



Willkommen zum online Brownbag ARIC 3.8.2021

Python, Numpy, Pandas für KI-Frischlinge - Ein Hands-On Einstieg

Prof. Dr. Randolph Isenberg und Jan-Malte Kapust HAW-Hamburg

HAW Hamburg, Berliner Tor 21, 20099 Hamburg
Institut: Produkt- und Produktionsmanagement
Forschungstransfer Zentrum: FTZ 3i – Intelligent Industrial Innovations

1. Einführung/Ziel (RI, JMK)

1. Der Autor und sein Team
2. Zeitleiste der AI
3. Typen der AI

2. Überblick Python, NumPy, Pandas, Matplotlib

1. Unser Lernumfeld – CNN für die Fashion Datenbank von Zalando
2. Colab
3. Was ist eigentlich?
4. Python
5. NumPy
6. Matplotlib
7. Pandas

Was ist unser Ziel heute?

- Schnell kann jeder ein CNN Neuronales Netz zur Bilderkennung laden und laufen lassen...
- ABER
- Wenn Sie die Daten verstehen oder Änderungen machen wollen...
- Dann beginnt das Suchen – Einarbeiten in Python etc...
- Wir geben Ihnen Tipps, die viel Zeit sparen – damit Sie den Spaß am vertieften Einstieg behalten.



**Prof. Dr.-Ing.
Randolf Isenberg**



Jan – Malte Kapust

Prof. Dr.-Ing. Randolph Isenberg

RTWH Aachen

- Seit 1982 Studium allg. Elektrotechnik/ Regelungstechnik
- 1990 Doktorarbeit: Knowledge-Based Integration of Production-Planning in Computer-Integrated-Manufacturing

Philips

- 1982 Forschung EU CIM/KI Projekte
- 1986 Kooperation KSL Stanford University

Daimler-Benz Aerospace Airbus GmbH

- 1991 Leitg. Prozessorganisation in der Produktionszentrale
- Leitg. Controlling Long Range

HAW Hamburg

- 1999 Professor Produktionsmanagement - Digitalisierung
 - 2017 DigiNet.Air BMBF – Digitalisierung Kompetenz Mittelstand Luftfahrt
 - 2021 ILIdenT – KI, Identifikation in der Supply Chain Luftfahrt
- CAE- ARIC Keynote Dez.2020
- AI in Engineering <https://tinyurl.com/9dvry7tu>

