**Künstliche Intelligenz in der Firma DRUCKREIF**

|  |  |
| --- | --- |
| Wahlpflichtfach | Digitale Transformation |
| Lernbereich | Lerngebiet 2  Funktionsweisen von Softwareanwendungen in Netzwerken beurteilen |
| Querverweise zu weiteren Themengebieten | Informationstechnik, Datenbanken, Datenanalyse, Softwareentwicklung |
| Zeitrahmen | 4 Unterrichtstunden |
| Benötigtes Material | Ein Computer je Schülerin und Schüler, Zugang zu einer freien Entwicklungsoberfläche im Browser über das Internet |

# **Kompetenzerwartungen**

Die Schülerinnen und Schüler …

* informieren sich über die Möglichkeiten des Einsatzes von Techniken der künstlichen Intelligenz in technischen und wirtschaftlichen Anwendungen und präsentieren die grundlegenden Funktionsweisen.

**Aufgabe für eine vollständige Handlung**

1. **Orientieren:**

In der Firma DRUCKREIF werden Rotationsdruckmaschinen nach dem Offsetdruckprinzip mit Rasterwalzen eingesetzt. In den einstellbaren Rasterwalzen befinden sich zur Überwachung des Rasterwinkels, des Schöpfvolumens und der Rastertiefe RFID-Sensoren, die diese Daten erfassen und über eine Schnittstelle verfügbar machen. Zudem wird gleichzeitig die Qualität der Lackschicht als 1 (in Ordnung) oder 0 (nicht in Ordnung) automatisch gemessen und über eine Schnittstelle zur Verfügung gestellt.

Mit den Eingangsdaten Rasterwinkel, Schöpfvolumen und Rastertiefe und dem Ausgangswert „1“ für ein gutes Druckergebnis, sowie „0“ für ein schlechtes Druckergebnis soll eine künstliche Intelligenz angelernt werden, um einen Zusammenhang in den Daten zu ermitteln, um diesen für eine vorbeugende Wartung zu nutzen.

1. **Informieren:**

Die Schülerinnen und Schüler bekommen das vollständige Programm eines neuronalen Netzes mit Erläuterungen und Literaturangaben für die Aufgabenstellung gestellt.

Die Schülerinnen und Schüler erarbeiten sich die Funktionsweise des neuronalen Netzes anhand der Programmbeschreibungen und Literatur in Eigenarbeit (Phase „Think“). Im Anschluss müssen sie sich gegenseitig den Programmaufbau erklären (Phase „Pair“). In einer Präsentation stellen die Schüler/innen ihre Ergebnisse der Experimente mit dem neuronalen Netz vor (Phase „Share“). Mit Verständnisfragen wird der Lernerfolg der Schülerinnen und Schüler überprüft und sichergestellt. Die Lehrkraft steht in der gesamten Lernzeit als Moderator und Unterstützer zur Verfügung.

1. **Planen:**

Die Schülerinnen und Schüler ermitteln die Einflussfaktoren für ein gut trainiertes neuronales Netz mit einem geringen Netzfehler und planen verschiedene Vorgehensweisen für die Aufgabe das Netz zu trainieren, zu testen und zu optimieren.

1. **Durchführen**

Die Schülerinnen und Schüler starten das neuronale Netz in einer freien Entwicklungsumgebung im Browser und nehmen über verschiedene Parameter wie z. B. Netzstruktur, Epochenanzahl, Lernschrittgröße - wie in der Planung festgelegt -Veränderungen zur Optimierung des neuronalen Netzes vor.

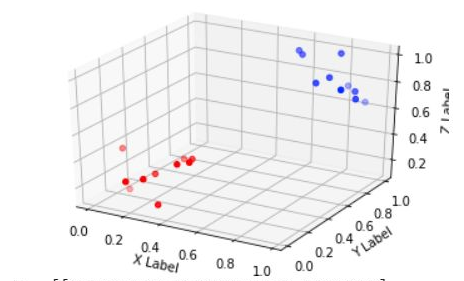
Die Parameter für einen möglichst minimalen Netzfehler werden unter den Schülerinnen und Schüler ausgetauscht und diskutiert.

1. **Kontrollieren und Bewerten**

Die Schüler/innen präsentieren die neuronalen Netze mit den in den Experimenten ermittelten neuen Parametern und erläutern den verbesserten Lernprozess. Die Lehrkraft stellt Fragen zur Funktionsweise des neuronalen Netzes und prüft die richtige Verwendung der Begrifflichkeiten.

