EDS_3_Michael_Friderich

November 7, 2020

1 Erkennung von Ziffern

Semesterarbeit 3

1.1 EDS-Einführung in Data Science

Klasse: BSc INF-P-IN010, BE1, HS20/21 Dozent: Dr. Tim vor der Brück Autor: Michael Friderich Datum: 07.11.2020

1.2 Einleitung

Im heutigen Zeitalter werden die künstliche Intelligenz und das Machnine Learning immer wichtiger. Da eer Mensch nicht in der Lage ist, die heutigen Datenflutten zu bewältigen, spielen Computer eine immer wichtigere Rolle in unserem Alltag. Um künstliches Wissen aus Erfahrungen zu generieren, wird versucht das menschliche Gehirn nachzustellen. Automatisch fahrende Autos, Text-, Bild-, Spracherkennung, Frühwarnsysteme, Optimierung um nur einige Beispiele zu nennen bei welchen künstliche neuronale Netzte eingesetzt werden. In dieser Thesis wird ein neuronales Netzwerk erstellt, trainiert und geprüft ob dies die persönliche Handschrift erkennen kann.

1.3 Neuronales Netz

Wie der Name besagt, versteht man unter einem Neuronalen Netzwerk eine beliebige Anzahl miteinander verbundener Neuronen. Dieser Begriff stammt aus den Neurowissenschaften und bezieht sich auf das menschliche Gehirn. In der Informatik wird versucht dessen Strukturen als künstliches neuronales Netz zu modellieren. Dabei wird künstliches Wissen aus Erfahrung generiert. Das System lernt aus Beispielen und verallgemeinert diese nach Abaschluss der Lernphase. Um möglichst Gute Ergebnisse zu erzielen sind auch möglichst Grosse Datenmengen erforderlich.

[1]:
```
from IPython.display import HTML
from IPython.core.display import display
display(HTML('<img src="neuronal_network.png" width="600px" style="margin:␣
 ↪auto" />'))
```

Abbildung 1 - Neuronales Netzwerk

In einem neuronalen Netz werden Daten in die Input-Nodes geben und nach Berechnung an den Output-Nodes ausgegeben. Dabei sind die Daten abhängig vom zu lösenden Problem. Das Netz verfügt über einen oder mehrere Hidden-Layers. Diese bilden die Masse des künstlichen Gehirns und wirken auf die Ausgabe an den Output-Nodes.

1.4 Schriterkennung

Einer der ersten Anwendungen für die Schritenerkennung wurde für die Post entwickelt. Um die Postsortierung zu automatisieren, wollte man die Postleizahlen erkennen können. Das neuronale Netzwerk wird dabei durch eine Vielzahl an Bildern trainiert, damit die verschiedenen Handschriten zuverlässig erkennt werden. In diesem Beispiel werden wir ein Neuronales Netzwerk erstellen, dieses trainieren und aneschliessend mit der eigenen Hanschrift überprüfen. Die Trainingsdaten stammen dabei aus dem MINST-Datenbank welche 60'000 [2] Beispiele enthält.

[8]:
```
import cv2 as cv                # need to import own images
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf         # need to create the neuronal Network
```

1.4.1 Daten

Die MINST-Datenbank enthält gelabelte Daten welche wir für das überwachte Lernen benötigen. Dabei stehen uns 60'000 [2] Bilder mit Ziffern zwischen 0-9 zur Verfügung. Mit den erhaltenen Daten bilden wir einen zufälligen Trainings- sowie einen Testdatensatz. Als nächstes normalisieren wrid die Bildpunkte der erhaltenen Bilderdaten zwischen 0 und 1.

[4]:
```
# load the mnist data and split into train and test
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = tf.keras.utils.normalize(x_train, axis=1)
x_test = tf.keras.utils.normalize(x_test, axis=1)
```

1.4.2 Neuronales Netz erstellen

Nun erstellen wir das neuronale Netzwerk mit 256 Hidden-Units. Zudem legen wir die Grösse der eigenen Ziffernbilder auf 28*28 Pixel fest.

