Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

inovex

AI Labor - Sommersemester 2020

Corona-Edition
– und –
1. Termin Computer Vision

Robin Baumann, Matthias Richter

Karlsruhe, 20. März 2019

Agenda für Heute

1. C O R O N A
2. Computer Vision
3. Tensorflow
4. MLflow
5. Overfitting
6. Datensätze
7. colab

CORONA

Pandemie-Modus

- In zwei Wochen: Sprintwechsel (Remote)
  - Review: Lösungen präsentieren
  - Retro: Was lief gut? Was lief schlecht?
  - Planning: Neue Aufgaben
- In einer Woche: Remote-Betreuung
  - Timeslots für jede Gruppe
- Dazwischen: Slack-Workspace

Pandemie-Modus

- Pool geschlossen → eigene Hardware / Cloud
- eigene Hardware
  - Docker-Setup im Git-Repository: [github.com/inovex/ai-lab]
  - Macht ohne GPU keinen Spaß
- Cloud: [Google Colab]
  - Kostenlose GPUs und TPUs!
  - Benötigt Google-Account (und Internet)

Gruppenfindung

1. Zu 3er Teams und ein 4er Team zusammenfinden
   - 16 Teilnehmer → 5 Teams

2. Team-Name überlegen

3. Mitglieder und Team-Name mitteilen (slack)

Lean-Coffee

Pandemie-Modus

1. Themen sammeln

2. Stimmen auf Themen verteilen (3 Stimmen / Person)

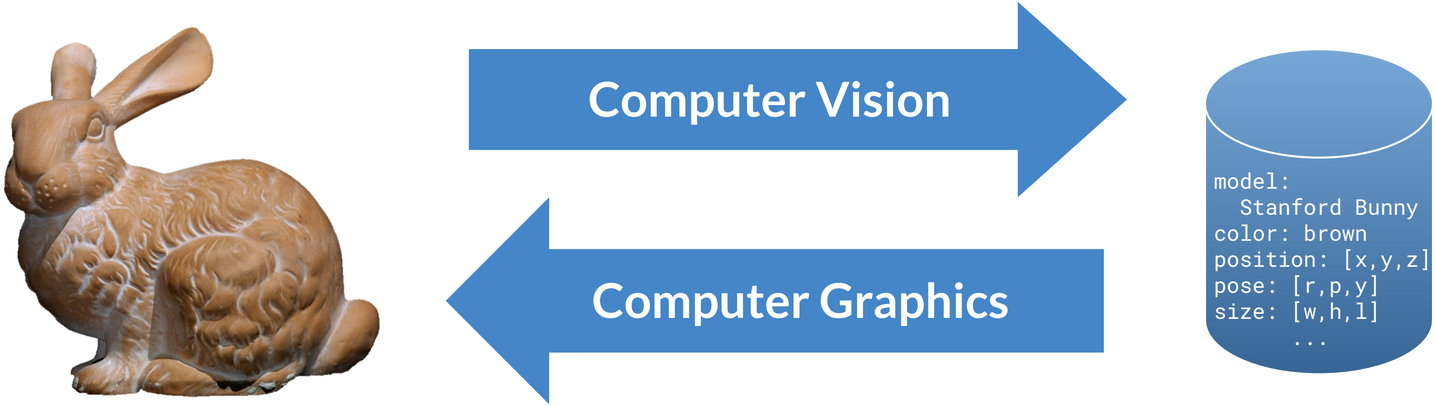
3. Ca. 30 min Diskussion

Computer Vision

"Computer Vision is an research area that studies how to **make computers efficiently** **perceive, process, and understand** **visual data** [...]. The ultimate goal is for computers to **emulate** the striking perceptual capability of **human eyes and brains,** or even to **surpass and assist the human** in certain ways".

– Definition von Microsoft Research

Computer Vision = ɘɔiʜqɒɿ⅁ ɿɘɈuqmoϽ



Bild

Wissen

Bildquelle Stanford Bunny: http://graphics.stanford.edu/data/3Dscanrep

Computer Vision ist schwierig

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

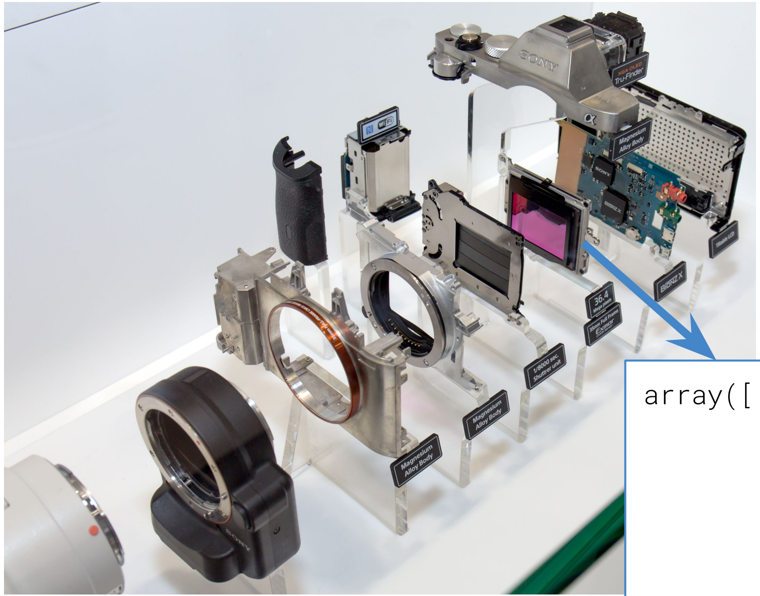
Artificial Intelligence Group July 7, 1966
Vision Memo. No. 100.

THE SUMMER VISION PROJECT

Seymour Papert

The summer vision project is an attempt to use our summer workers effectively in the construction of a significant part of a visual system.

