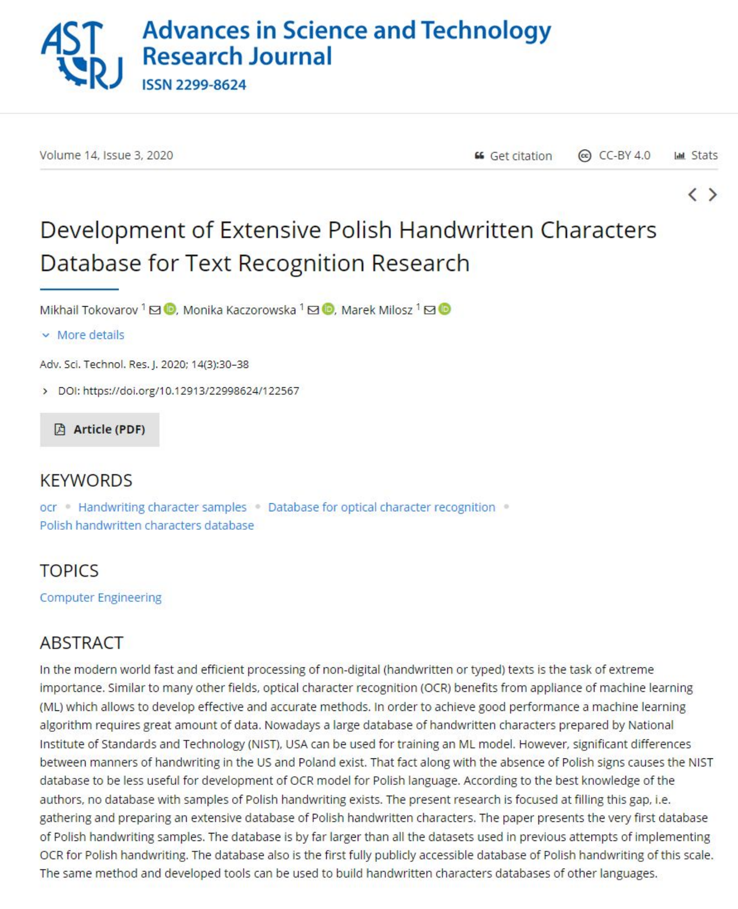
Polish handwritten character recognition

Marcin Ciesiul

Data documentation

http://www.astrj.com/Development-of-Extensive-Polish-Handwritten-Characters-Database-for-Text-Recognition,122567,0,2.html



Advances in Science and Technology Research Journal
ISSN 2299-8624

Volume 14, Issue 3, 2020

Development of Extensive Polish Handwritten Characters Database for Text Recognition Research

Mikhail Tokovarov [1], Monika Kaczorowska [1], Marek Milosz [1]

Adv. Sci. Technol. Res. J. 2020; 14(3):30–38

DOI: https://doi.org/10.12913/22998624/122567

KEYWORDS

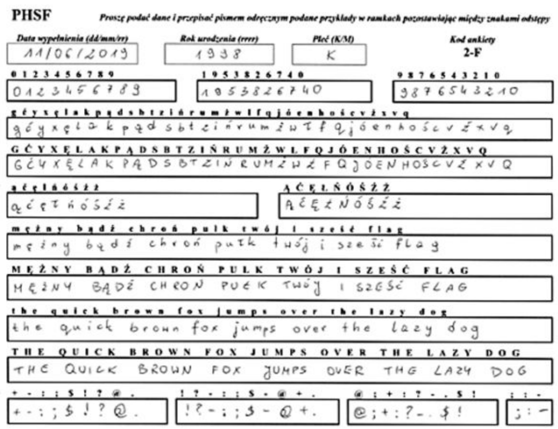
ocr • Handwriting character samples • Database for optical character recognition • Polish handwritten characters database