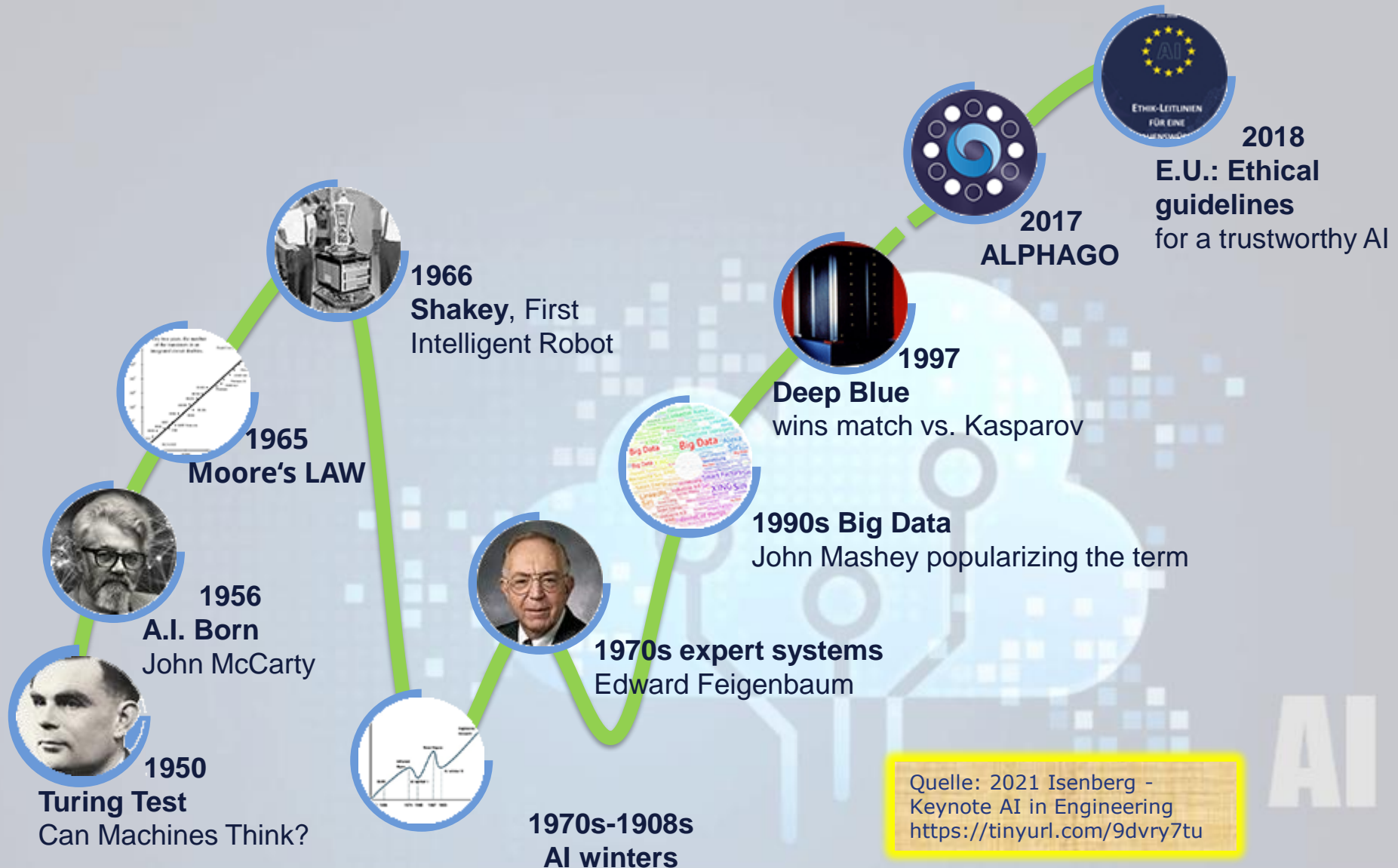
Jan - Malte Kapust

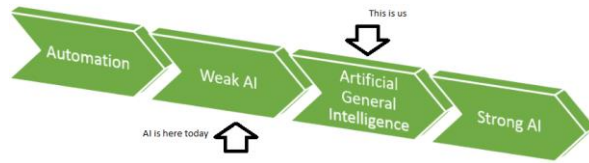
Hochschule Stralsund

- 2016 Bachelorstudium Maschinenbau
- 2020 Abschlussarbeit Mercedes Benz Werk Hamburg

HAW Hamburg