Warum ist Computer Vision schwierig?



```
array([[...,
        [0.83137256, 0.7176471 , 0.6627451 , 1.],
        [0.8235294 , 0.70980394, 0.654902  , 1.],
        [0.8117647 , 0.7019608 , 0.64705884, 1.],
        ...
       ],
       [[0.8392157 , 0.72156864, 0.6784314 , 1.],
        [0.8156863 , 0.7058824 , 0.65882355, 1.],
        [0.8117647 , 0.7019608 , 0.654902  , 1.],
        ...
       ],
       ...
      ], dtype=float32)
```

„Tensor“

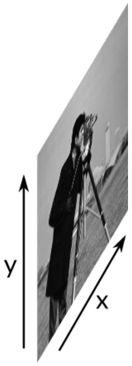
 Bildquelle: https://de.wikipedia.org/wiki/Datei:Sony_Alpha_ILCE-7R_taken_apart_2014_CP%2B.jpg (Nutzer: Morito)

inovex

Tensoren

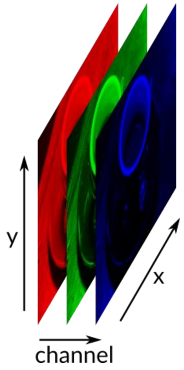
- Verallgemeinerung von Vektoren und Matrizen
  - Tensor von Rang 0 → Skalar
  - Tensor von Rang 1 → Vektor / Kovektor
  - Tensor von Rang 2 → Matrix
  - Tensor von Rang 3 → ???
- Praktisch: mehrdimensionaler Array
  - Rang = # Dimensionen
  - `rank_1_tensor = np.array([1, 2, 3])`

Bilddaten als Tensoren



Graubild → 2D Array

```
img.shape = (h, w)
gray = img[y, x]
```

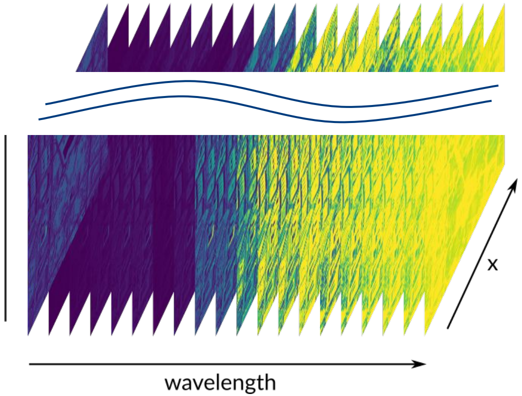


Farbbild → 3D Array

```
img.shape = (h, w, c)
r = img[y, x, 0]
g = img[y, x, 1]
b = img[y, x, 2]
```

auch:

```
img.shape = (c, h, w)
```



Spektral → 3D Array

```
img.shape = (h, w, λ)
wl_0 = img[y, x, 0]
wl_1 = img[y, x, 1]
wl_2 = img[y, x, 2]
...
```

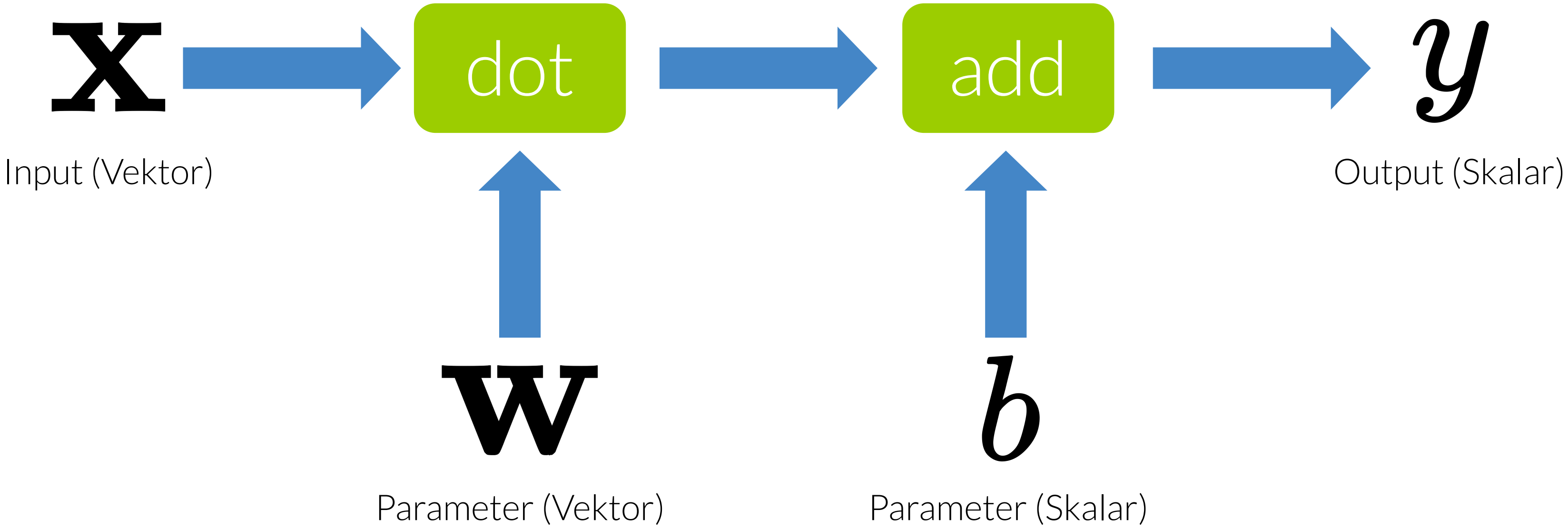
TensorFlow

TensorFlow

- Populäres Framework für Machine Learning
  - Vor allem (aber nicht nur) Deep Learning
- Kern: Graph zur Beschreibung von Datenflüssen
  - Knoten = Transformation (add, sum, mult, …)
  - Kanten = Tensoren
- Berechnung transparent auf GPU
- Hier Fokus auf high-level Keras API

Ein einfacher Graph:

$y = \mathbf{w}^\top \mathbf{x} + b$




```
def y(x):
  return tf.tensordot(w, x, 1) + b
```

Hello, TensorFlow

```
from tensorflow.keras import Sequential, layers
from tensorflow.keras.losses import CategoricalCrossentropy

model = Sequential([
    layers.Dense(64, activation='relu', input_shape=32),
    layers.Dense(10)])

model.compile(optimizer=tf.keras.optimizers.Adam(0.01),
              loss=CategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

model.fit(x=training_data, y=labels, epochs=10, batch_size=32)
predictions = model.predict(data)
```


Mehr hier: https://www.tensorflow.org/guide/keras/overview

1. Modell-Definition

```
model = Sequential([
    layers.Dense(64, activation='relu', input_shape=32)])
model.add(layers.Dense(10))
```

- Sequential-API: Layer als Liste
  - Erster Layer benötigt `input_shape`
  - Letzter Layer definiert Modell-output
- Mitgelieferte Layer:
  - Core: Dense, Dropout, Flatten, Reshape, ...
  - Convolutional: Conv1D, Conv2D, …
  - usw.