TOPICS

Computer Engineering

ABSTRACT

In the modern world fast and efficient processing of non-digital (handwritten or typed) texts is the task of extreme importance. Similar to many other fields, optical character recognition (OCR) benefits from appliance of machine learning (ML) which allows to develop effective and accurate methods. In order to achieve good performance a machine learning algorithm requires great amount of data. Nowadays a large database of handwritten characters prepared by National Institute of Standards and Technology (NIST), USA can be used for training an ML model. However, significant differences between manners of handwriting in the US and Poland exist. That fact along with the absence of Polish signs causes the NIST database to be less useful for development of OCR model for Polish language. According to the best knowledge of the authors, no database with samples of Polish handwriting exists. The present research is focused at filling this gap, i.e. gathering and preparing an extensive database of Polish handwritten characters. The paper presents the very first database of Polish handwriting samples. The database is by far larger than all the datasets used in previous attempts of implementing OCR for Polish handwriting. The database also is the first fully publicly accessible database of Polish handwriting of this scale. The same method and developed tools can be used to build handwritten characters databases of other languages.



PHSF *Proszę podać dane i przepisać pismem odręcznym podane przykłady w ramkach pozostawiając między znakami odstępy*

Data wypełnienia (dd/mm/rr)	Rok urodzenia (rrrr)	Płeć (K/M)	Kod ankiety
11/06/2019	1938	K	2-F

0 1 2 3 4 5 6 7 8 9 — 0 1 2 3 4 5 6 7 8 9

1 9 5 3 8 2 6 7 4 0 — 1 9 5 3 8 2 6 7 4 0

9 8 7 6 5 4 3 2 1 0 — 9 8 7 6 5 4 3 2 1 0

g ć y x ę ł a k p ą d s b t z i ń r u m ż w l f q j ó e n h o ś c v ź x v q

g ć y x ę ł a k p ą d s b t z i ń r u m ż w ł f q j ó e n h o ś c v ź x v q

G Ć Y X Ę L A K P Ą D S B T Z I Ń R U M Ż W L F Q J Ó E N H O Ś C V Ź X V Q

G Ć Y X Ę Ł A K P Ą D S B T Z I Ń R U M Ż W Ł F Q J Ó E N H O Ś C V Ź X V Q

ą ć ę ł ń ó ś ź ż — ą ć ę ł ń ó ś ź ż

Ą Ć Ę Ł Ń Ó Ś Ź Ż — Ą Ć Ę Ł Ń Ó Ś Ź Ż

mężny bądź chroń pułk twój i sześć flag

mężny bądź chroń pułk twój i sześć flag

MĘŻNY BĄDŹ CHROŃ PULK TWÓJ I SZEŚĆ FLAG

MĘŻNY BĄDŹ CHRON PUŁK TWÓJ I SZEŚĆ FLAG

the quick brown fox jumps over the lazy dog

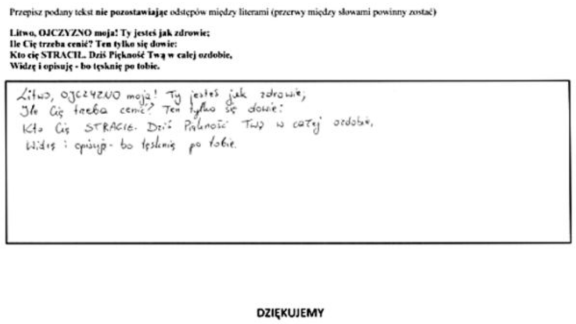
the quick brown fox jumps over the lazy dog

THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

+ - : ; $! ? @ .	! ? - : ; $ - @ + .	@ ; + : ? - . $!	; : -
+ - : ; $! ? @ .	! ? - : ; $ - @ + .	@ ; + : ? - . $!	; : -

Fig. 1. Filled Polish handwritten sample form (first side)



Przepisz podany tekst **nie pozostawiając** odstępów między literami (przerwy między słowami powinny zostać)

Litwo, OJCZYZNO moja! Ty jesteś jak zdrowie;
Ile Cię trzeba cenić? Ten tylko się dowie:
Kto cię STRACIŁ. Dziś Piękność Twą w całej ozdobie,
Widzę i opisuję - bo tęsknię po tobie.

Litwo, OJCZYZNO moja! Ty jestes jak zdrowie;
Ile Cię trzeba cenić? Ten tylko się dowie:
Kto Cię STRACIŁ. Dziś Piękność Twą w całej ozdobie,
Widzę i opisuję - bo tęsknię po tobie.