- 2021 Masterstudium ‚nachhaltige Energiesysteme im Maschinenbau‘
- 2021 ILIdenT – KI, Identifikation in der Supply Chain Luftfahrt





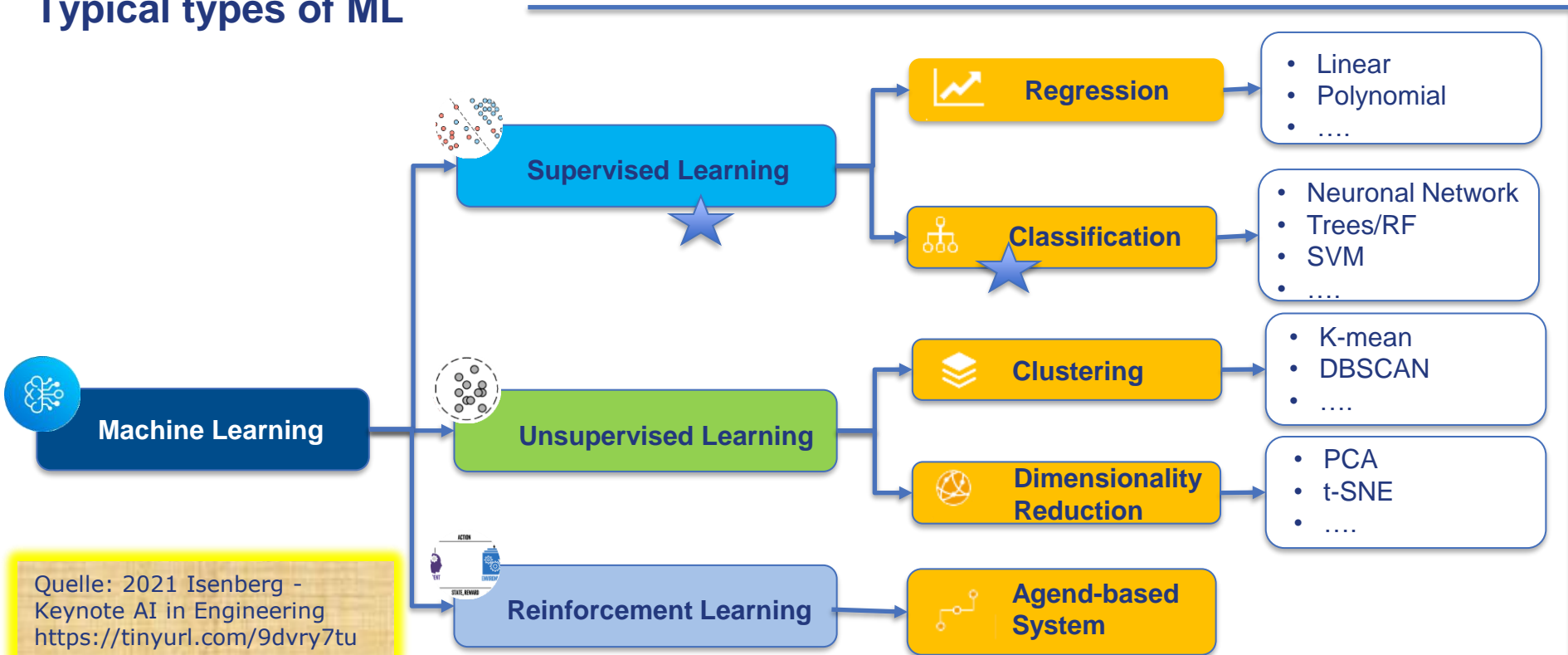
(<https://vincentlauzon.com/2015/09/16/strong-ai-existential-risks/>)