Siehe tensorflow.org/api_docs/python/tf/keras/Sequential und tensorflow.org/api_docs/python/tf/keras/layers

2. Modell-Ziel

```
model.compile(optimizer=tf.keras.optimizers.Adam(0.01),
              loss=CategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

- Vorbereitung für das Training
  - Wie optimieren? → `optimizer`
  - Was optimieren? → `loss` (und `loss_weights`)
  - Wie überwachen? → `metrics` (und `weighted_metics`)
- Typische Losses
  - `CategoricalCrossentropy` → Klassifikation
  - `MeanSquaredError` → Regression

Siehe tensorflow.org/api_docs/python/tf/keras/Sequential

3. Modell-Training

```
model.fit(x=data, y=labels, epochs=10, batch_size=32, callbacks…
model.fit(generator, epochs=10, batch_size=32, callbacks=[...])
```

- Trainingsdaten: Samples + Labels oder Generator
  - Mehrerer Klassen → One-hot-encoding:

    [1, 3, 2] → [[1, 0, 0], [0, 0, 1], [0, 1, 0]]

    `utils.to_categorical(class_indices, num_classes)`
- Trainingsplan:
  - `epochs, batch_size`
  - `callbacks` (zu Beginn/Ende von Batch/Epoche)

Siehe tensorflow.org/api_docs/python/tf/keras/Sequential#fit

4. Modell-Auswertung

```
predictions = model.predict(data)
test_loss = model.evaluate(data, true_labels)
```

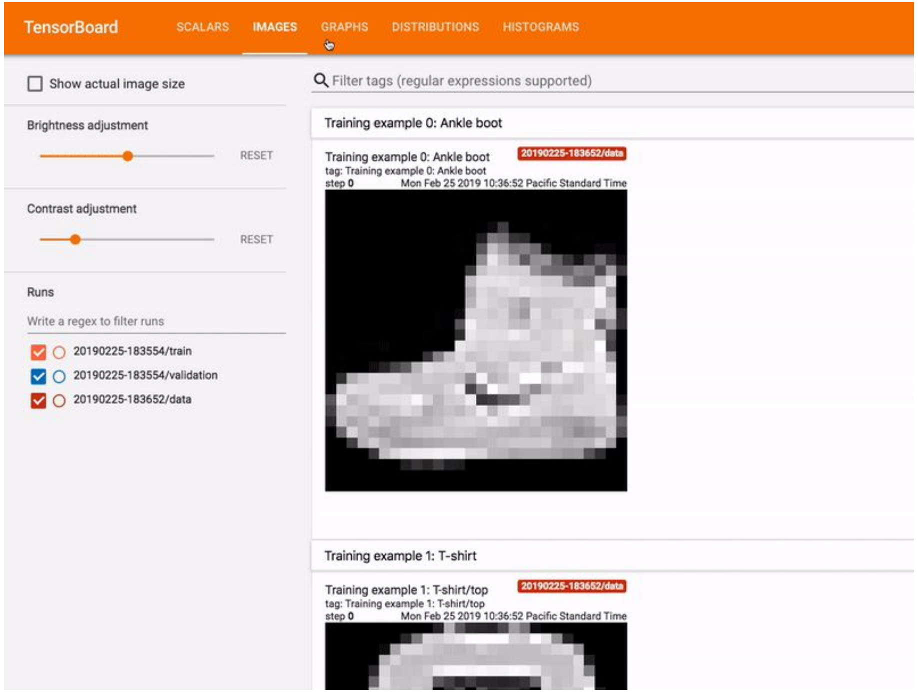
- Wertet das Modell aus (forward-pass)
- `predictions.shape = (data.shape[0],`
  `<shape-of-last-layer>)`
- `test_loss.shape = (len(metrics),)`

 Siehe tensorflow.org/api_docs/python/tf/keras/Sequential#predict

TensorBoard

Visualisierung und Tooling für ML-Experimente

- Metriken
- Modellgraphen und -parameter
- Trainingsdaten und (Zwischen-)Ergebnisse
- Embeddings
- Profiles
- …



Bildquelle: tensorflow.org/tensorboard

MLFlow

Hauptfunktion

- Experimente tracken und vergleichen
- ML-Code verpacken
- Modelle deployen

→ mlflow.org/docs/latest ←

Hello, MLflow

› Experimente tracken

```python
from mlflow import log_metric, log_param, log_artifact

log_param('batch_size', 32)
log_param('the_answer', 42)
log_metric('accuracy', test_metrics[0])
log_artifact(path_to_the_model)
```

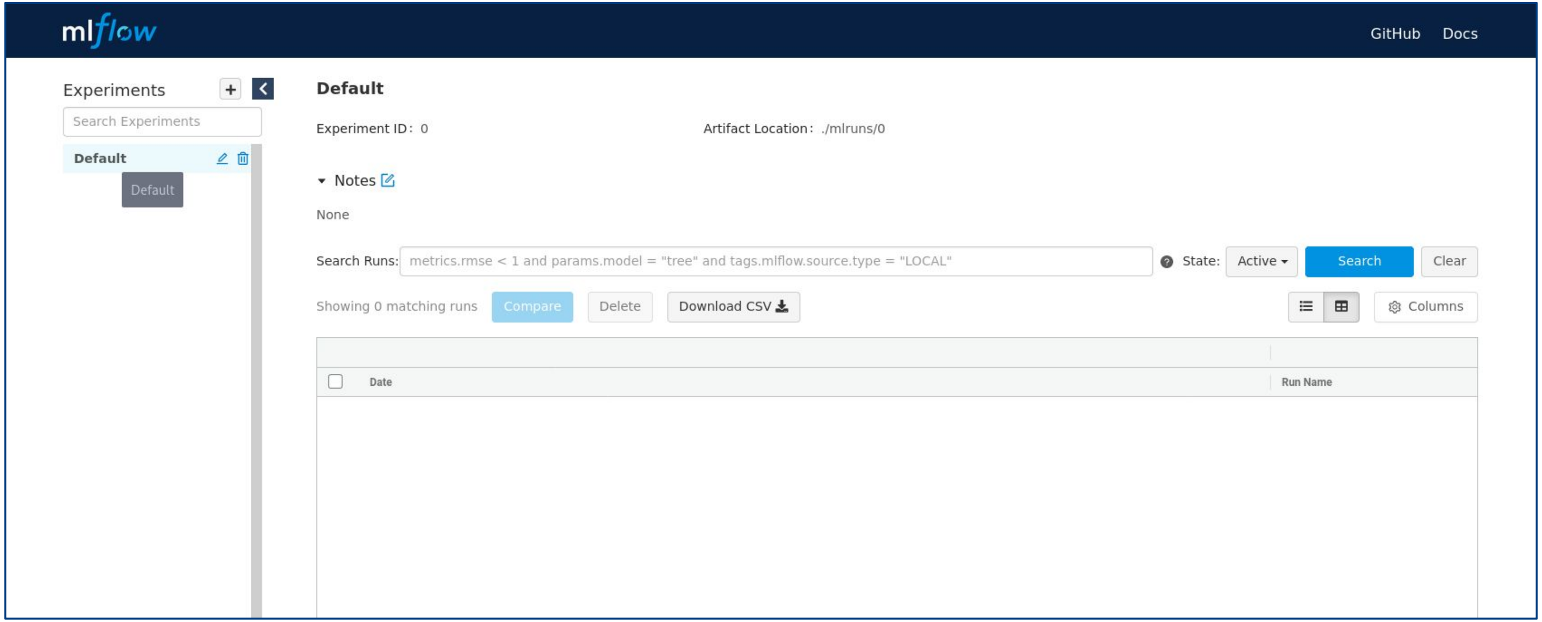
› Automatisches Logging

