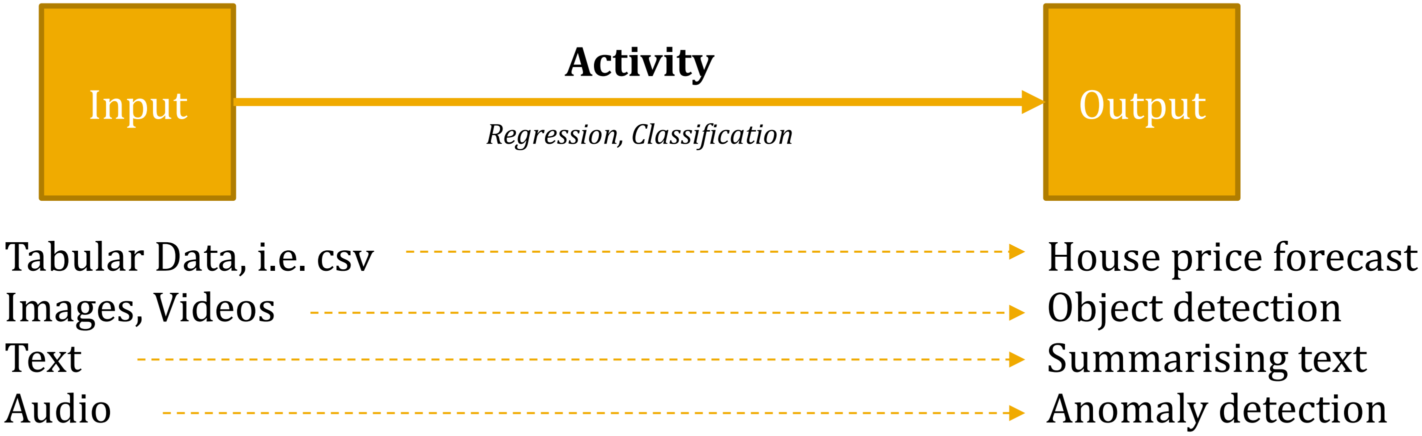
Patterns are extracted from data using special mathematical methods, also known as neural networks. The analysis of these patterns in "new" data is in turn used to complete tasks, e.g. recognising objects in images.



References: [10, 11, 12, 13]

# Machine Learning

„[Machine Learning is the] field of study that gives **computers** the ability to **learn without being explicitly programmed**.“



Input

**Activity**

*Regression, Classification*

Output

| | |
|---|---|
| Tabular Data, i.e. csv | House price forecast |
| Images, Videos | Object detection |
| Text | Summarising text |
| Audio | Anomaly detection |

References: [32]

# Machine Learning Lifecycle

**Legend:**

Initialize

Two phases initialize with ML lifecycle and create context. It always starts here when planning a ML project.

Data

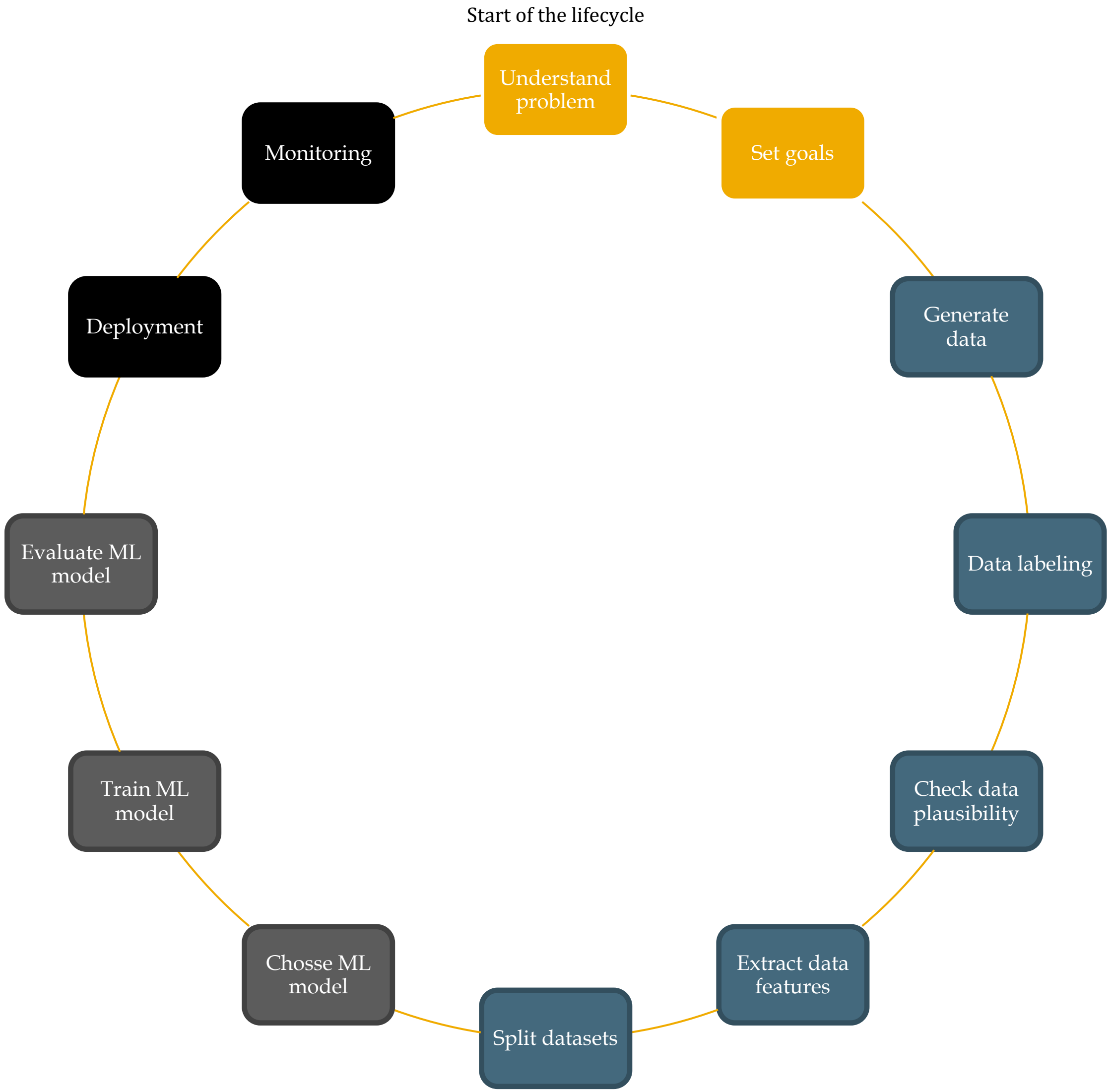
Five phases deal with data pre-processing to prepare any kind of model training & testing. Bad data leads to bad models.