- **Weak AI** is nowhere near matching human intelligence, and it isn't trying to.
- We don't yet have **strong AI** in the world; it exists only in theory
- Of the types of AI, **super AI** is the one most people mean when they talk about robots taking over the world.

So far, we've only **achieved the first** of the three types of AI — **weak AI**.

(thinkautomation.com)

Typical types of ML

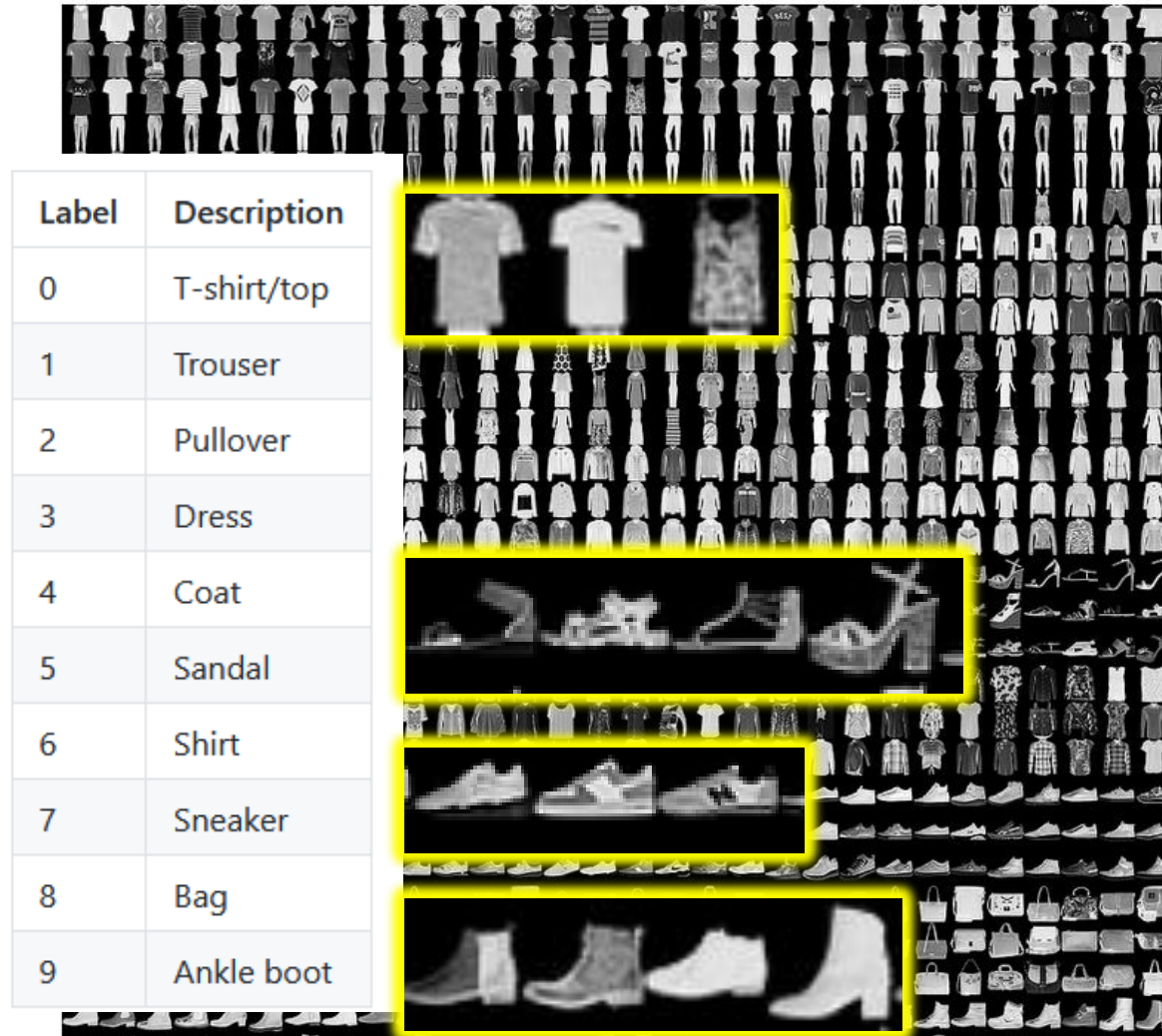


Aufgabe für die KI:

- Ordne ein Bild dem richtigen Kleidungstyp zu.

Es gibt:

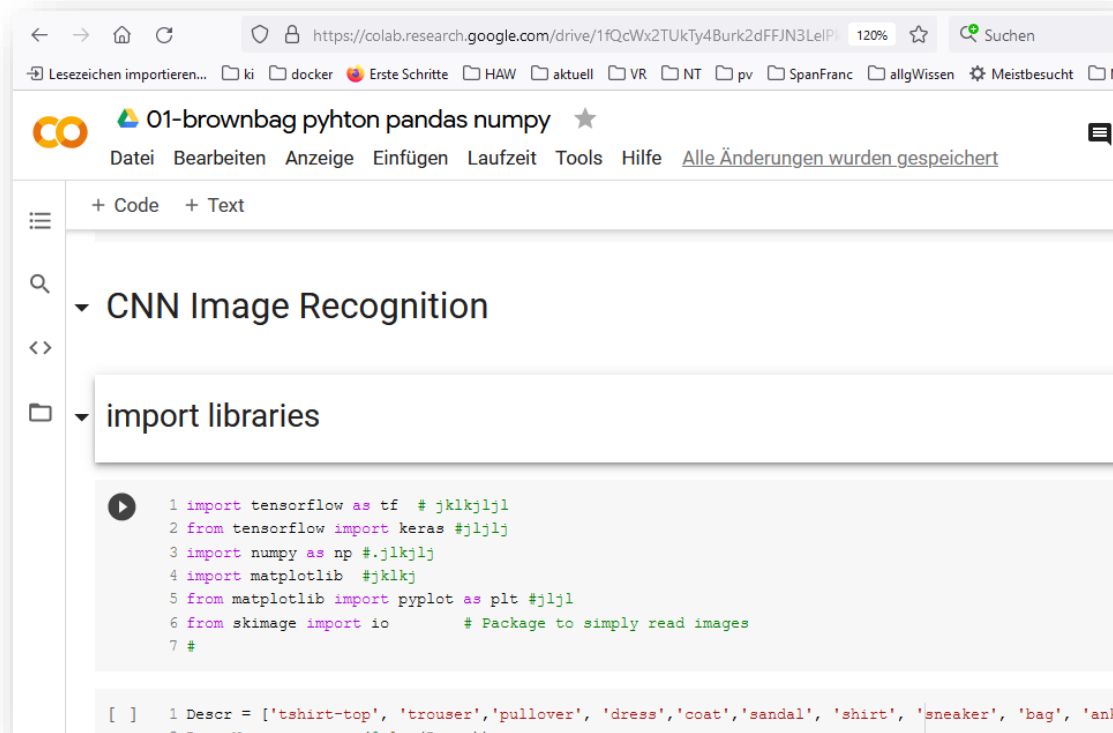
1. **Daten: 10 Kleidungstypen**
 - von T-Shirt/Top bis Ankle boot
 - Bildgröße: 28x28 Pixel, grau
 - Label Nr von 0 bis 9
2. **60000 Übungs-Bilder, -Label**
 - $X_{\text{train}}, y_{\text{train}}$
3. **10000 Test-Bilder, -Label**
 - $X_{\text{test}}, y_{\text{test}}$
 - 99,5% Genauigkeit acc
 - 10000.... ? falsch



Zalando SE. *Fashion-MNIST in GitHub*. Python. 2017. Reprint, Zalando Research, 2021. <https://github.com/zalando-research/fashion-mnist>.