[5]:
```
# define the model
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten(input_shape=(28, 28)))    # Input size of the␣
 ↪images
model.add(tf.keras.layers.Dense(units=256, activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(units=256, activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(units=10, activation=tf.nn.softmax))

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',␣
 ↪metrics=['accuracy'])
```

1.4.3 Neuronales Netz trainieren

Nachdem das neuronale Netz erstellt wurde, trainieren wir dieses mit den Daten aus der MNIST-Datenbank. Dabei sehen wir das sich die Genauigkeit mit jedem Durchlauf verbessern. Unser neuronales Netz weisst ungefähr eine Genauigkeit von 97% auf, was für unser Netz ein akzeptabler

Wert ist. Durch den “epochs” Wert kann festegelgt werden wie oft der Datensatz durchlaufen werden soll.

[6]:
```
# train the neuronal network
model.fit(x_train, y_train, epochs=10)

loss, accuracy = model.evaluate(x_test, y_test)
print(accuracy * 100)
print(loss)

model.save('digits.model')
```

```
Epoch 1/10
1875/1875 [==============================] - 3s 1ms/step - loss: 0.2205 -
accuracy: 0.9346
Epoch 2/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.0889 -
accuracy: 0.9726
Epoch 3/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.0582 -
accuracy: 0.9811
Epoch 4/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.0425 -
accuracy: 0.9863
Epoch 5/10
1875/1875 [==============================] - 3s 1ms/step - loss: 0.0319 -
accuracy: 0.9893
Epoch 6/10
1875/1875 [==============================] - 3s 1ms/step - loss: 0.0256 -
accuracy: 0.9915
Epoch 7/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.0210 -
accuracy: 0.9931
Epoch 8/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.0170 -
accuracy: 0.9943
Epoch 9/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.0159 -
accuracy: 0.9947
Epoch 10/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.0115 -
accuracy: 0.9958
313/313 [==============================] - 0s 979us/step - loss: 0.1362 -
accuracy: 0.9748
97.47999906539917
0.13619005680084229
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-
packages\tensorflow\python\training\tracking\tracking.py:111:
```

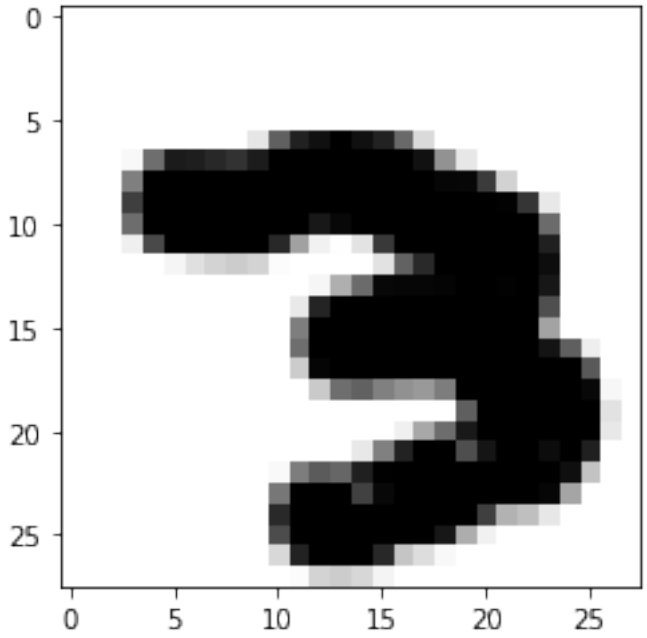
```
Model.state_updates (from tensorflow.python.keras.engine.training) is deprecated
and will be removed in a future version.
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied
automatically.
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-
packages\tensorflow\python\training\tracking\tracking.py:111: Layer.updates
(from tensorflow.python.keras.engine.base_layer) is deprecated and will be
removed in a future version.
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied
automatically.
INFO:tensorflow:Assets written to: digits.model\assets
```