Modeling

Three phases deal with finding the right model & architecture, proper hyperparameters and evaluation metrics to learn how good your model actually is

Deployment

Two phases deal with bringing the model to life so it creates value at customers or consumers. It is about integrating the model to your IT systems and making sure it runs robustly.



References: [17]

# Types of Machine Learning

## Supervised Learning

- Data and labels are available
- The system attempts to learn the relationships (patterns) between data and labels.
- $\{(x1, y1), (x2, y2), \ldots, (xn, yn)\}$ , with $xi$ as input and $yi$ as output. Learn $f: yi = f(xi)$
- Regression & classification
- Example: Prediction of house prices

## Unsupervised Learning

- Data is available, but no labels
- The system attempts to learn relationships (patterns) in data $\{x1, x2, \ldots, yn\}$ without knowledge of $f: yi = f(xi)$.
- Clustering
- Example: Customer segmentation

## Reinforcment Learning

- Simulation-based approach
- An agent undertakes activities in a virtual environment
- Good activities are "rewarded", bad activities are "penalised"
- System improves by learning from "good" & "bad" activities
- Example: Autonomous driving

References: [14, 15, 16, 18, 30]

# Outline

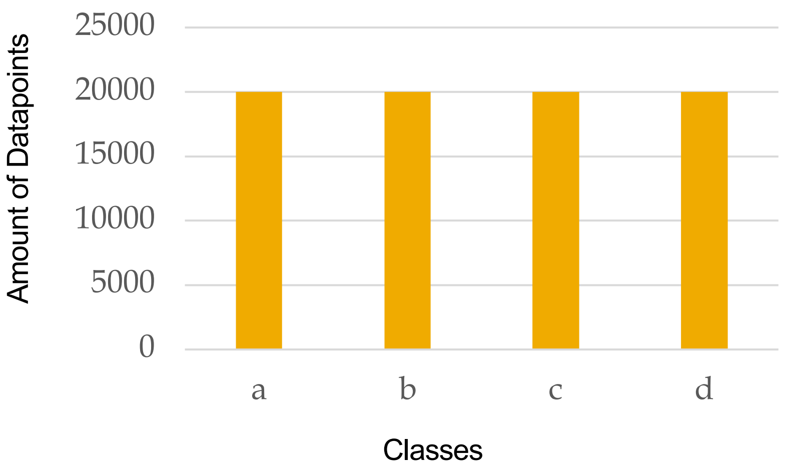
## 1.2 Machine Learning Modeling

# Working with Data: Class Imbalance

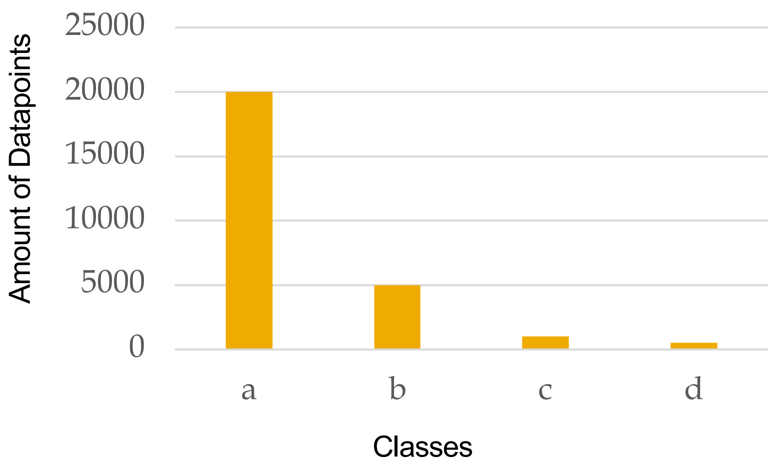
When training ML models, the data should be equally distributed along their classes. This prevents overfitting and promotes the generalisability of the model to new data.

Goal: good performance for predictions with new data.

OK



Not OK



References: [39, 40]

# Working with Data: Class Imbalance

**Problems associated with class imbalance:**

1. Not enough data features to train robust ML models. Models are biased with respect to the overrepresented classes ("bias").

2. Those classes of overrepresented data are predicted more frequently, as the model is weak for underrepresented classes. This means that if the model is uncertain, → often predicts an overrepresented class.

**Reasons for class imbalance:**

1. Incorrectly annotated data points during the labelling process.

2. Imbalance of the data set itself. Care should be taken to ensure that a data set for ML training is evenly distributed with regard to the classes to be predicted.

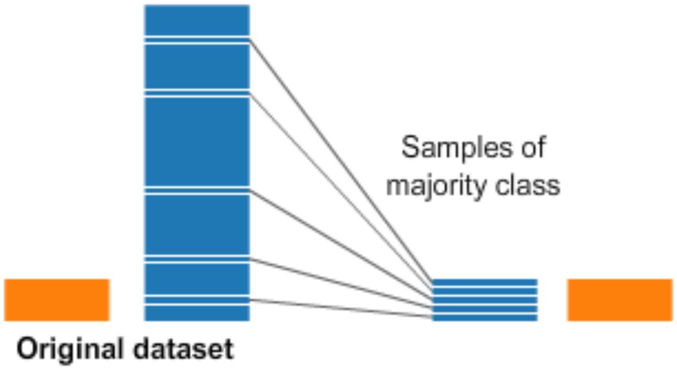
3. Missing data and therefore unbalanced data set.

**Solution: Resampling!**

References: [39, 40]

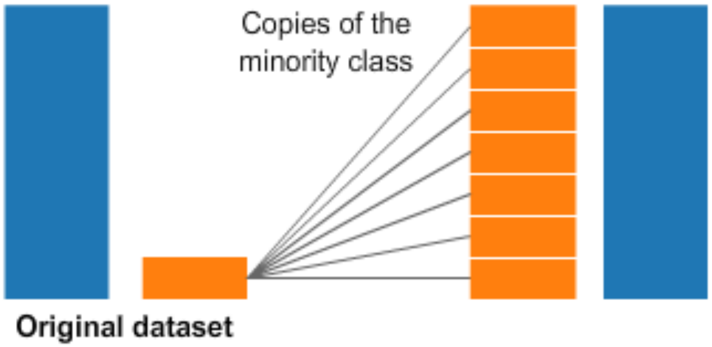
# Working with Data: Resampling

**Undersampling**



**Remove** data points from the overrepresented class

**Oversampling**



**Add data** points to the underrepresented class add data points

References: [39, 41]

# When to consider machine learning?

Machine learning should be considered if the following results arise during the initialisation of an AI project:

- The problem can be solved in a data-driven way.
- The problem cannot be fully solved with a rule-based software solution per se, as there are too many rules (some of which may not yet be known).
- The problem can be solved with a software solution that evolves in a continuously changing environment and automatically adapts to new scenarios.
- The problem should be solved with a software solution that extracts suitable results from large amounts of data (big data).