```python
mlflow.tensorflow.autolog(every_n_iter=23)
```

Hello, MLflow

› ... und vergleichen

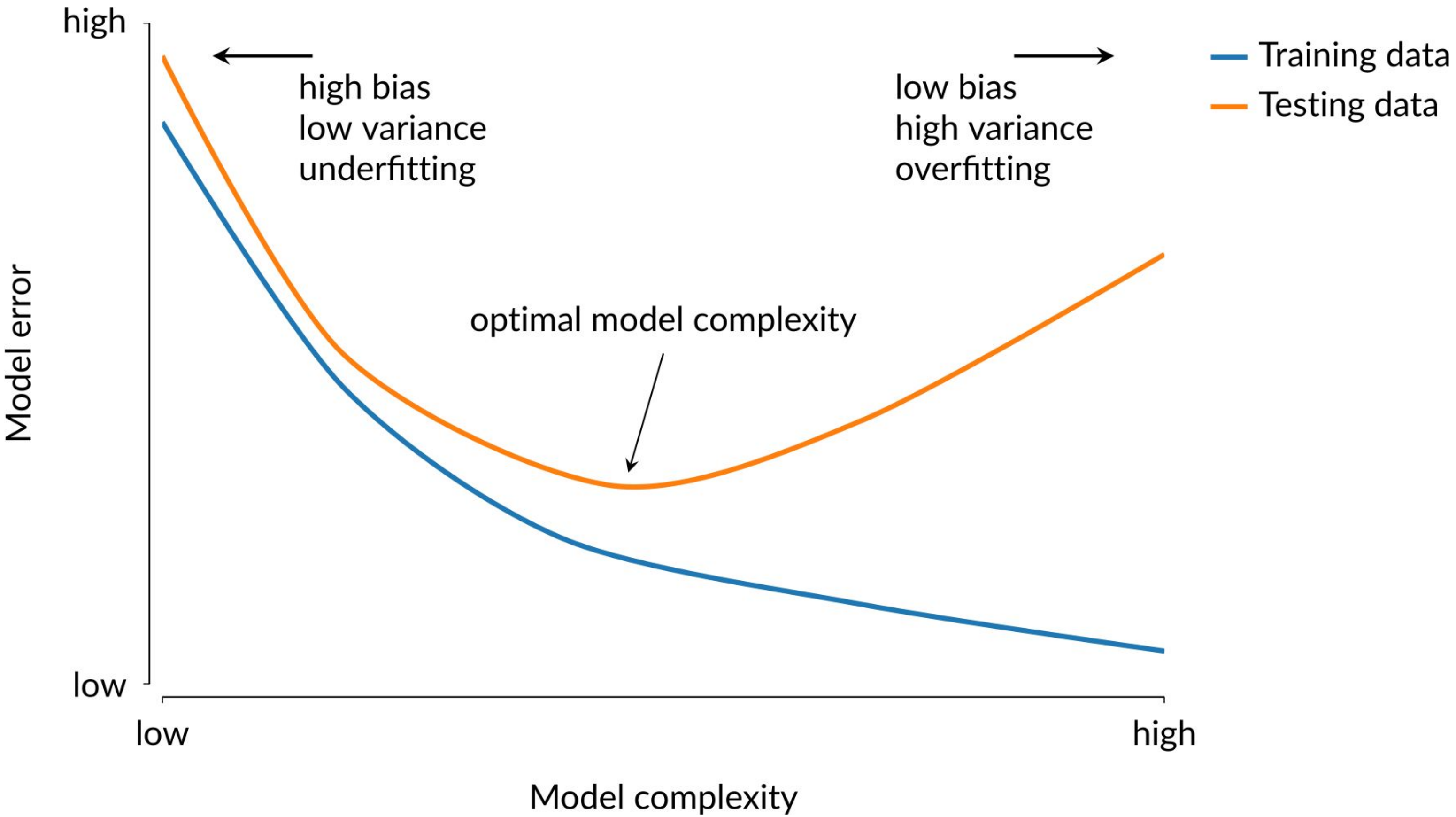
```
$ mlflow ui
```



inovex

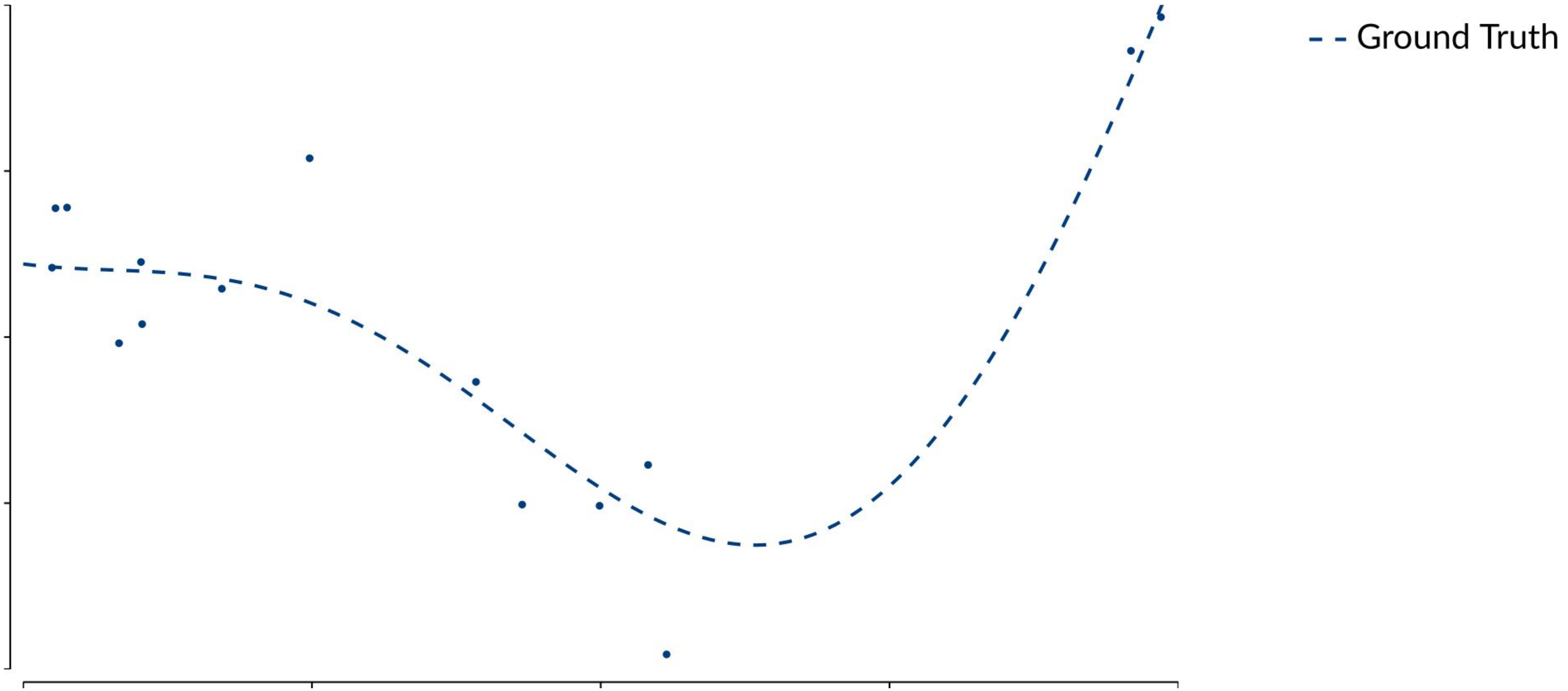
Overfitting

Bias-Variance-Tradeoff



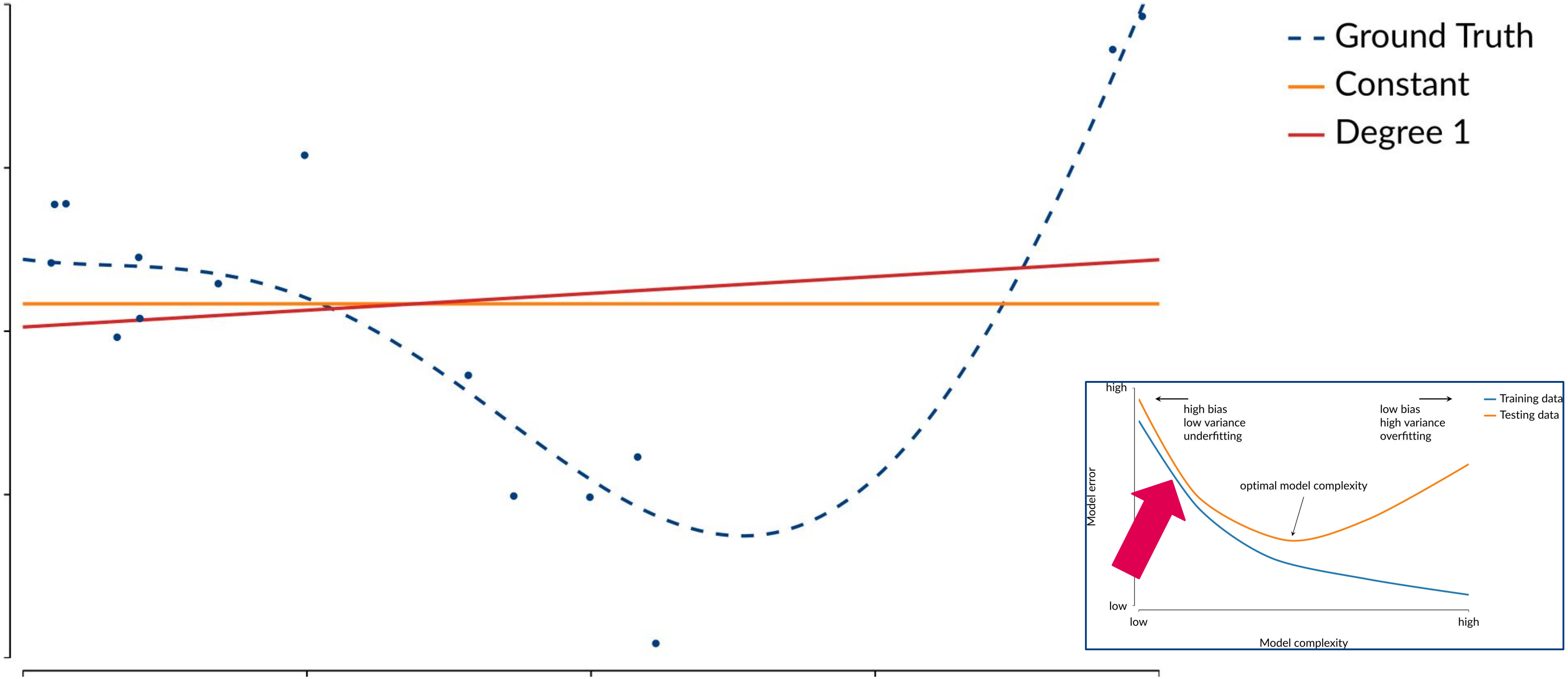