# **Beispiele für Produkte und Lösungen der Schülerinnen und Schüler**

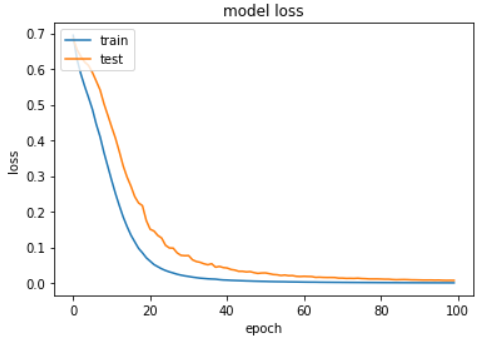
Die drei Eingangsdaten Rasterwinkel (x-Achse), Schöpfvolumen (y-Achse) und Rastertiefe (z-Achse) werden simuliert (Zufallswerte).  
Die Ausgangswerte „1“ für ein gutes Druckergebnis werden rot und die Werte „0“ für ein schlechtes Druckergebnis blau dargestellt.  
Es handelt sich dabei um eine Klassifizierungsaufgabe mit drei Eingangswerten und einem Ausgangswert.



Das bereits vollständige Programm eines neuronalen Netzes wird den Schülerinnen und Schülern von der Lehrkraft zur Verfügung gestellt oder wie im untenstehenden Programmablauf von den Schülerinnen und Schülern mit Programmierkenntnissen aus der Softwareentwicklung selbstständig erstellt (Schülerlösung):