DZIĘKUJEMY

Fig. 2. Filled Polish handwritten sample form (second side)

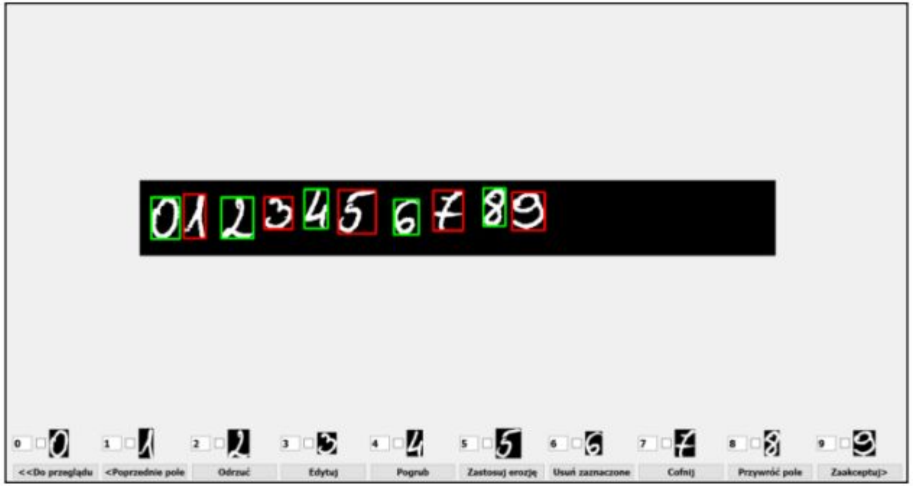
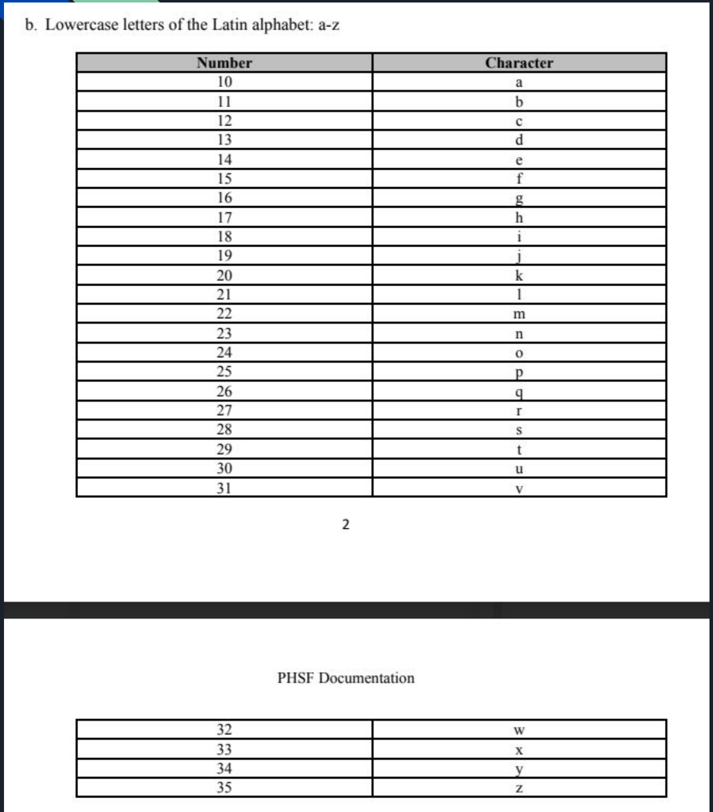