# When to not consider machine learning?

Machine learning should not be considered if the following results arise during the initialisation of an AI project:

- There are existing, rule-based software solutions that already solve the problem.
- If the process of determining the results of a software solution (e.g. calculations) must be comprehensible.
- If the software solution must have no tolerance for "errors" in order to solve the problem.
- If insufficient data is available or cannot be collected.

# Data-centred vs. model-centred

## Data-centred ML

ML models fixed

Improve data

How can the data for training and testing be customised to improve the performance of the ML system?

## Model-centred ML

Data fixed

Improve ML models

How can ML models be customised in their (1) architecture and (2) hyperparameters so that the performance of the ML system improves?

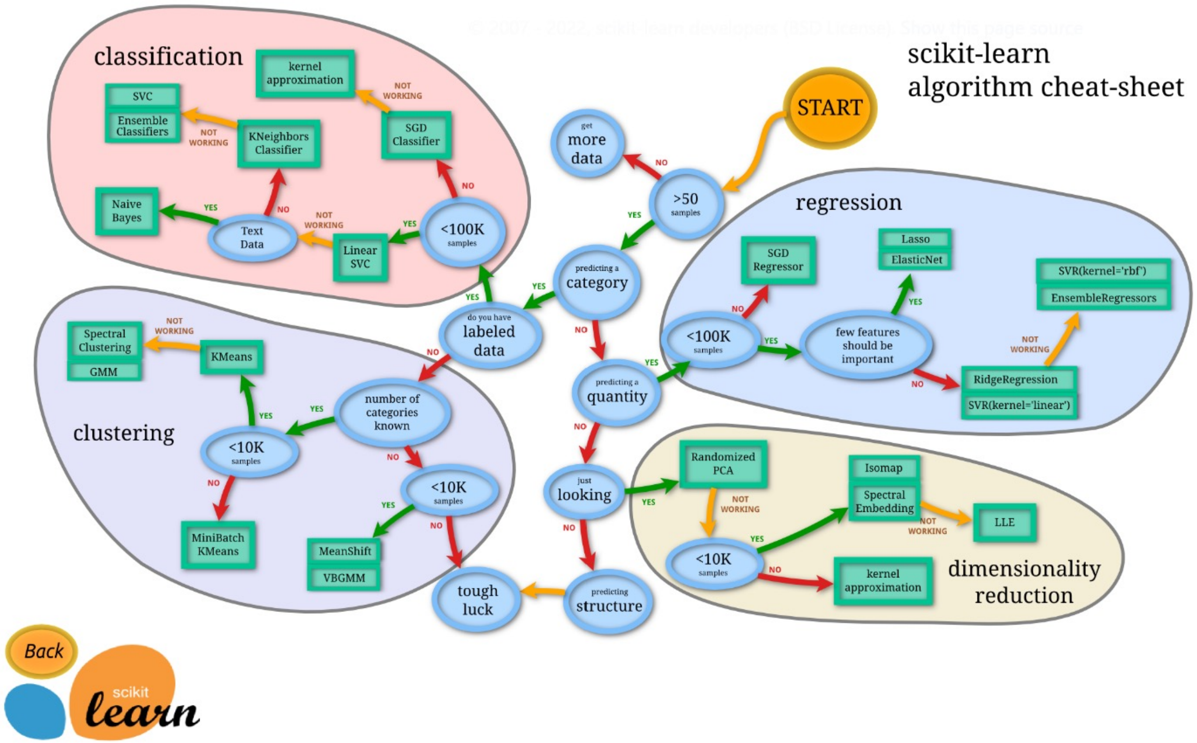
References: [51]

# Selection of suitable algorithms

## Structured Data

The appropriate algorithms are selected depending on how the data is structured.

The "cheat sheet" helps with orientation, but does not replace the final selection of an algorithm by experimenting with the data itself.



**Practical tip:** As a rule, decision tree-based models (random forest, gradient boosting) show good results and their results are easy to understand. This is why they are often used in practice and frequently tried out first.

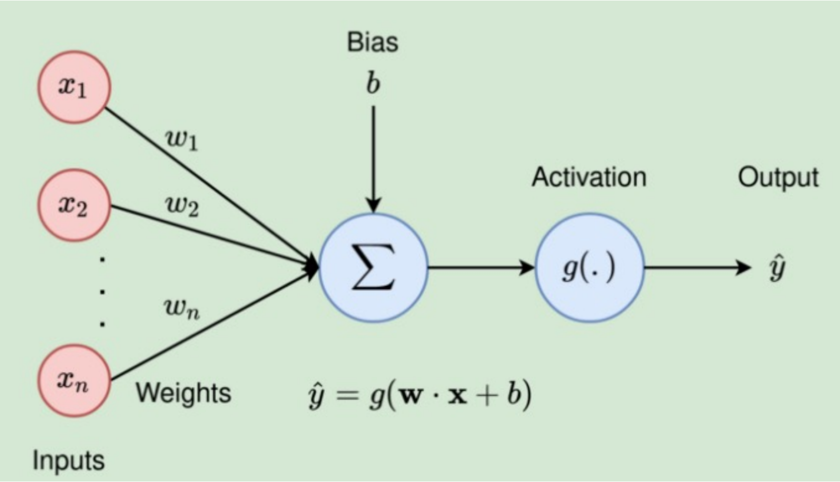
References: [37]

# Neural Networks

For unstructured data, it is advisable to use neural networks. In research and practice, they show by far the best results when processing unstructured data - in comparison to rule-based algorithms.

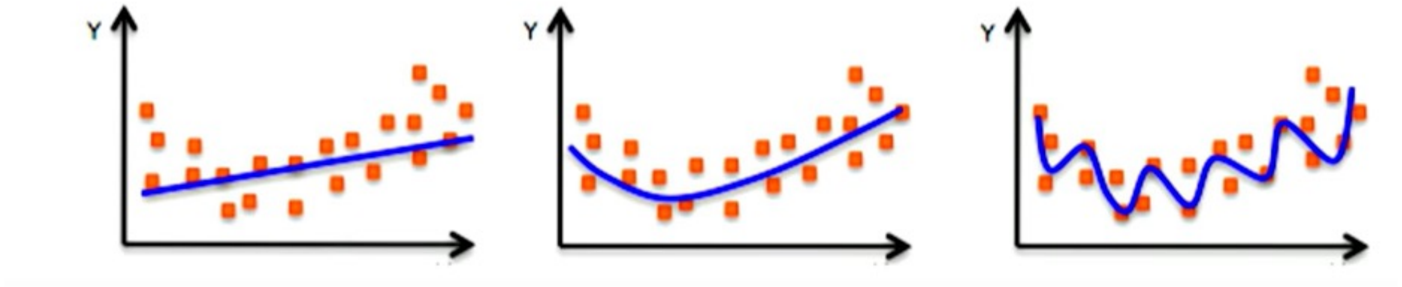
- **Neuron** is the most basic unit of an NN.
- **Input** is the data/values that are passed to the neurons
- **Weights** describe the "strength" of the connection between any two neurons in different layers of the NN. layers of the NN.
- **Bias** is a constant value that is added to the sum of the element-wise multiplication of input values and corresponding weights. Bias is used to strengthen or weaken the activation of a particular neuron.
- **Activation function**: introduces non-linearity into the NN. This feature enables the network to learn complex patterns.
- **Output**: the data/values that are passed on to the next layer of an NN, for example (multi-layer perceptron)