Bias-Variance-Tradeoff

Underfitting: Modell zu einfach



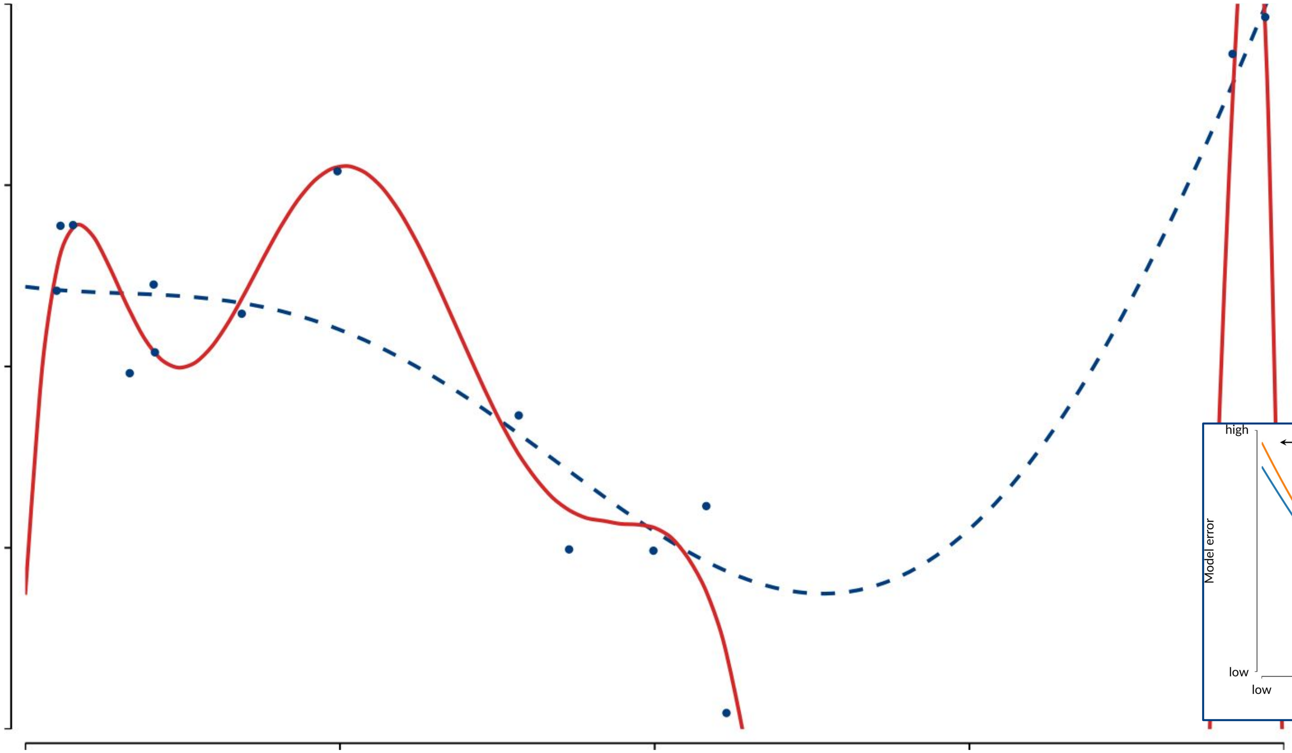
Bias-Variance-Tradeoff

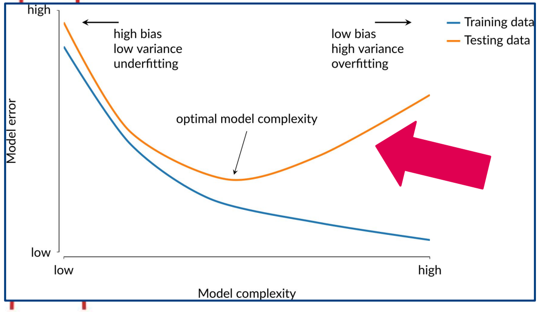
Underfitting: Modell zu einfach



Bias-Variance-Tradeoff

Overfitting: Modell zu komplex