Fig. 3. GUI of the dedicated application

Data documentation

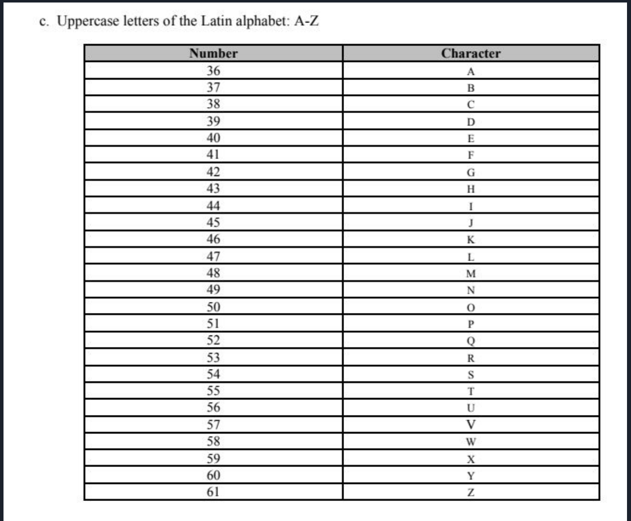
[...] Nowadays a large database of handwritten characters prepared by National Institute of Standards and Technology (NIST), USA can be used for training an ML model. **However, significant differences between manners of handwriting in the US and Poland exist. That fact along with the absence of Polish signs causes the NIST database to be less useful for development of OCR model for Polish language.** According to the best knowledge of the authors, no database with samples of Polish handwriting exists. [...]

Data documentation



b. Lowercase letters of the Latin alphabet: a-z

Number	Character
10	a
11	b
12	c
13	d
14	e
15	f
16	g
17	h
18	i
19	j
20	k
21	l
22	m
23	n
24	o
25	p
26	q
27	r
28	s
29	t
30	u
31	v
32	w
33	x
34	y
35	z

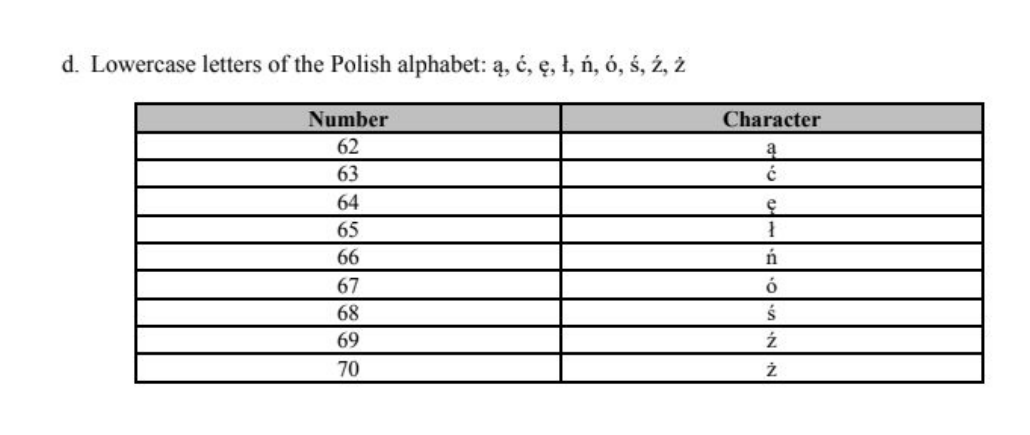


c. Uppercase letters of the Latin alphabet: A-Z

Number	Character
36	A
37	B
38	C
39	D
40	E
41	F
42	G
43	H
44	I
45	J
46	K
47	L
48	M
49	N
50	O
51	P
52	Q
53	R
54	S
55	T
56	U
57	V
58	W
59	X
60	Y
61	Z

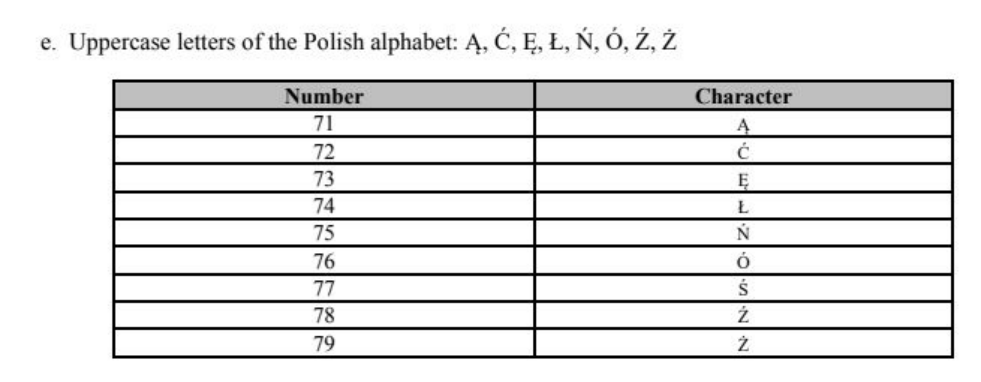
- 70 Klassen
- 5000 Bilder pro Klasse
- 350k Bilder waren das Basis für das Modell