Was ist Colab?

1. Von Google
2. Direkt im Browser nutzbar auf diversen Plattformen
3. Notebook nutzbar – Text und Programm
4. Python und sehr viele Bibliotheken verfügbar



Was ist eigentlich?

Python – Numpy – Pandas – Matplotlib

Python

- höhere Programmiersprache mit dem Anspruch der guten Lesbarkeit
- beliebteste Programmiersprache für artificial intelligence/machine learning



<https://www.inovex.de/wp-content/uploads/2021/04/training-python.png>

Programmbibliotheken für die Programmiersprache Python:

- NumPy (Numeric Python)
 - schnelle numerische Datenverarbeitung durch das Arbeiten mit Arrays
 - Ideal bis 50.000 Datenreihen
- Pandas
 - schnelle Datenverarbeitung von sehr vielen Daten durch Arbeiten in Tabellenstruktur
 - Ideal ab 500.000 Datenreihen (Industrie)
- Matplotlib
 - mathematische Darstellungen aller Art
 - Darstellungen von Bildern möglich



https://numpy.org/images/logo_numpy.svg

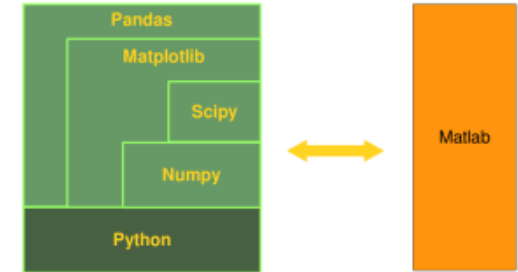


https://upload.wikimedia.org/wikipedia/commons/thumb/e/ed/Pandas_logo.svg/2880px-Pandas_logo.svg.png



https://matplotlib.org/_static/logo2_compressed.svg

Python, eine Alternative zu Matlab



Klein, Bernd. *Numerisches Python: arbeiten mit NumPy, Matplotlib und Pandas*. München: Hanser, 2019. Seite 24

Python

1. Datentypen (Bsp.)

1. List – Geordnet, **änderbar**, indiziert
 - `list1 = [1, 'Ball', 23, [2, 'Markus']]`
 - `list2 = [4, 6, 9]`
 - `list3 = [3, 4, 5]`
2. Tuple – Geordnet, **nicht änderbar**, indiziert
 - `tuple1 = (1, 'Ball', 23, (2, 'Markus'))`

2. Funktionen (Bsp.)

1. Len >>gibt die Länge des Datentyps aus
 - `len(list1) >> 4`
2. ZIP >>paart Elemente miteinander, Rückgabewert: Iterator
 - `zip(list2, list3) >> [[4, 3], [6, 4], [9, 5]]`
3. List >>erzeugt aus einem Iterator eine Liste
 - `list(iterator)`
4. Print >>schreibt einen Text im korrekten Format
 - `print(f'Unser Spieler: {list1[3][1]}') >> 'Unser Spieler: Markus'`
5. Type >>gibt den Typen der Variablen aus
 - `type(tuple1) >> Tuple`
6. Map >>führt eine Funktion auf „Iterable“ (list, tuple), Returns Iterator
 - `map(function, iterable)`



Python

1. Datentypen (Bsp.)

1. List – Geordnet, **änderbar**, indiziert
 - `list1 = [1, 'Ball', 23, [2, 'Markus']]`
 - `list2 = [4, 6, 9]`
 - `list3 = [3, 4, 5]`
2. Tuple – Geordnet, **nicht änderbar**, indiziert
 - `tuple1 = (1, 'Ball', 23, (2, 'Markus'))`

2. Funktionen

1. Zugriff auf Range >>greift auf einen Bereich zu
 - `Teilliste = list1[1:3];`
`Teilliste >> ['Ball', 23]`
2. enumerate >>erzeugt 2 Variablen:
 - `Var1, Var2 = Enumerate(list2)`
`Var1 >> 0, 1, 2`
`Var2 >> 4, 6, 9`