1.4.4 Neuronales Netz prüfen

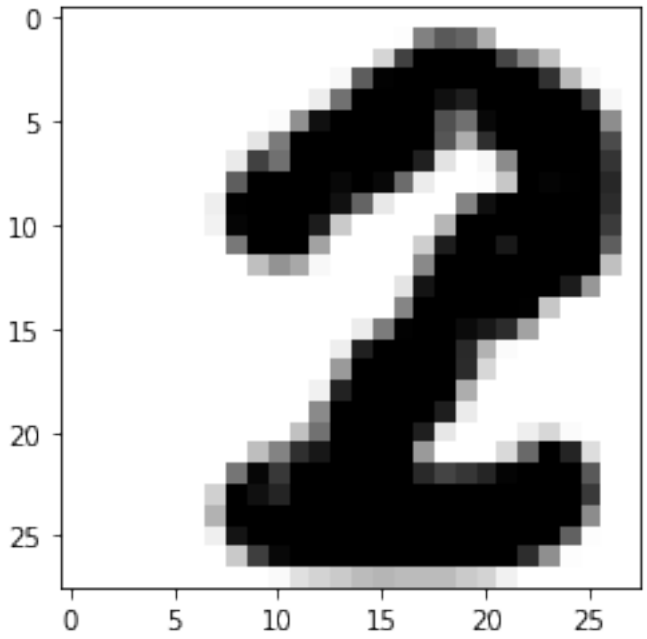
Nun da wir unser neuronales Netz trainiert haben, wollen wir dieses durch eigene Bilder überprüfen. Diese werden in schwarz/weiss Bilder umgewandelt um einen möglichst hohen Konstrast zu erhalten. Anschliessend werden diese dem trainierten neuralen Netz übergeben und wir erhalten eine Antwort mit der vermuteten Lösung.

[7]:
```
for x in range(1, 6):
    img = cv.imread(f'{x}.png')[:, :, 0]
    img = np.invert(np.array([img]))
    prediction = model.predict(img)
    print(f'The result is probably: {np.argmax(prediction)}')
    plt.imshow(img[0], cmap=plt.cm.binary)
    plt.show()
```