Data documentation



d. Lowercase letters of the Polish alphabet: ą, ć, ę, ł, ń, ó, ś, ź, ż

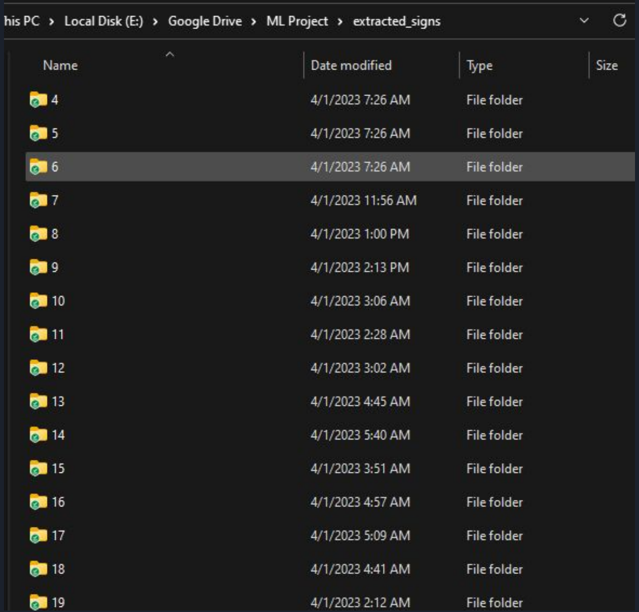
Number	Character
62	ą
63	ć
64	ę
65	ł
66	ń
67	ó
68	ś
69	ź
70	ż



e. Uppercase letters of the Polish alphabet: Ą, Ć, Ę, Ł, Ń, Ó, Ź, Ż

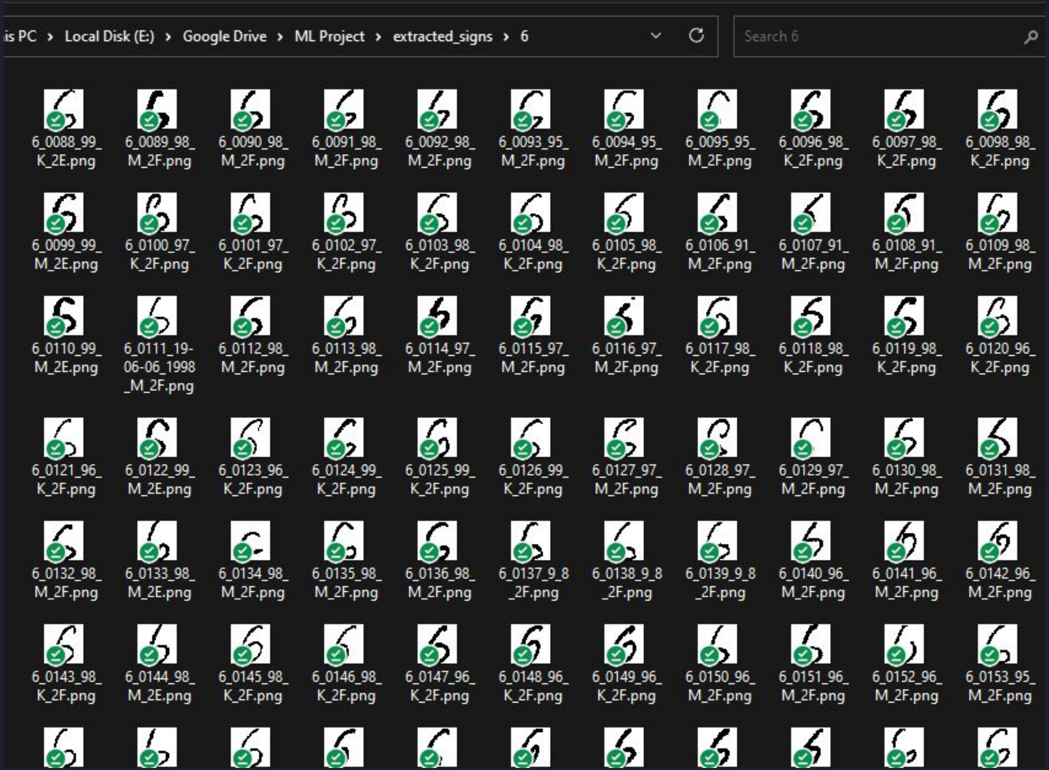
Number	Character
71	Ą
72	Ć
73	Ę
74	Ł
75	Ń
76	Ó
77	Ś
78	Ź
79	Ż