**The perceptron as the basis for NN**



References: [52]

# Learn patterns correctly



**Underfitting**

**OK**

**Overfitting**

ML model does not sufficiently learn the patterns in the data

ML model learns the patterns in the data by heart, but may generalise poorly on new data.

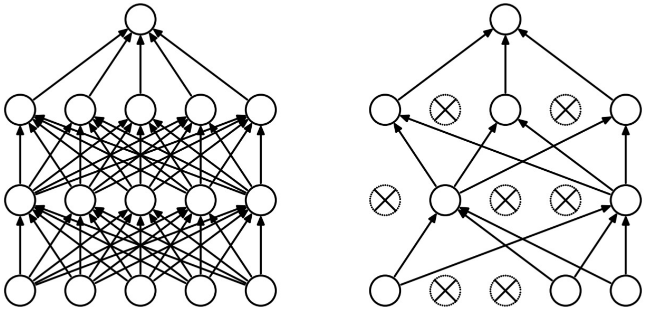
References: [12]

# Regularisation

The aim of an ML model: good generalisability, i.e. ML model shows good performance on "new" data.

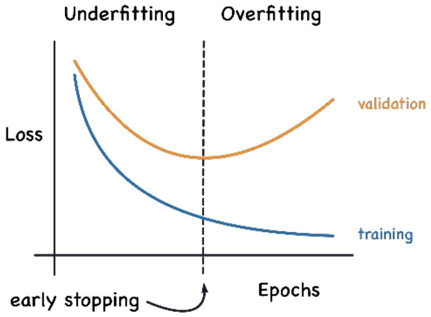
Regularisation helps to ensure that the ML model does not overfit on the training data set, for example. The model is deliberately restricted during training. The following methods are frequently used:

## Dropout



During the training of the NN, a random number of neurons per layer are not activated. **Practical tip**: Usually start with p = 0.5 (50%).

## Early Stopping



Training is stopped as soon as overfitting occurs. The model is then saved with the optimum weights. **Practical tip**: Early stopping is always worth trying out.

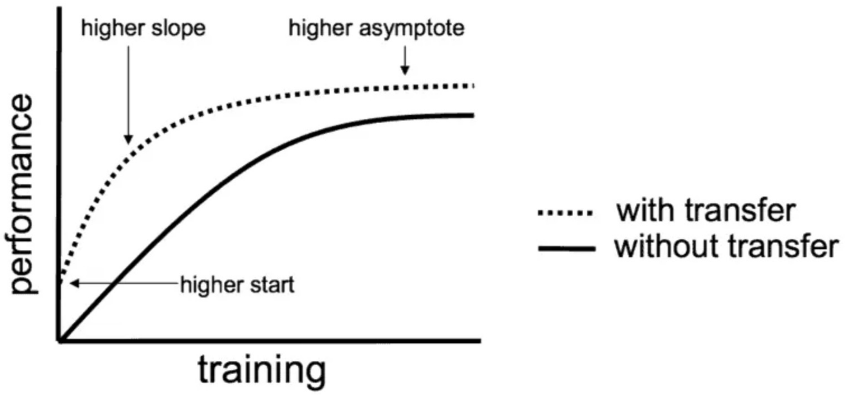
References: [12, 70, 71]

# Transfer Learning

Transfer learning is a machine learning method in which a pre-trained ML model is used as the starting point for a deep learning project.

Areas of application are, for example, image classification and object detection in the field of computer vision, but they can also originate from the fields of NLP or audio.

- Reuse the weights of a model trained on a data set and for a task, e.g. ImageNet for computer vision
- Time saving
- Saves resources
- Usually higher performance than when starting with random weights.

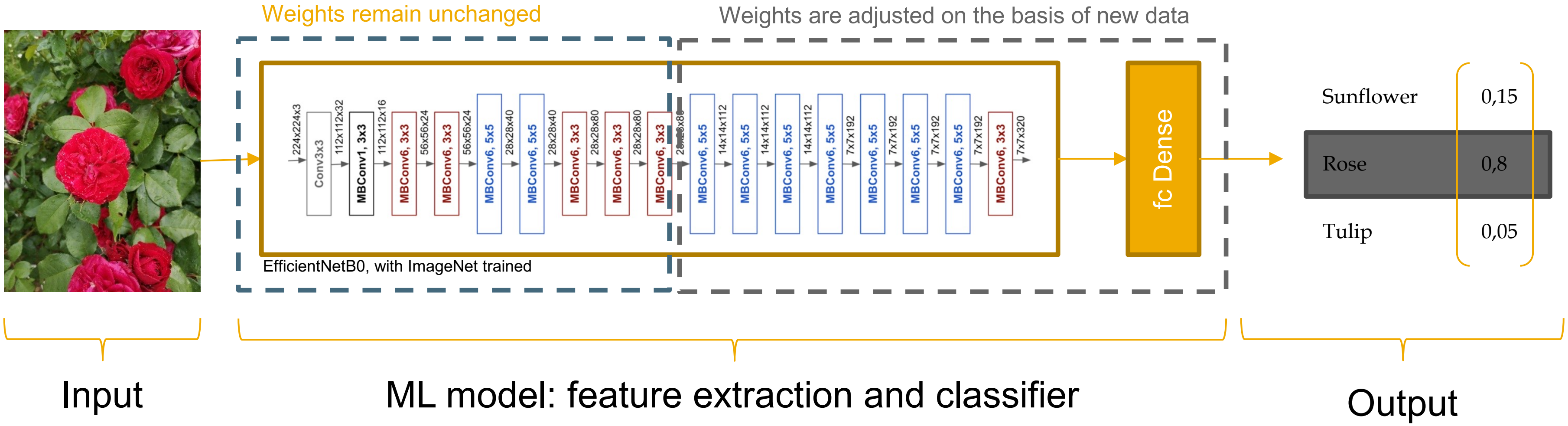


**Practical tip**: use transfer learning wherever possible. It is very useful for images, audio and text. Research pre-trained models.