Var1: Index des Elements
Var2: Wert des Elements
3. `For x in Range(5):`
`print(x)`
Ausgabe: 0, 1, 2, 3, 4 >>for-loop in Python mit Doppelpunkt, ohne Klammern: Indentprinzip!
4. `def function(Var):` >>definiert eine Funktion
`print(f' Die Variable ist: {Var}')`
`function(„ein Panda“) >> Die Variable ist: ein Panda`



NumPy

1. Datentypen

1. Arrays

- `npAr1 = np.array([1, 'Ball', 23, [2, 'Markus']])`

2. Funktionen

1. Arange

>>erstellt ein NumPy-Array mit 5 Elementen startend bei 0

- `np.arange(0,5) >> [0,1,2,3,4]`

2. Shape

>>gibt die Anzahl der Elemente einer Dimension aus

- `np.shape(npAr1) >> 4`
- `np.shape(image) >> (28, 28, 3)`

3. Zugriff

>>gibt ein bestimmtes Element aus

- `npAr1[3] >> [2, 'Markus']` >>4. Element
- `npAr1[3][1] >> 'Markus'` >>2. Element des 4. Elementes
- `npAr1[3][0] >> 2` >>1. Element des 4. Elementes

4. Newaxis

>>erzeugt eine neue zusätzliche Dimension

- `npAr1 = npAr1[...,np.newaxis]`
`npAr1 >> [[1],['Ball'], [23], [[2, 'Markus']]])`
- `npAr1.shape >> (4,1)`

5. argmax

>>findet den Index mit dem Maximalen Wert



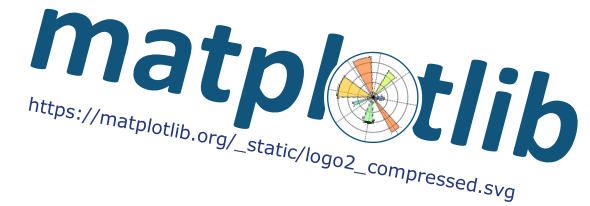
Matplotlib

1. Datentypen

1. Diagramm-Koordinaten
2. Bilder aus NumPy, jpg, png

2. Funktionen

1. `Plt.imshow('Pfad')` >>erstellt ein Bild mit zugehörigem Pfad
2. `Plt.title('Bildtitel')` >>erstellt einen Bildtitel
3. `Plt.show()` >>zeigt das erstellte Bild



Pandas

1. Datentypen

1. Tabellen

2. Funktionen

1. `Pd.Read_excel(Pfad)` >>liest eine Excel-Datei ein
2. `Pd.Head(2)` >>greift auf die ersten 2 Zeilen zu
3. `Pd.loc` >>zeigt alle Werte der Zeile mit bestimmten Inhalt
 - `Pd.loc['Kundennummer 232344']`
4. `Pd.iloc` >>zeigt alle Zeilenwerte der Spalte 2 und 3
 - `Pd.iloc[Zeile, Spalte]`
`Pd.iloc[:, [1,2]]`





Willkommen zum online Brownbag ARIC 3.8.2021
Python, Numpy, Pandas für KI-Frischlinge - Ein Hands-On Einstieg
Prof. Dr. Randolph Isenberg und Jan-Malte Kapust HAW-Hamburg

HAW Hamburg, Berliner Tor 21, 20099 Hamburg
Institut: Produkt- und Produktionsmanagement
Forschungstransfer Zentrum: FTZ 3i – Intelligent Industrial Innovations

1. 4 Einführung/Ziel (RI, JMK)

1. Der Autor und sein Team
2. Zeitleiste der AI
3. Typen der AI 1

2. Überblick Python, [NumPy](#), Pandas, [Matplotlib](#)

1. Unser Lernumfeld – CNN für die Fashion Datenbank von Zalando
2. [Colab](#)
3. Was ist eigentlich?
4. Python
5. [NumPy](#)
6. [Matplotlib](#)
7. Pandas

Was ist unser Ziel heute?

- Schnell kann jeder ein CNN Neuronales Netz zur Bilderkennung laden und laufen lassen...

ABER

- Wenn Sie die Daten verstehen oder Änderungen machen wollen....
- Dann beginnt das Suchen – Einarbeiten in Python etc...
- Wir geben Ihnen Tipps, die viel Zeit sparen – damit Sie den Spaß am vertieften Einstieg behalten.