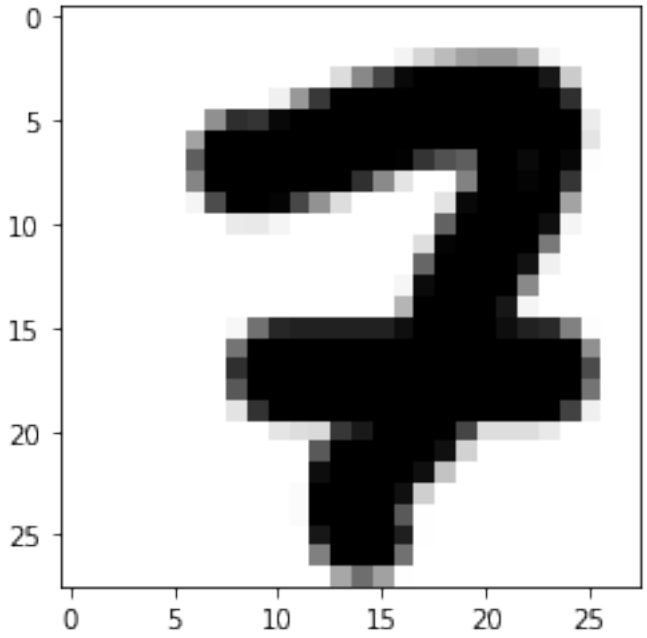
```
The result is probably: 3
```



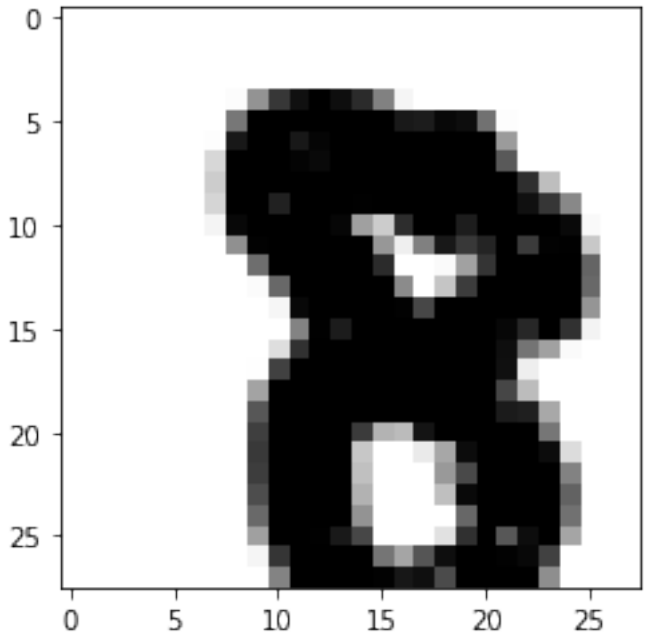

```
The result is probably: 5
```



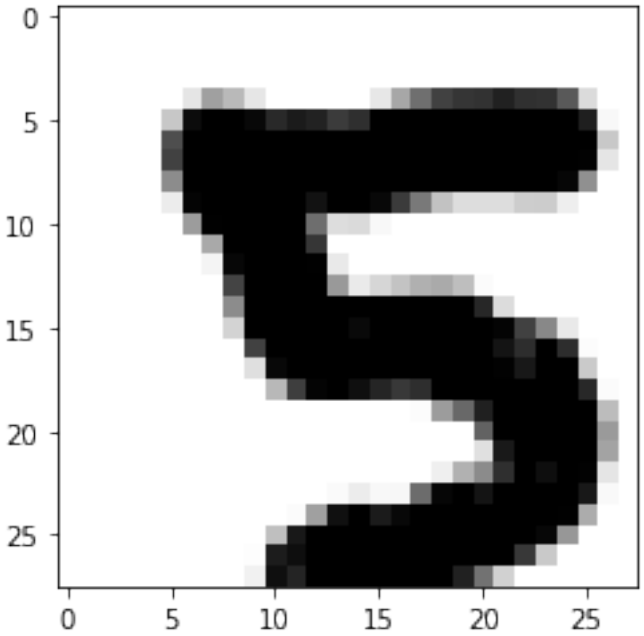

```
The result is probably: 7
```




```
The result is probably: 9
```




```
The result is probably: 5
```



1.5 Findings

Wir sehen das 3 von 5 Zahlen richtig angegeben wurden. Dies Unregelmässigkeit kann verschiedene Ursachen haben. So treten sicher bei der Synthese der Datenbilder Unregelmässigkeiten auf. Mögliche Diskrepanzen können der Scanner, Kontrastwert, die Ausschnittsgrösse und die Position der Ziffer sein. Eine weiter Möglichkeit könnte sein, die persönliche Handschrift bereits in die Trainingsadten einfliessen zu lassen. Ausserdem könnte das verändern der Hidden-Layouts Verbesserungen bei der Erkennung bewirken. Eine weitere Ursache kann der Zufal sein. In meinem Beispiel konnte ich nach Anpassen das “epchs” Wert auf 20, bereits 4 von 5 Zahlen erkennen. Was darauf hindeutet das mit genügend Training, die Genauigkeit zunimmt und eventuell alle Ziffern erkennt werden.

1.6 Quellenverzeichnis

1.6.1 Abbildungsverzeichnis

[1] Abbilidung 1: Russell, Matthew A. / Mikhail Klassen (2019): Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Instagram, GitHub, and More, 3. Aufl., Sebastopol, USA, California: O’Reilly Media. s. 98

1.6.2 Literaturverzeichnis

[1] Russell, Matthew A. / Mikhail Klassen (2019): Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Instagram, GitHub, and More, 3. Aufl., Sebastopol, USA, California: O’Reilly Media.

[2] Wikipedia, https://de.wikipedia.org/wiki/MNIST-Datenbank, Abgerufen am 07.11.2020