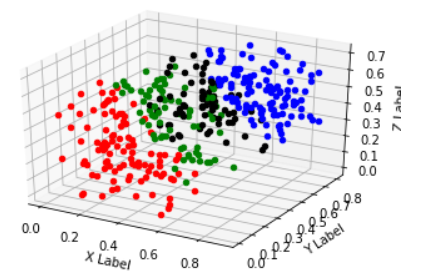
|  |
| --- |
| #NEURONALES NETZ FÜR DAS UNTERRICHTSBEISPIEL "Firma Druckreif"  #-------------------------------------------------------------  import numpy  import scipy.special  import matplotlib as mlp  from random import choice  from random import \*  from sklearn.datasets import load\_digits  import matplotlib.pyplot as plt  import matplotlib.pyplot as plt2  import numpy as np  import io  import pandas as pd  #---DATENMENGEN ERZEUGEN---  #3D Datenwolken mit scikit erzeugen  import numpy as np  import matplotlib.pyplot as plt  from mpl\_toolkits.mplot3d import axes3d  # Create data  **N = 10 #Anzahl von Punkten**  tar = []  ro = []  bl = []  xra = []  yra = []  zra = []  xba = []  yba = []  zba = []  pkte = []  i = 0  #rote Punkte zwischen 0 ... 0.4  for i in range(0,N):  **xr = (np.random.rand()\*0.4) #rot = target = 1**  **yr = (np.random.rand()\*0.4)**  **zr = (np.random.rand()\*0.4)**  #print("xr: ",xr)  xra.append(xr)  yra.append(yr)  zra.append(zr)  r = [xr,yr,zr]  ro.append(r)  pkte.append(r) #einen roten Punkt anhängen  print("rot: ",ro)  print("roten Punkte: ", "x: ", xr,"y: ", yr,"z:",zr)  **xb = (0.6 + np.random.rand()\*0.4) #blau = target = 0**  **yb = (0.6 + np.random.rand()\*0.4)**  **zb = (0.6 + np.random.rand()\*0.4)**  xba.append(xb)  yba.append(yb)  zba.append(zb)  b = [xb,yb,zb]  bl.append(b)  pkte.append(b) #einen blauen Punkt anhängen  print("blau: ", bl)  print("blauen Punkte: ", "x: ", xb,"y: ", yb,"z:",zb)  tar = tar + [0,1] #Targets - erst rot = 0, dann blau = 1 usw.  print(tar)  #blaue Punkte zwischen 0.6 ... 1  fig = plt.figure()  ax = fig.add\_subplot(111, projection='3d')  ax.scatter(xra,yra,zra, c="red", marker='o')  ax.scatter(xba,yba,zba, c="blue", marker='o') #eine zweite Wolke im Graphen  ax.set\_xlabel('X Label')  ax.set\_ylabel('Y Label')  ax.set\_zlabel('Z Label')  plt.show()  #---------------------------------Code pass-----------------------------------------  #----------------------------------------------------------------------------------##  #3D PUNKTEWOLKEN mit zwei Farben  #pkt = ro + bl #rote und blaue Punktkoordinaten  pkt = pkte; #Wechsel von roten und blauen Punkten!!  #print("Alle Punkte:", pkt)  #print("Alle Targets: ",tar)  #Nun die Daten pkt = Koordinaten der Punkte und tar = Target (Farbe) in Arrays umwandeln  x = np.array(pkt) #die Koordinaten aller Punkte; np = numpy  y = np.array(tar) #der Lösungsvektor für alle Punkte  print("x: ", x,"Target: ",y)  #for i in range(0,N):  #print("Eingangswerte: ",x[i], "Targetwert: ", y[i])  # x und y werden im NN verwendet - davon 80 % Trainingsdaten und 20 % Testdaten  # x sind die Koordinaten aller Punkte = Eingänge = 3, y sind die Targetwerte  print("OK")  #--------------------------------------Code pass----------------------------------------  #---NN NETZ MIT TENSORFLOW--------------------------------------------------------------  #x alle Eingangsdaten, y alle Targetwerte zu den Eingangsdaten  from \_\_future\_\_ import absolute\_import, division, print\_function  #KLASSIFIZIERUNGSAUFGABE MIT ZWEIFARBIGEN PUNKTEWOLKEN  # TensorFlow and tf.keras  import tensorflow as tf  from tensorflow import keras  import numpy as np  from sklearn.utils import shuffle  print(tf.\_\_version\_\_)  lenTrain = int(len(x) \* 0.8)  #Anzahl der Trainingsdaten sind 80 % der Gesamtdaten  lenTest = len(x) - lenTrain  #Anzahl der Testdaten sind der Rest d.h. 20 % der Gesamtdaten  #Aus den Arrays x = alle Koordinaten und y = der Lösungsvektor  #zwei Teilarrays für das Training und Test erzeugen  x\_train = x[:lenTrain]  y\_train = y[:lenTrain]  x\_test = x[lenTrain:]  y\_test = y[lenTrain:]  #EIN NN MODELL ERZEUGEN > input\_dim = 3 bei x,y,z Koordinaten je Punkt  **model = keras.Sequential([**  **keras.layers.Dense(32, activation=tf.nn.relu, input\_dim=3), #input dim = 3 bei drei Koordinaten**  **keras.layers.Dense(32, activation=tf.nn.relu),**  **keras.layers.Dense(2, activation=tf.nn.softmax) #rot oder blau Klassifizierung**  **])**  model.compile(optimizer='adam',  loss='sparse\_categorical\_crossentropy',  metrics=['accuracy'])  #------------------------------------------------------------------------------------------------  #DAS MODELL TRAINIEREN > hier Epochenanzahl angeben > mit 15 Epochen doch zahlreiche Fehlvorhersagen, ab 100 Epochen sehr gute Netzqualität  history = model.fit(x\_train, y\_train, **epochs=100**, batch\_size=1, validation\_data=(x\_test, y\_test))  print(history.history.keys())  #------------------------------------------------------------------------------------------------  #VISUALISIERUNG DER ERGEBNISSE  test\_loss, test\_acc=model.evaluate(x\_test, y\_test)  print('Test accuracy:', test\_acc)  #Block auskommentieren:  print(history.history.keys())  plt.plot(history.history['accuracy'])  plt.plot(history.history['val\_accuracy'])  plt.title('model acc')  plt.ylabel('acc')  plt.xlabel('epoch')  plt.legend(['train', 'test'], loc='upper left')  plt.show()  # summarize history for loss  plt.plot(history.history['loss'])  plt.plot(history.history['val\_loss'])  plt.title('model loss')  plt.ylabel('loss')  plt.xlabel('epoch')  plt.legend(['train', 'test'], loc='upper left')  plt.show()  # ENDE DES NN  #------------------------------------------------------------------------------------------------------------------------------------  import io  import pandas as pd2  #EINE 3D GRAFIK AUSGEBEN  from mpl\_toolkits.mplot3d import Axes3D  import matplotlib.pyplot as plt2  fig2 = plt2.figure()  ax2 = fig2.add\_subplot(111, projection='3d')  # mit .evaluate wird der test loss ausgegeben - nicht die Ausgabedaten  erg= model.evaluate(x,y) #gibt den Netzwerkfehler aus: z.b. 0.058 = 5,8% x = Koordinaten der Punkte, y = Targetwerte = Lösungsvektor  print("Ergebnis von evaluate mit x,y: ", erg)  #-------------------------------------------------------------------------------------------------------------------------------------  ergb = model.predict(x) #Übergabe der Koordinaten aller neuer unbekannten Punkte > [0 1] = 1 und [1 0] = 0  #print("Alle Ausgaben nach predict: ", ergb)  anz = len(x)  fehler = 0  #++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  for i in range(0,anz):  print("x-y-z Wert: ", x[i], " Vorhersage: ", ergb[i], "Target: ", y[i]) #Alle Werte    if ((tar[i] == 1) & (ergb[[i],1] **< 0.95**)):  fehler = fehler + 1  print("Fehler bei 1")  farb = "black"  elif (((tar[i] == 1) & (ergb[[i],1] >= 0.95))):  farb = "blue"  elif ((tar[i] == 0) & (ergb[[i],0] < 0.95)):  fehler = fehler + 1  print("Fehler bei 0")  farb = "green"  else:  farb = "red"  #print("Farbe: ", farb)  #Größe der Anzeige von scatter ändern?  ax2.scatter(x[[i],0], x[[i],1], x[[i],2], c= farb, marker='o') #hier farb nicht farbe  ax2.set\_xlabel('X Label')  ax2.set\_ylabel('Y Label')  ax2.set\_zlabel('Z Label')  plt2.show()  #++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  print("Anzahl der Fehler <0.95: ", fehler)  fehl = fehler/(anz/100)  print("Fehlerquote in %: ", fehl, "%" )  print("ENDE") |