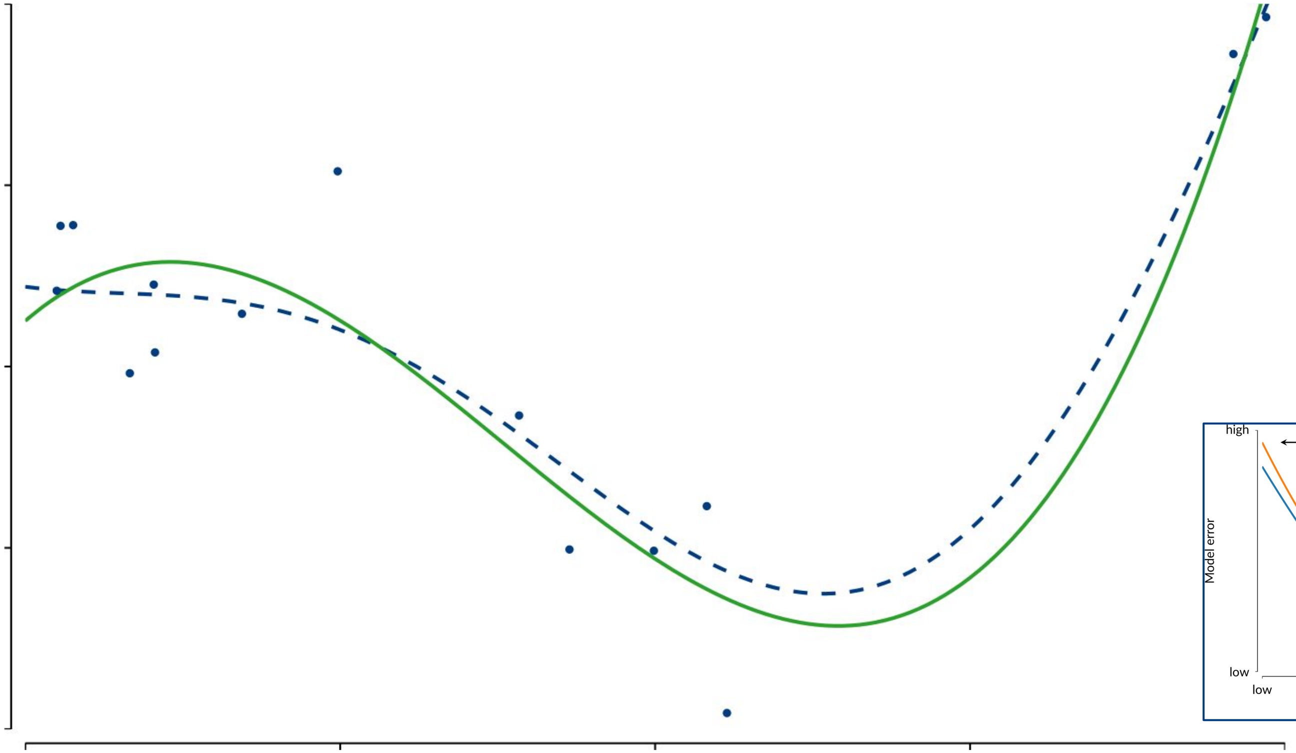


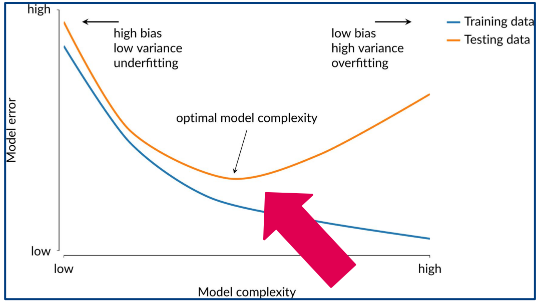


inovex

Bias-Variance-Tradeoff

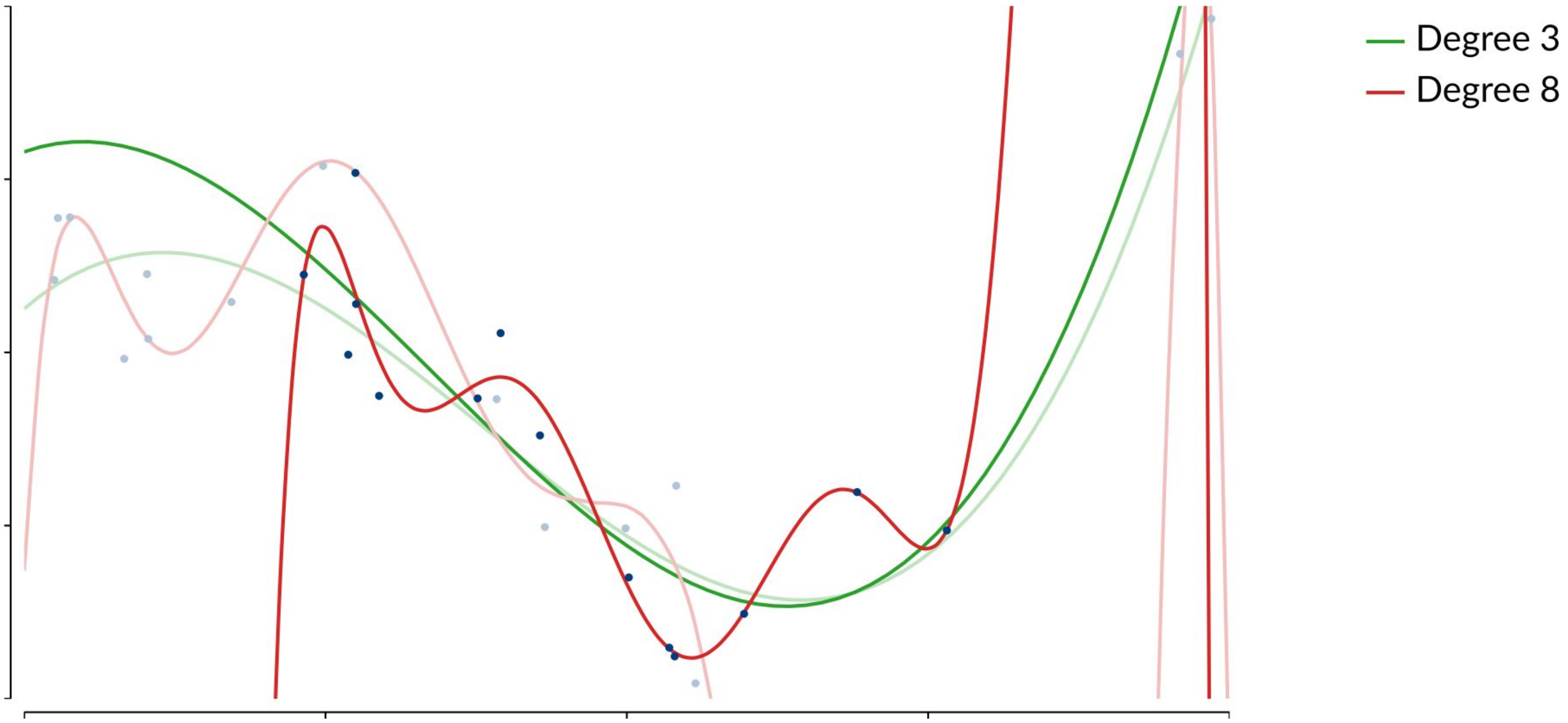
Optimale Modellkomplexität





Bias-Variance-Tradeoff

Was ist die Varianz eines Modells?