References: [65]

# Why Transfer Learning?

Transfer learning therefore reuses existing NNs for my task and adapts them to my data set and my classes. We therefore take advantage of the fact that an NN has already learnt low-level and mid-level features with another (large) data set and simply sensitise it to our data set.



References: [66, 67]

# Improve ML Models

How can we proceed to improve our model in addition to transfer learning - if we are not yet satisfied with the performance to date?

- More (balanced) data
- Add more layers to the NN
- Add more neurons per layer
- Change the activation function
- Change the optimiser
- Train for more epochs
- Change the learning rate

Change hyperparameters! Carry out experiments!

But what criteria should I use to adapt my hyperparameters and in which direction? → Random Search, Grid Search!

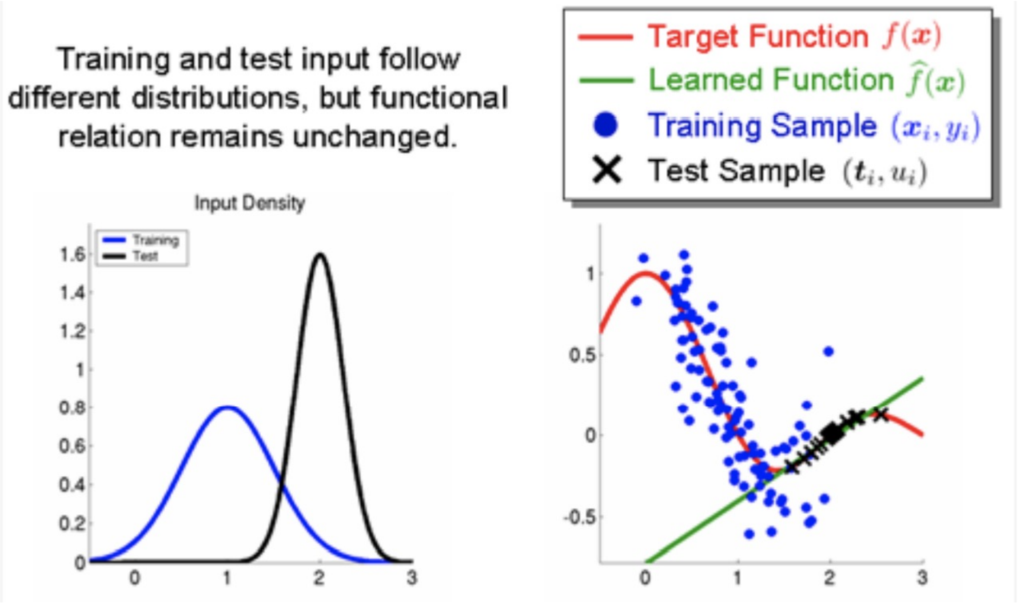
# Improve ML Models

| Type | Description | Framework |
|---|---|---|
| Random Search | Set up a grid of hyperparameters and select random combinations to train the model and evaluate it using the validation data. The number of combinations to be tried can be limited. | sklearn.model_selection.RandomizedSearchCV |
| Grid Search | Set up a grid of hyperparameters and train a model for each possible combination and evaluate it using the validation data. With this approach, every single combination of hyperparameters is tried out, which can be very resource-intensive (time, computing power)!<br>Grid Search takes longer than Random Search with the same use of resources - usually also leads to a better performance of the model. | sklearn.model_selection.GridSearchCV |

**Result**: List of hyperparameter configurations optimised in terms of evaluation metrics, including trained model.

**Practical tip**: first start with Random Search and assess whether performance is suitable. If not, consider Grid Search.

# Data Drift


Training and test input follow different distributions, but functional relation remains unchanged.

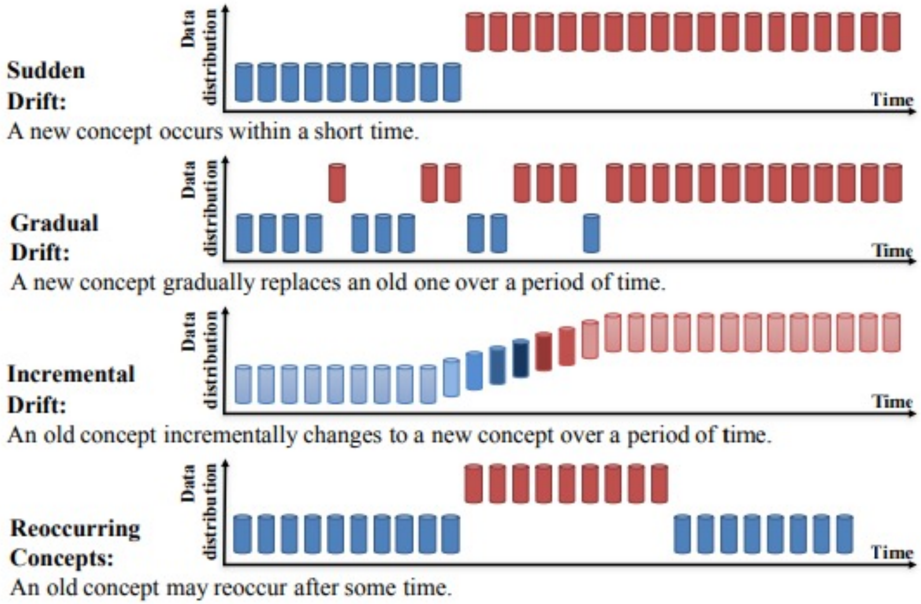
**Problem**:
Distribution of input data during operation (or test data) ≠ Distribution of training data

**Difficulty**:
It is difficult to estimate in advance whether the input data for a model will have the same distribution during operation as during training.