Die Schülerinnen und Schüler können nach einer Planungsphase an diesem Programm auch ohne Programmierkenntnisse verschiedene Parameter wie z. B. die Netzgeometrie (Anzahl der Neuronen und Schichten), sowie die Epochenanzahl, die Lernschritte verändern und das Ergebnis bewerten.  
Die von den Schülerinnen und Schülern leicht veränderbaren Parameter sind im Programm fett gekennzeichnet.



Anhand der Netzfehlerkurve können die Veränderungen am neuronalen Netz bewertet werden. Im obigen Diagramm ist bereits nach ca. 40 Epochen ein sehr geringer Netzfehler erkennbar und damit die Netzqualität sehr hoch.

An untenstehender Abbildung ist ein ungenügend trainiertes Netz mit einem hohen Netzfehler aufgrund einer geringen Epochenanzahl und geringen Trennung der beiden Datenbereiche zu erkennen. Die schwarzen und grünen Punkte geben die vom neuronalen Netz nicht korrekt zugeordneten Ausgangswerte an.



# **Hinweise zum Unterricht**

Vor dieser Lernsituation müssen das Verständnis und die Arbeitsweise eines neuronalen Netzes im Unterricht erarbeitet sein. Der grundlegende Aufbau eines neuronalen Netzes in der Programmsprache Python, sowie die Programmierumgebung „Tensorflow“ muss vorher erläutert werden. Diese Lernsituation ist erfahrungsgemäß gegen Ende des Themengebiets Künstliche Intelligenz im Wahlpflichtfach Digitale Transformation zu setzen. Die Lernsituation kann sehr gut in Form eines Wettbewerbs um die Suche nach den besten Parametern für eine hohe Netzqualität durchgeführt werden.

**Querverweise zu anderen Fächern / Fachrichtungen**

Der Einsatz der Künstlichen Intelligenz wird aufgrund der Vielzahl der Anwendungen in allen Fachrichtungen an einer Technikerschule Bedeutung erlangen. Dabei bietet sich eine fächerübergreifende Verbindung mit den Modulen Softwareentwicklung und Datenbanken besonders an. Dennoch kann dieses Modul auch in Fachrichtungen ohne IT-Schwerpunkt z.B. in wirtschaftlichen Fachrichtungen unterrichtet werden.

# **Quellen- und Literaturangaben**

* Neuronale Netze selbst programmieren, Tariq Rashid, O’Reilly
* Deep Learning mit Python und Keras, Francios Chollet, mitp-Verlag
* Hands-on Machine Learning, Aurelien Geron, O‘Reilly
* Fortbildungen im Rahmen der Initiative „Digitale Transformation/Wirtschaft 4.0“ an der Akademie für Lehrerfortbildung und Personalführung in Dillingen
* Staatsinstitut für Schulqualität und Bildungsforschung: Wirtschaft 4.0 an beruflichen Schulen, Handreichung, München, 2018, [http://www.isb.bayern.de/download/20645/ wirtschaft\_4.0\_an\_beruflichen\_schulen.pdf](http://www.isb.bayern.de/download/20645/%20wirtschaft_4.0_an_beruflichen_schulen.pdf), (Zugriff 02-02-2020. 17:18 MEZ)
* Grundlagenkurs <https://course.elementsofai.com/de/> vom Bundesministerium für Wirtschaft und Energie