Voraussetzungen für Overfitting

› Modell zu komplex ≈ zu viele Parameter

→ Mit einfachen Modellen starten!

› Zu lange trainiert

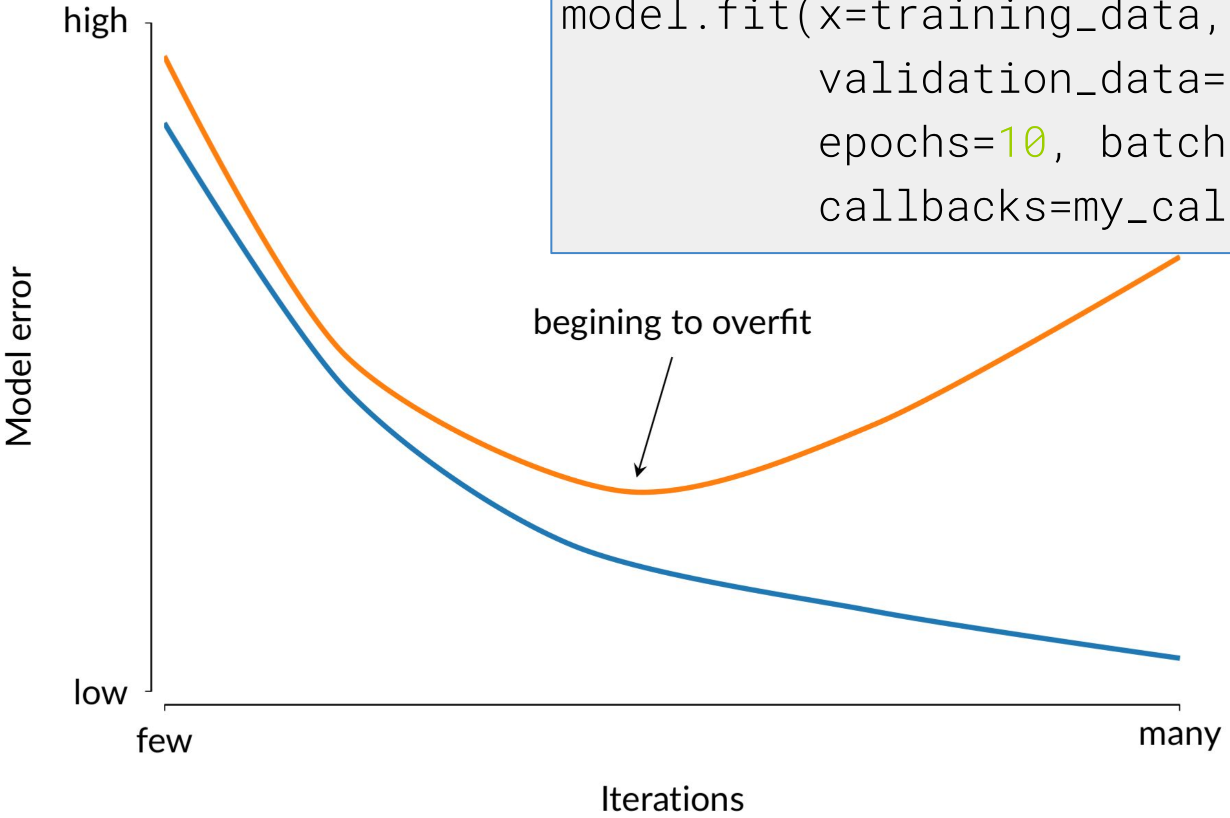
→ Training überwachen!

› Zu wenig / nicht repräsentative Daten

→ Daten anschauen!

Overfitting erkennen