**Reasons**:
Bias in the training data

Quellen: [75, 76]

# Concept Drift


Sudden Drift:
A new concept occurs within a short time.
Gradual Drift:
A new concept gradually replaces an old one over a period of time.
Incremental Drift:
An old concept incrementally changes to a new concept over a period of time.
Reoccurring Concepts:
An old concept may reoccur after some time.
Data distribution
Time

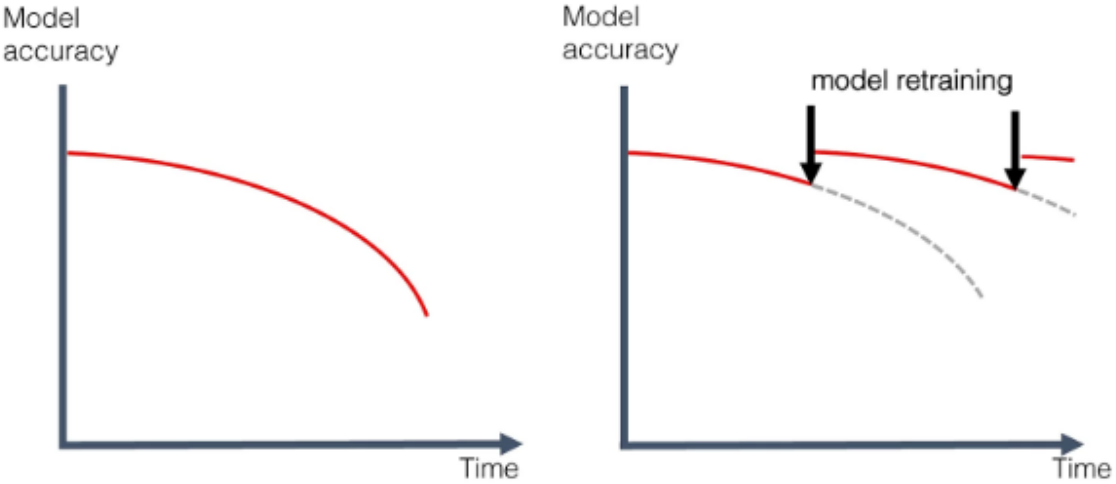
**Problem**:
Distribution of input data remains the same during operation (or test data), but output changes unpredictably over time.

**Difficulty**:
Is a gradual process. Predictions therefore become less accurate over time. Early recognition is important.

**Reasons**:
Bias in the training data, e.g. traffic before and during the COVID-19 pandemic

References: [45, 76]

# How to deal with Data Drift & Concept Drift

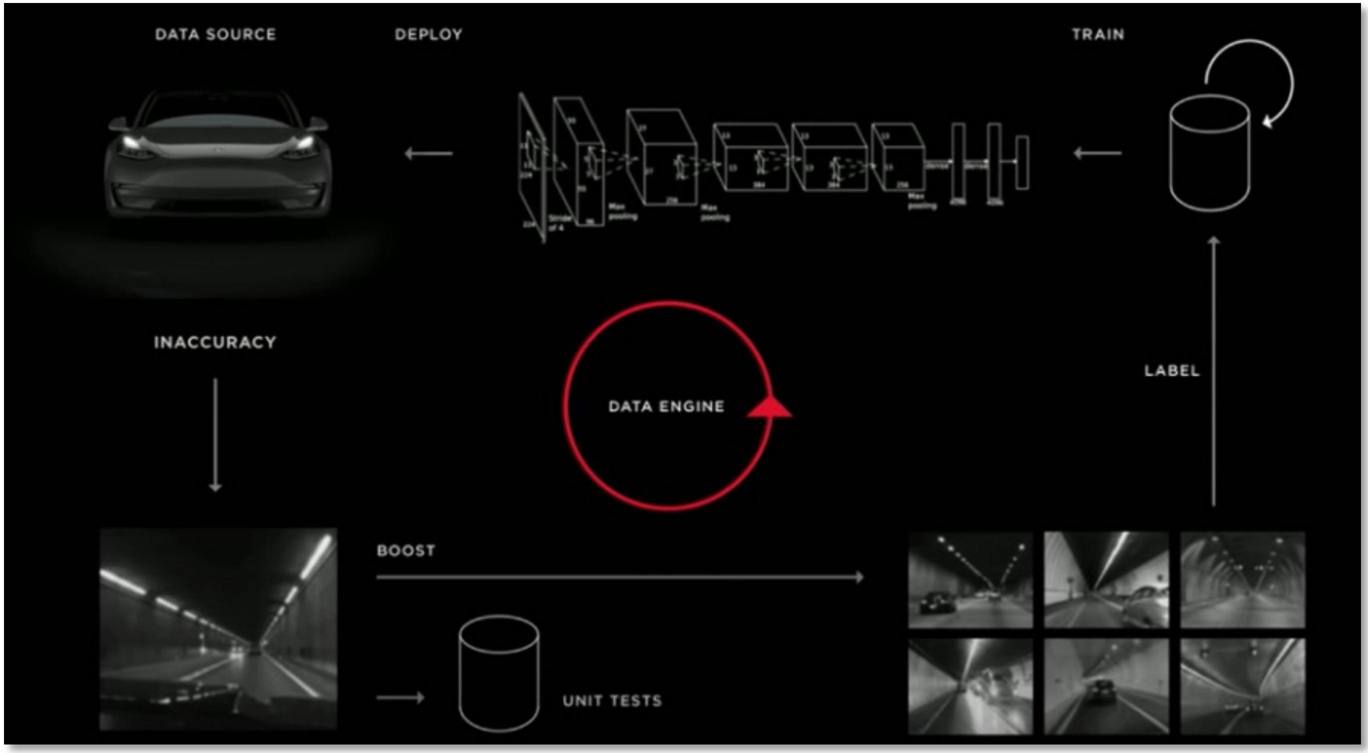


**Continuous re-training** of ML models! ML models and their performance must be continuously monitored during operation. If there are signs of data drift or concept drift, models must be trained with the "new" data and put back into operation.

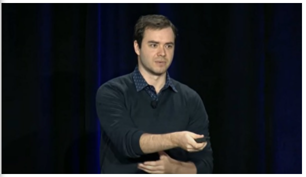
References: [44]

# Continuous improvement process

## „Data Flywheel“



A. Karpathy (Formerly Tesla)



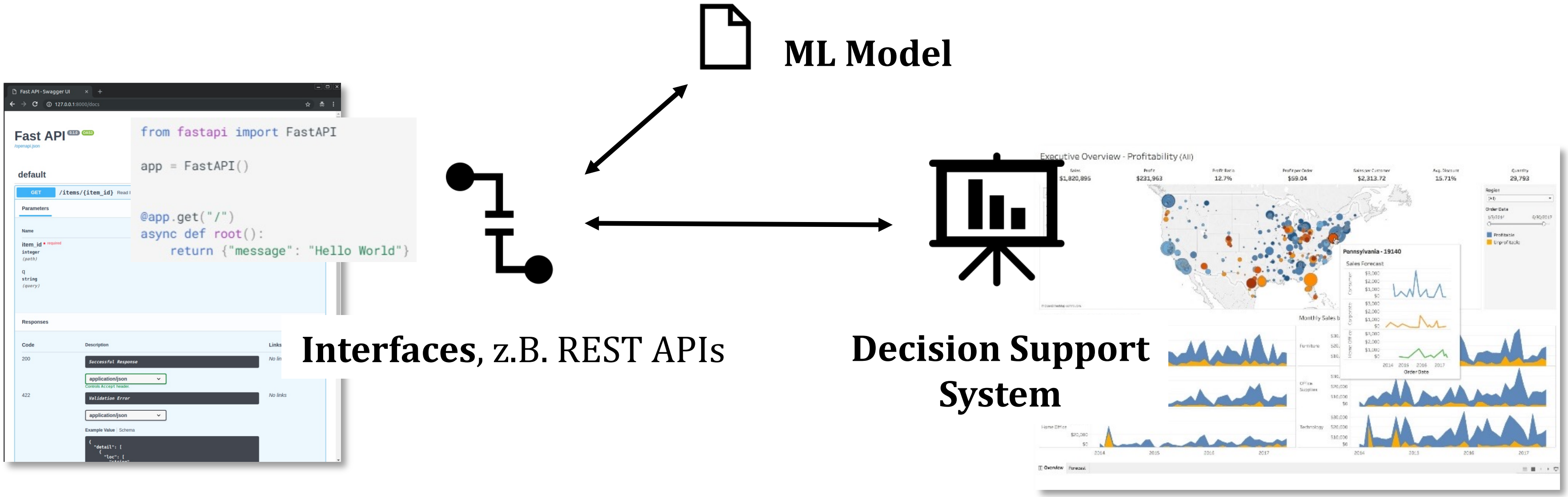
References: [47]

