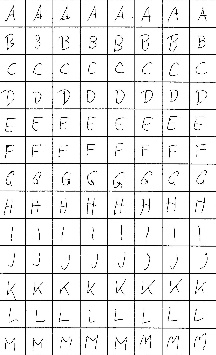
Arbeitsauftrag

Schrifterkennungs-KI

In diesem Projekt wollen wir gemeinsam einen Datensatz aus Blockbuchstaben generieren, die richtigen Labels vergeben und eine Kategorisierungs-KI damit trainieren.

Level 1

1. **Datensatz generieren**
   1. Alle Blockbuchstaben des Alphabets in den Raster A4\_raster.png schreiben (zwei Raster pro Schüler\*in , nicht bis an den Rand malen)
   2. Einscannen und genau Zuschneiden mit einem Bildbearbeitungsprogramm.
   3. Automatisiertes Ausschneiden und Abspeichern der einzelnen Buchstaben mit Buchstaben\_extrahieren.py

[Dokumentation mit Screenshots, A1 erreicht]

1. **Labels für Datensatz erstellen**
   1. Einen Ordner labelled\_letters\_nachname am KI-Server ( home/shared) erstellen. Unterordner erstellen, die den Namen des Buchstabens haben.
   2. Du kannst gerne ein Labeling-Tool verwenden (z.B. Label Studio, <https://labelstud.io/guide/get_started>)
   3. Alle Bilder in den richtigen Ordnern abspeichern.

[Dokumentation mit Screenshots, A2 erreicht]

1. **KI-Modell trainieren**

Schreibe ein python Programm, das die Trainingsdaten und Labels so vorbereitet, dass sie von tensorflow verarbeitet werden können. Tensorflow benötigt einen

numpy array N x R x C,

N … Anzahl der Bilder

R … Anzahl der Pixelreihen

C … Anzahl der Pixelspalten.

Wähle für R=C (quadratische Bilder) einen Wert zwischen 25 und 50 Pixel. Skaliere alle Bilder entsprechend und normiere die Grauwerte zwischen 0 und 1. Auch die Labels müssen in einem N x1 numpy array gespeichert werden.

Hilfestellung:

import os

os.listdir(folder) #gibt alle Dateien/Folder in einem Folder

import cv2

image = cv2.imread(filename,0) #liest ein Graustufenbild ein

resized\_img=cv2.resize(image,(final\_width,final\_height)) # skaliert Bild

arr1=np.array(pythonliste) #erzeugt einen numpy array aus einer Liste

np.save(“arr1.np“,arr1) #speichert numpy array ab

import os  
import cv2  
import numpy as np  
  
  
def prepare\_data(folder, image\_size=28):  
 images = []  
 labels = []  
 label\_dict = {} # Dictionary zur Speicherung der Label-Zuordnung  
  
 # Durch alle Unterordner iterieren  
 for idx, subfolder in enumerate(sorted(os.listdir(folder))):  
 subfolder\_path = os.path.join(folder, subfolder)  
  
 if os.path.isdir(subfolder\_path):  
 label\_dict[idx] = subfolder # Speichere die Label-Zuordnung  
  
 for filename in sorted(os.listdir(subfolder\_path)):  
 filepath = os.path.join(subfolder\_path, filename)  
  
 if os.path.isfile(filepath):  
 # Bild einlesen und in Graustufen konvertieren  
 image = cv2.imread(filepath, cv2.IMREAD\_GRAYSCALE)  
  
 # Bild skalieren auf das richtige Format  
 resized\_img = cv2.resize(image, (image\_size, image\_size))  
  
 # Normalisieren (Werte zwischen 0 und 1)  
 normalized\_img = resized\_img / 255.0  
  
 # Falls CNN: Kanal hinzufügen (Height, Width, 1)  
 normalized\_img = normalized\_img.reshape(image\_size, image\_size, 1)  
  
 images.append(normalized\_img)  
 labels.append(idx) # Verwenden des Index als Label  
  
 # Konvertiere in NumPy-Arrays  
 images\_array = np.array(images, dtype=np.float32)  
 labels\_array = np.array(labels, dtype=np.int32) # Flaches Array  
  
 # Speichern der Dateien  
 np.save("images.npy", images\_array)  
 np.save("labels.npy", labels\_array)  
 np.save("label\_dict.npy", label\_dict) # Speichern der Label-Zuordnung  
  
 print(f"Daten gespeichert: {len(images)} Bilder verarbeitet.")  
 print(f"Shape der Bilder: {images\_array.shape}") # Debug-Info  
 print(f"Shape der Labels: {labels\_array.shape}") # Debug-Info  
 print(f"Label-Zuordnung: {label\_dict}")  
  
 return images\_array, labels\_array, label\_dict  
  
  
folder\_path = "C:/1BHEL\_HTLInn/KISY/Python/Digit\_Erkennen/BIGDATASET"  
images, labels, label\_dict = prepare\_data(folder\_path, image\_size=28)

1. Schreibe ein Programm zum Trainieren eines neuronalen Netzes mit tensorflow. Das Programm liest zuerst die Trainingsdaten und Labels ein und spaltet diese zufällig auf in einen Testdatensatz (ca. 10%) und einen Trainingsdatensatz (ca. 90%). Erstelle ein sequentielles feedforward neuronales Netzwerk-Modell und trainiere das Modell mit den Daten. Orientiere dich dabei am [Keras Fashion Tutorial](https://www.tensorflow.org/tutorials/keras/classification).

import numpy as np  
from sklearn.model\_selection import train\_test\_split  
from model import create\_model # Importiere das Modell aus model.py  
  
# Pfad zum Hauptordner mit den Unterordnern A-Z  
folder\_path = r"C:/1BHEL\_HTLInn/KISY/Python/Digit\_Erkennen"  
img\_size = 28 # Skalierung auf 28x28 Pixel  
  