```
model.fit(x=training_data, y=labels,
          validation_data=(val_x, val_y),
          epochs=10, batch_size=32,
          callbacks=my_callbacks)
```



inovex

Datensätze

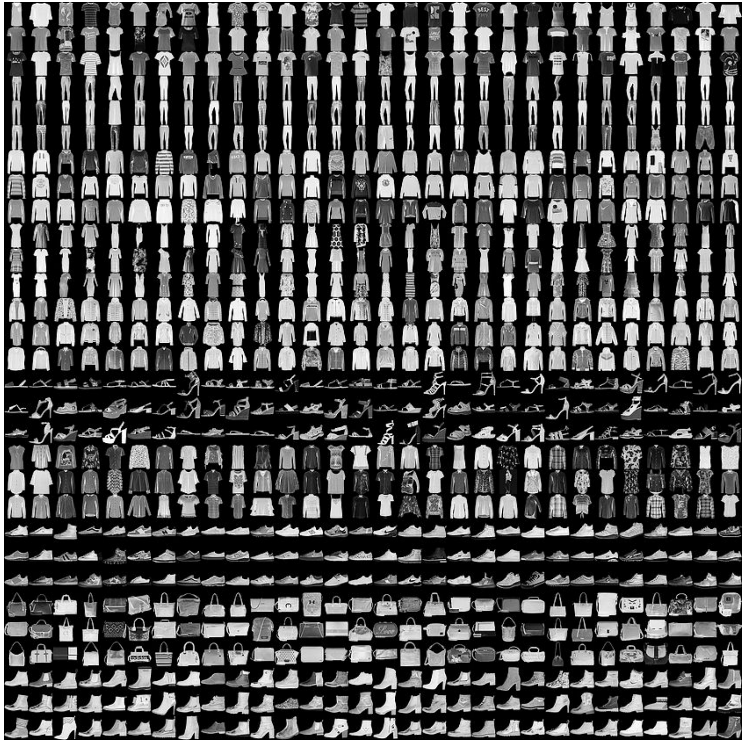
Fashion-MNIST (Zalando Research)

Abstract

We present Fashion-MNIST, a new dataset comprising of 28×28 grayscale images of $70,000$ fashion products from 10 categories, with $7,000$ images per category. The training set has $60,000$ images and the test set has $10,000$ images. Fashion-MNIST is intended to serve as a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms, as it shares the same image size, data format and the structure of training and testing splits. The dataset is freely available at `https://github.com/zalandoresearch/fashion-mnist`.

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. https://arxiv.org/pdf/1708.07747.pdf

Fashion-MNIST (Zalando Research)



t-SNE

Bildquelle: https://github.com/zalandoresearch/fashion-mnist

CIFAR-10

(Canadian Institute For Advanced Research)

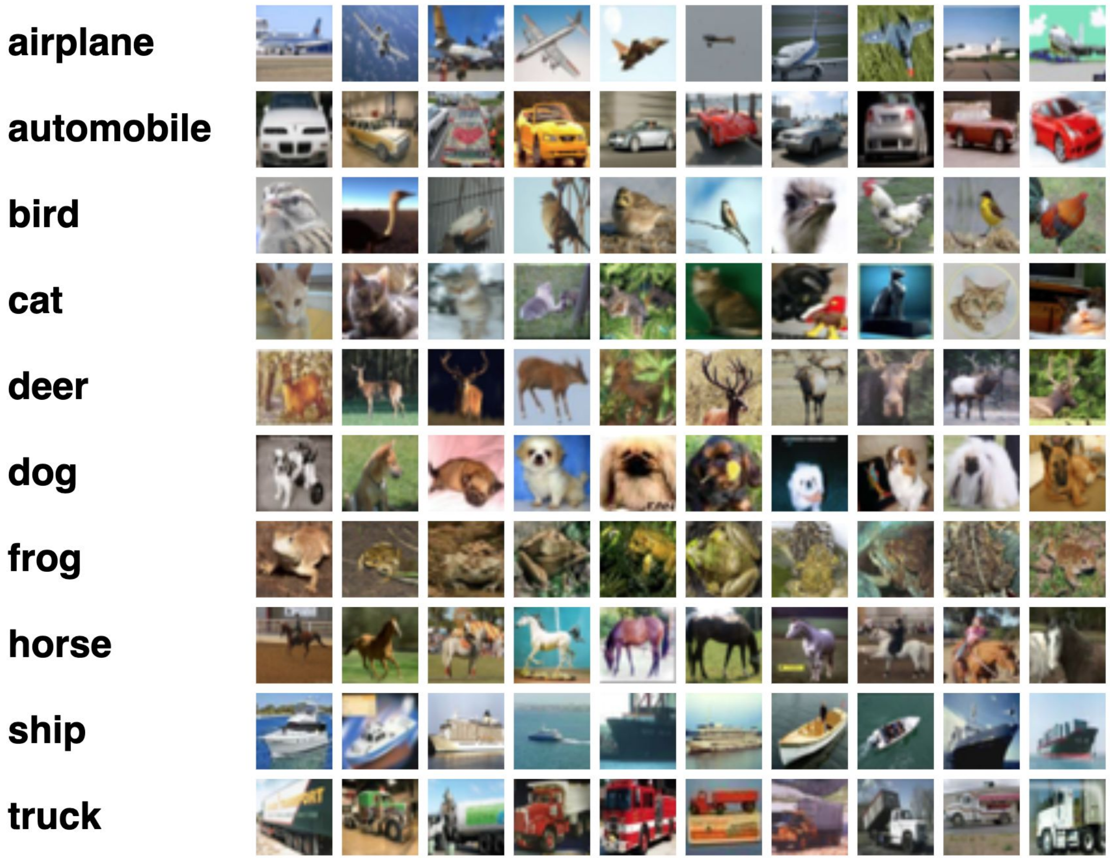
“The CIFAR-10 and CIFAR-100 are **labeled subsets of the 80 million tiny images** dataset. They were collected by “Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.

[...]

The CIFAR-10 dataset consists of **60000 32x32 colour images** in **10 classes,** with **6000 images per class.** There are 50000 training images and 10000 test images.”

https://www.cs.toronto.edu/~kriz/cifar.html

CIFAR-10



Bildquelle: https://www.cs.toronto.edu/~kriz/cifar.html colab

Google Colab

- In GCP gehosteter Jupyter Notebook Service
- Kollaborative Arbeit (ähnlich GDoc) möglich
- Frei nutzbar (inklusive GPU/TPU!)
  - → Ressourcen sind limitiert
  - → GPU/TPU Runtime sparsam nutzen!
- Notebooks und Daten liegen im Google Drive

inovex

Feedback

https://forms.gle/qKrigjB75GfZmw8P7

Vielen Dank

Robin Baumann
rbaumann@inovex.de

Matthias Richter
mrichter@inovex.de

inovex GmbH
Ludwig-Erhard-Allee 6
76131 Karlsruhe

inovex