# Outline

## 1.3 Deployment of ML Models

# Deployment of ML Models

Deployment of ML models $\xrightarrow{\text{means}}$ Integration into processes and IT systems



```
from fastapi import FastAPI

app = FastAPI()


@app.get("/")
async def root():
    return {"message": "Hello World"}
```

ML Model

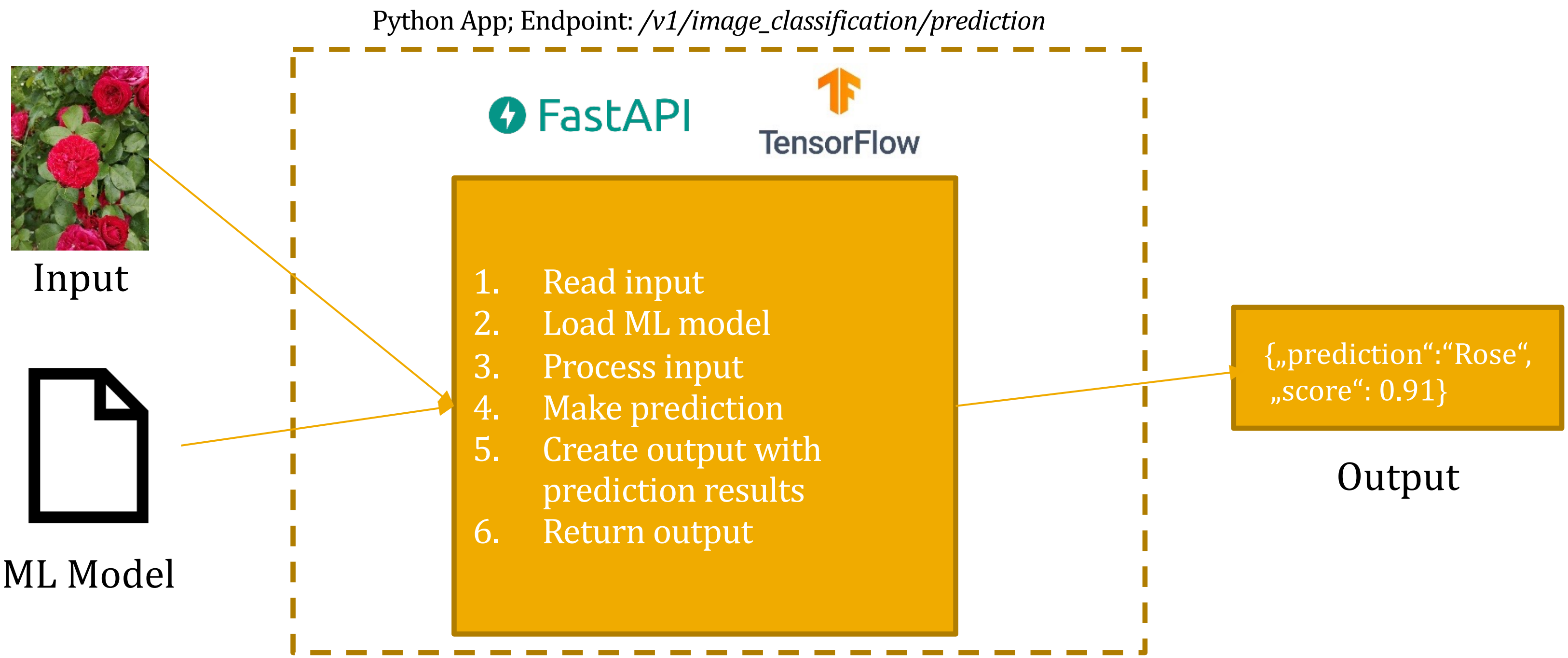
**Interfaces**, z.B. REST APIs

**Decision Support System**

References: [42, 43]

# Deployment of ML Models

The structure of a data interface for the integration of ML models into an existing system architecture can look as follows:



Python App; Endpoint: */v1/image_classification/prediction*

1. Read input
2. Load ML model
3. Process input
4. Make prediction
5. Create output with prediction results
6. Return output

{„prediction“:“Rose“, „score“: 0.91}

# Outline

## 1.4 Use Case Example

# STGCN for Traffic Insights

Spatio-Temporal Graph Convolutional Networks (STGCN) model relationships and **interactions over both space and time within graph-structured data**. They effectively capture dynamic patterns by applying convolutional operations across the spatial components of a graph (e.g., nodes and their connections) and temporal sequences (e.g., changes or movements over time).