Gerne beantworten wir Ihre Fragen

1. Verbinden mit dem eigenen Google Drive

```
[ ] from google.colab import drive
    drive.mount('/content/drive')
```

Mounted at /content/drive

2. Importieren von Bibliotheken

import libraries

```
import tensorflow as tf # ist ein Framework zur datenstromorientierten Programmierung mit Tensoren (wikipedia)
from tensorflow import keras # Python Bibliothek für Deep Learning
import numpy as np #Erläuterung Bibliotheken
import matplotlib
from matplotlib import pyplot as plt
from skimage import io # Package to simply read images
#
```

```
[ ] Descr = ['tshirt-top', 'trouser', 'pullover', 'dress', 'coat', 'sandal', 'shirt', 'sneaker', 'bag', 'ankle-boot'] #Erläuterung
DescrNo = np.arange(0, len(Descr)) #Erläuterung arange
Label = list(zip(DescrNo, Descr)) #2 Mal - Erläuterung zip, list
print(f'Label: {Label}') #Erläuterung print
#
```

Label: [(0, 'tshirt-top'), (1, 'trouser'), (2, 'pullover'), (3, 'dress'), (4, 'coat'), (5, 'sandal'), (6, 'shirt'), (7, 'sneaker'), (8, 'bag'), (9, 'ankle-boot')]

3. Beispiel-Datensatz aus Keras laden

load dataset & show test image

```
[ ] [(X_train, y_train), (X_test, y_test)] = keras.datasets.fashion_mnist.load_data()
#
print(type(X_train))      #Erläuterung Type
print(X_train.shape)      #Erläuterung Shape

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
32768/29515 [=====] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26427392/26421880 [=====] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
8192/5148 [=====] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4423680/4422102 [=====] - 0s 0us/step
<class 'numpy.ndarray'>
(60000, 28, 28)
```


4. Bild anzeigen

```
display(X_test[0])      ##Erläuterung Zugriff
plt.imshow(X_test[0] , cmap='gray')
plt.title(f'Kleidungsstück: {y_test[0]} {Label[9][1]}')
plt.show() # Was ist weiß.. 0 oder 255?
```

[illegible][illegible]

```
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0], dtype=uint8)
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0], dtype=uint8)
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0], dtype=uint8)
```

Kleidungsstück: 9 ankle-boot



0 5 10 15 20 25

5. Datennormierung und Modellierung

create, compile and fit model

```
[7] # Vorbereitung der Daten
X_train_norm = X_train/255
X_test_norm = X_test/255
#print(X_test_norm.shape)
X_train_norm = X_train_norm[..., np.newaxis] ##Erläuterung newaxis
X_test_norm = X_test_norm[..., np.newaxis]
#print(X_test_norm.shape)

[8] # Neuronale Netz
model = keras.models.Sequential()
# erster conv Block
model.add(keras.layers.Conv2D(32, kernel_size=(5, 5), padding='valid', activation='relu',
                             input_shape=(28,28,1)))
model.add(keras.layers.MaxPooling2D(pool_size=2))
# flattening
model.add(keras.layers.Flatten()) # Umwandeln für Vollkontakt-Netz
# Vollkontakt-Netz
model.add(keras.layers.Dense(10, activation='softmax'))
model.summary()
#
model.compile(loss="sparse_categorical_crossentropy",
              optimizer="nadam",
              metrics=["accuracy"])
history = model.fit(X_train_norm, y_train, epochs=2,
                   validation_data=(X_test_norm, y_test))
test_loss, test_acc = model.evaluate(X_test_norm, y_test)
#
#print('Modell:', test_acc)
```

Model: "sequential"