# Lade und verarbeite die Bilder und Labels  
images\_array = np.load("images.npy")  
labels\_array = np.load("labels.npy")  
  
# Normalisiere die Bilder  
images\_array = images\_array.reshape(-1, 28, 28, 1) # Anpassung auf die Eingabeform für CNNs  
images\_array = images\_array.astype('float32') # Um sicherzustellen, dass die Werte als float32 vorliegen  
labels\_array = labels\_array.astype('int32') # Um sicherzustellen, dass die Labels als Integer vorliegen  
  
# Splitte die Daten in Trainings- und Testdaten  
x\_train, x\_test, y\_train, y\_test = train\_test\_split(images\_array, labels\_array, test\_size=0.1, random\_state=42)  
  
# Erstelle das Modell  
model = create\_model(input\_shape=(28, 28, 1), num\_classes=26)  
  
# Trainiere das Modell  
model.fit(x\_train, y\_train, epochs=30, validation\_data=(x\_test, y\_test))  
  
# Teste das Modell  
test\_loss, test\_acc = model.evaluate(x\_test, y\_test, verbose=2)  
print(f"Genauigkeit: {test\_acc \* 100:.2f}%")  
  
# Speichern des Modells  
model.save('model.h5')  
print(f"Modell gespeichert")

1. Überprüfe dein Netzwerk auf den Testdaten durch manuelle Überprüfung der Korrektheit der vorhergesagten Labels. Schreibe dazu eine Funktion, die aus der maximalen Wahrscheinlichkeit den vorhergesagten Buchstaben rückberechnet. Wende die Funktion auf alle Testdaten an.

import numpy as np  
from tensorflow.keras.models import load\_model  
import matplotlib.pyplot as plt  
  
# Lade das trainierte Modell  
model = load\_model("model.h5")  
  
# Lade die Testdaten  
images = np.load("images.npy") # Bilddaten  
labels = np.load("labels.npy") # Echte Labels  
  
# Wähle ein zufälliges Testbild  
index = np.random.randint(0, len(images))  
test\_image = images[index]  
true\_label = labels[index]  
  
# Vorhersage machen  
prediction = model.predict(test\_image.reshape(1, \*test\_image.shape))  
predicted\_label = np.argmax(prediction)  
  
# Bild und Vorhersage anzeigen  
plt.imshow(test\_image.squeeze(), cmap="gray")  
plt.title(f"Vorhersage: {chr(predicted\_label + ord('A'))} | Wahres Label: {chr(true\_label + ord('A'))}")  
plt.axis("off")  
plt.show()

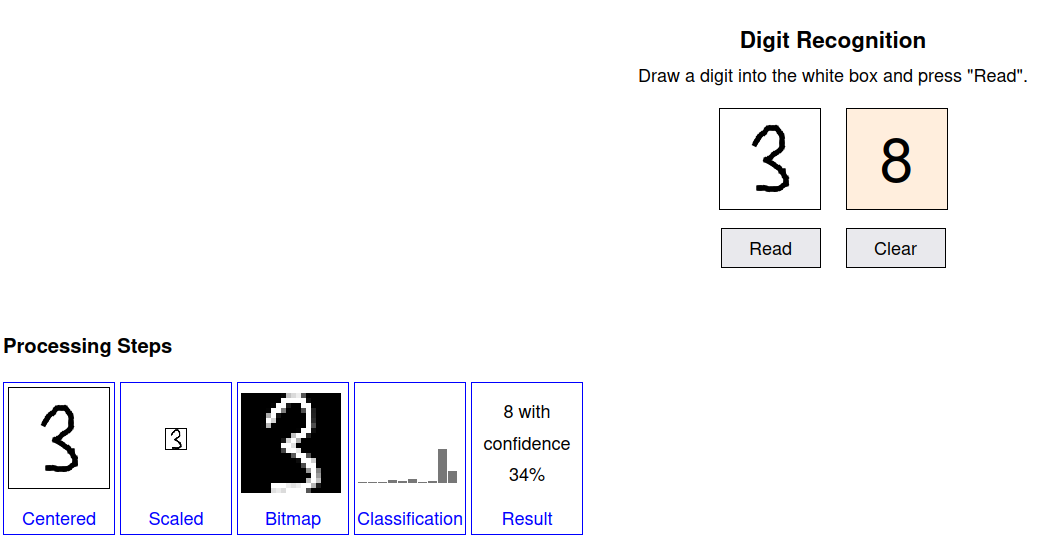
1. Speichere dein Modell mit der Funktion model.save(“dateiname.keras“)



1. Erstelle/Nutze eine graphische Oberfläche (z.B. tkinter), mit der eine Benutzerin Buchstaben eingeben kann, welche mit deinem KI-Modell erkannt und dargestellt werden sollen. Optional kannst du auch Informationen zur Vorhersage visualisieren, z.B. so:

Ein Bild, das Text, Screenshot, Design enthält.

KI-generierte Inhalte können fehlerhaft sein.



1. [Dokumentation mit Screenshots, A7 erreicht]

Level 2:

1. Vergrößere deinen Datensatz durch zufällige kleine Drehungen und zufällige kleine Skalierungen der Buchstaben, trainiere ein neues Modell und vergleiche die Ergebnisse mit dem alten Modell. (siehe <https://www.tensorflow.org/tutorials/images/data_augmentation> )
2. Nutze den KI-Server und dessen GPU für das Training des Modells. Stelle mit tf\_gpu\_test.py sicher, dass die GPU erkannt und verwendet wird.
3. Nutze ein CNN Modell für bessere Resultate. (siehe <https://www.tensorflow.org/tutorials/images/classification> )