Agenda

Start: 09:00 Uhr

1 Einführung

2 Business Understanding

3 Data Understanding

4 Data Preparation

5 Modeling

6 Evaluation

7 Modelloptimierung

8 Pause

9 Deployment

10 Diskussion und Abschluss

Ende: Ca. 15:00 Uhr

Modelling

ToDo: Hier möchten wir zunächst ein sehr einfaches Modell aufbauen. Diese soll aus den folgenden Schichten bestehen:

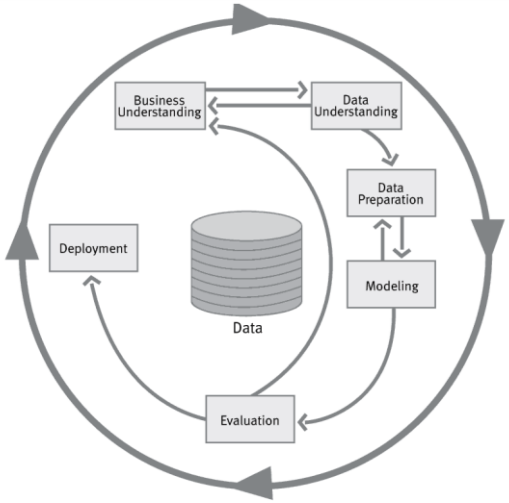
- Convolutional layer mit 8 Filtern und kernel_size 3x3
- MaxPooling layer mit pool_size 2x2 und Schrittweite 2
- Flatten layer
- Vollverbundens layer mit 16 Neuronen und Aktivierungsfunktion 'relu'
- Ausgabe layer --> Vollverbunden aber nur ein Neuron und Aktivierungsfunktion 'sigmoid'

Die layer können mit der Funktion model.add(*Definition der Schicht*) nacheinander hinzugefügt werden. Die layer können w

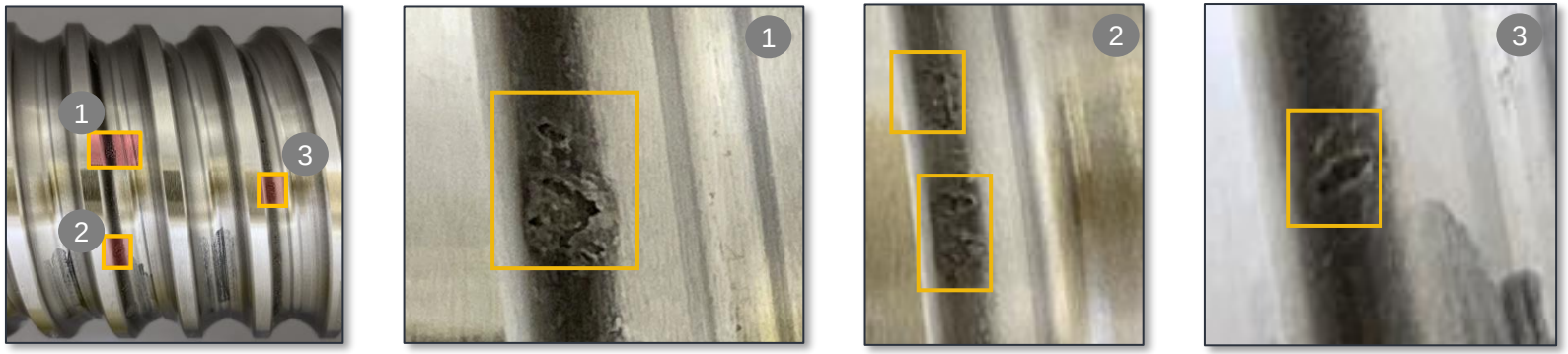
- Convolutional layer: layers.Conv2D(filter=*Anzahl der Filter*, kernel_size=(*f,f*), activation='*Aktivierungsfunktion*') --> in d input_shape=(150, 150,1). Dies ist bei weiteren Schichten nicht mehr nötig.
- MaxPooling layer: layers.MaxPooling2D(pool_size=(*q,q*), strides=*s*)
- Flatten layer: layers.Flatten()
- Vollverbundenes layer: layers.Dense(units = *Anzahl der Neuronen*, activation='*Aktivierungsfunktion*')

Anschließend wird das Modell trainiert, das Ergebniss des Trainings gezeigt und der Trainingsprozess visualisiert.

```
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt

# Festlegen der Modellarchitektur
model = models.Sequential()
# Ab hier können layer hinzugefügt werden
model.add(layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(150, 150, 1)))
model.add(layers.MaxPooling2D(pool_size=(2, 2), strides=2))
model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2), strides=2))
model.add(layers.Flatten())
model.add(layers.Dense(units=32, activation='relu'))
model.add(layers.Dense(units=1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Training des Modells
history = model.fit(x_train, y_train, epochs=3, batch_size=32,  validation_data=(x_test, y_test))
```



Angeleitete Entwicklung einer KI-Applikation im Rahmen des Vorgehensmodells Cross-Industry Standard Process for Data Mining (CRISP-DM)



Anwendungsfall: Bildbasierte Defekterkennung auf Gewindespindeln mit Convolutional Neural Networks

wbk Institut für Produktionstechnik
Nachwuchs-stiftung Maschinenbau
Kofinanziert von der Europäischen Union