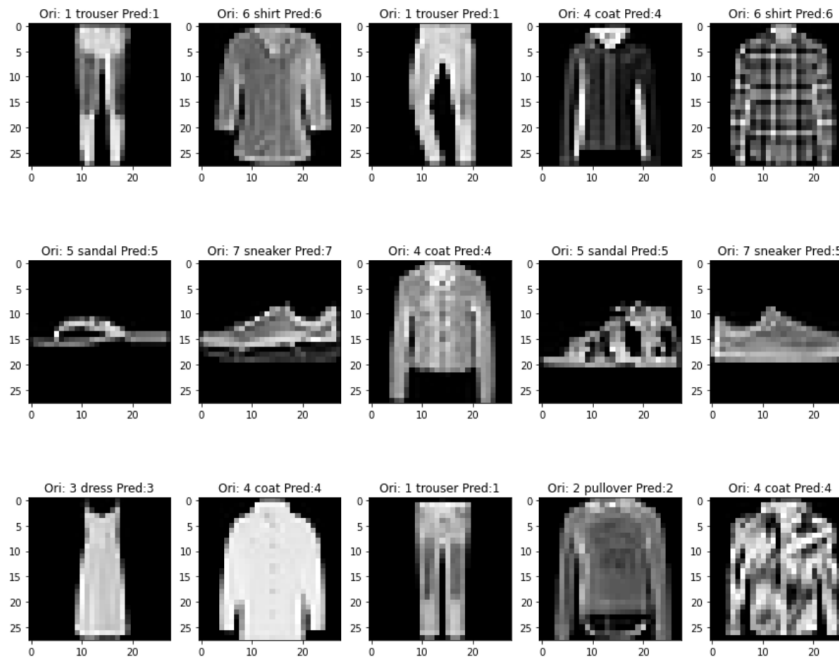
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 24, 24, 32)	832
max_pooling2d (MaxPooling2D)	(None, 12, 12, 32)	0
flatten (Flatten)	(None, 4608)	0
dense (Dense)	(None, 10)	46090
Total params: 46,922		
Trainable params: 46,922		
Non-trainable params: 0		

```
Epoch 1/2
1875/1875 [=====] - 34s 18ms/step - loss: 0.4593 - accuracy: 0.8386 - val_loss: 0.3593 - val_accuracy: 0.8745
Epoch 2/2
1875/1875 [=====] - 32s 17ms/step - loss: 0.3220 - accuracy: 0.8865 - val_loss: 0.3289 - val_accuracy: 0.8840
313/313 [=====] - 2s 5ms/step - loss: 0.3289 - accuracy: 0.8840
```

6. Label Vorhersage und for-loop

predict label of an object

```
[ ] imgStart = 3
imgEnd = 20
plt.figure(figsize=(15,17))
pred = list(map(np.argmax,model.predict(X_test_norm[imgStart:imgEnd+1]))) # 3 MAL - Erläuterung argmax, Map, Zugriff auf range
picno = 1
for imgno, predl in enumerate(pred, start=imgStart): # 2 Mal - Erläuterung enumerate, for
    plt.subplot(4,5,picno)
    plt.imshow(X_test[imgno], cmap='gray')
    plt.title(f'Ori: {y_test[imgno]} {Label[y_test[imgno]][1]} Pred:{predl}')
    picno = picno + 1
plt.show()
```



7. Pandas import und Bildausgabe von einem Bild aus dem eigenen Google Drive

Pandas - Bild und Excel einlesen und Dataframe anzeigen

opensource library in Python

used for data science an analytics

<https://www.journaldev.com/29055/python-pandas-module-tutorial>

```
[ ] import pandas as pd
import os
#
cwd = os.getcwd()
print(cwd)
```

/content

```
[ ] pfad_ = "/content/drive/MyDrive/ColabNotebooks/rijmk/Bilder/panda01-pexels-mike-van-schoonderwalt-5504764 (1).jpg"
img = plt.imread(pfad_)
plt.imshow(img)
plt.title('Panda')
plt.axis('off')
plt.show()
```



8. Pandas Tabellenimport und -darstellungen

```
[ ] def printb(str):                ##Erläuterung def function
    # print bold an insert line above
    print(f'\n\033[1m{str}\033[0m')

pathcol = '/content/drive/MyDrive/ColabNotebooks/rijmk/Data/bb-productlist.xlsx'
df1 = pd.read_excel(pathcol, index_col=1)
#
printb("Komplette Tabelle:")
display(df1)
#
printb("Nur ersten beiden Zeilen:")
display(df1.head(2))
#
printb("Nur Spalte 2 und 3:")
df2 = df1.iloc[:, [1,2]]
display(df2)
#
printb("Nur für Label KX")
df3 = df1.loc['KX']
df3
```

Komplette Tabelle:

	Nr	Gewicht	Reichweite	Tragkraft
Produktkürzel				
KX	1	10	1	5
KP	2	13	2	7
KN	3	14	4	9
KD	4	20	8	10
KX	5	11	1	4

Nur ersten beiden Zeilen:

	Nr	Gewicht	Reichweite	Tragkraft
Produktkürzel				
KX	1	10	1	5
KP	2	13	2	7

Nur Spalte 2 und 3:

	Gewicht	Reichweite
Produktkürzel		
KX	10	1
KP	13	2
KN	14	4
KD	20	8
KX	11	1

Nur für Label KX

	Nr	Gewicht	Reichweite	Tragkraft
Produktkürzel				
KX	1	10	1	5
KX	5	11	1	4