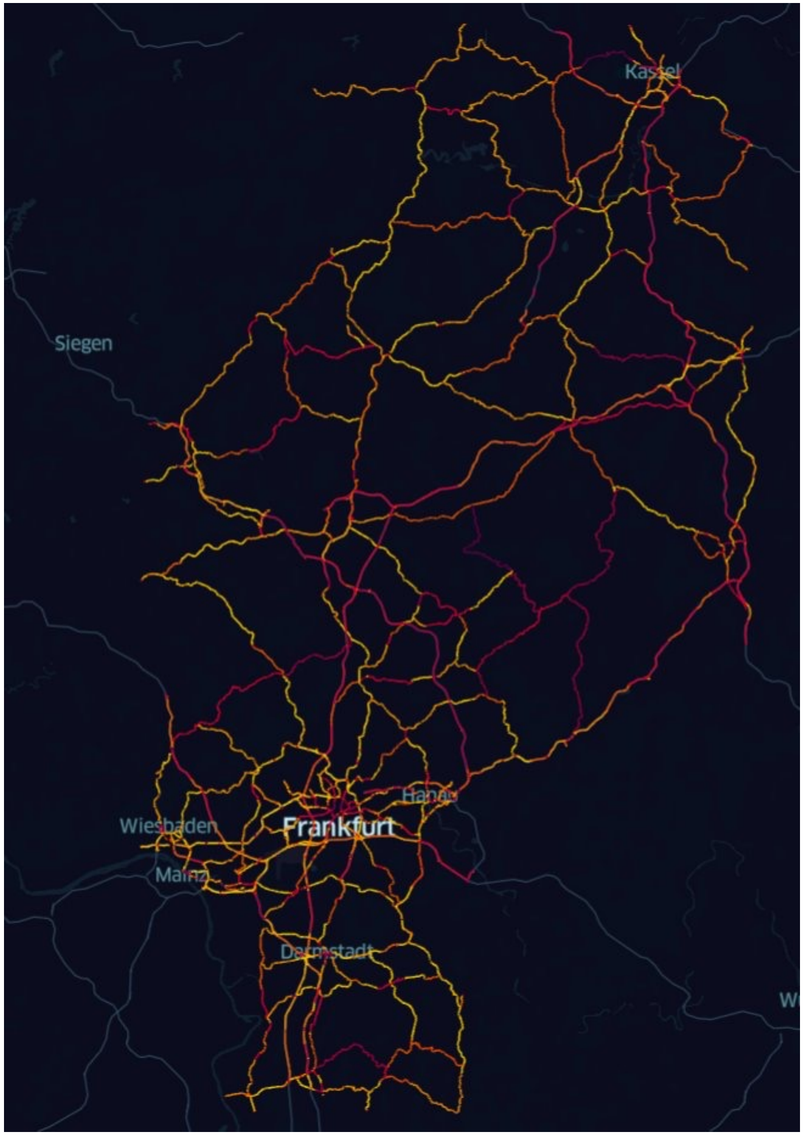


Figure right: Travel time prediction with STGCN, based on Floating Car Data and OpenStreetMap

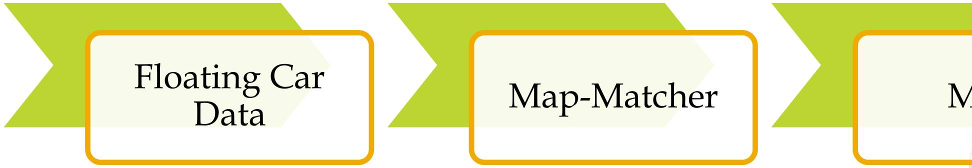
**Legend**: red = high travel times, orange = medium travel times, yellow = low travel times.
All compared to free-flow travel time which is 85th percentile of given travel time in a time intervall and street segment.

# STGCN for Traffic Insights

Why STGCNs?

- Often single model per segment, hence difficult to manage model in case of a large network. Imagine a street network!
- Information from adjacent segments is ignored with model-per-segment approach.

How it works?



Floating Car Data

Map-Matcher

Model

- Latitude
- Longitude
- Timestamp
- Speed
- Heading

Match FCD points on streets.

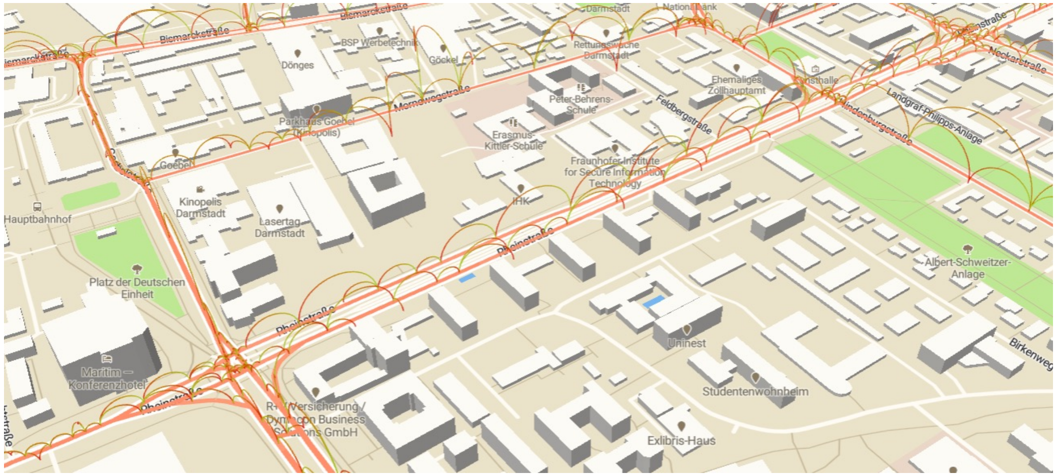
Predictions for each segment of the graph at timestamp t

Road network OSM tertiary and above segments in Darmstadt

# STGCN for Traffic Insights

Represent road network as graph:

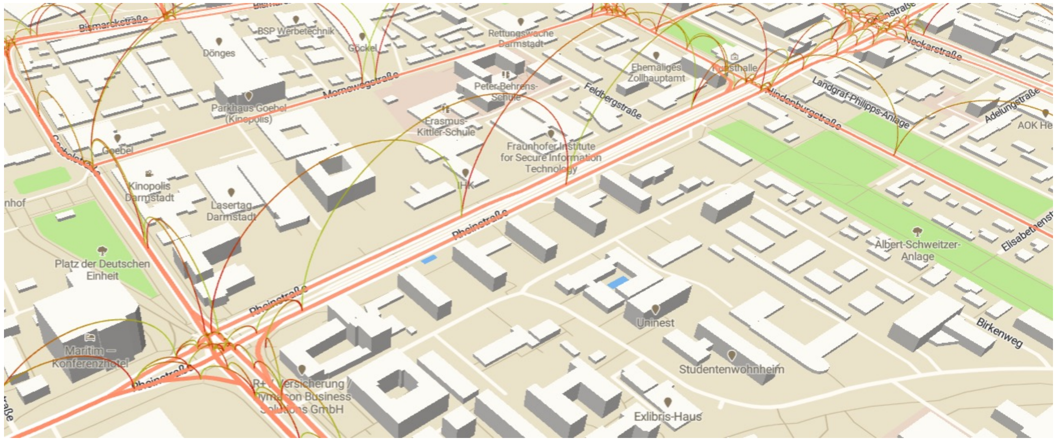
- Road segments as nodes
- Connection between segments as edges
- Connectivity strength between road segment as the edge weight



Transitions represented with arcs on Rheinstr. before graph reduction

Collapse graph:

- Reduces computational complexity
- Allows to model bigger areas
- Remove redundant information



Transitions represented with arcs on Rheinstr. after graph reduction

# References

For further details about references, please check the attached Excel file published in Moodle.