Data documentation



his PC › Local Disk (E:) › Google Drive › ML Project › extracted_signs

Name	Date modified	Type	Size
4	4/1/2023 7:26 AM	File folder	
5	4/1/2023 7:26 AM	File folder	
6	4/1/2023 7:26 AM	File folder	
7	4/1/2023 11:56 AM	File folder	
8	4/1/2023 1:00 PM	File folder	
9	4/1/2023 2:13 PM	File folder	
10	4/1/2023 3:06 AM	File folder	
11	4/1/2023 2:28 AM	File folder	
12	4/1/2023 3:02 AM	File folder	
13	4/1/2023 4:45 AM	File folder	
14	4/1/2023 5:40 AM	File folder	
15	4/1/2023 3:51 AM	File folder	
16	4/1/2023 4:57 AM	File folder	
17	4/1/2023 5:09 AM	File folder	
18	4/1/2023 4:41 AM	File folder	
19	4/1/2023 2:12 AM	File folder	



is PC › Local Disk (E:) › Google Drive › ML Project › extracted_signs › 6

Search 6

6_0088_99_K_2E.png 6_0089_98_M_2F.png 6_0090_98_M_2F.png 6_0091_98_M_2F.png 6_0092_98_M_2F.png 6_0093_95_M_2F.png 6_0094_95_M_2F.png 6_0095_95_M_2F.png 6_0096_98_K_2F.png 6_0097_98_K_2F.png 6_0098_98_K_2F.png

6_0099_99_M_2E.png 6_0100_97_K_2F.png 6_0101_97_K_2F.png 6_0102_97_K_2F.png 6_0103_98_K_2F.png 6_0104_98_K_2F.png 6_0105_98_K_2F.png 6_0106_91_M_2F.png 6_0107_91_M_2F.png 6_0108_91_M_2F.png 6_0109_98_M_2F.png

6_0110_99_M_2E.png 6_0111_19-06-06_1998_M_2F.png 6_0112_98_M_2F.png 6_0113_98_M_2F.png 6_0114_97_M_2F.png 6_0115_97_M_2F.png 6_0116_97_M_2F.png 6_0117_98_K_2F.png 6_0118_98_K_2F.png 6_0119_98_K_2F.png 6_0120_96_K_2F.png

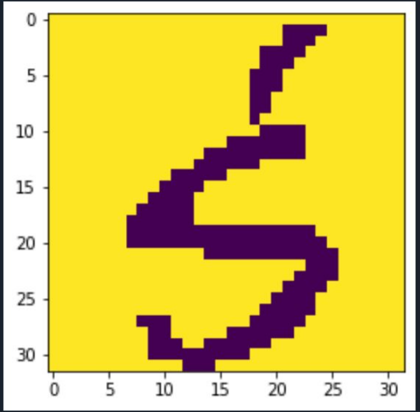
6_0121_96_K_2F.png 6_0122_99_M_2E.png 6_0123_96_K_2F.png 6_0124_99_K_2F.png 6_0125_99_K_2F.png 6_0126_99_K_2F.png 6_0127_97_M_2F.png 6_0128_97_M_2F.png 6_0129_97_M_2F.png 6_0130_98_M_2F.png 6_0131_98_M_2F.png

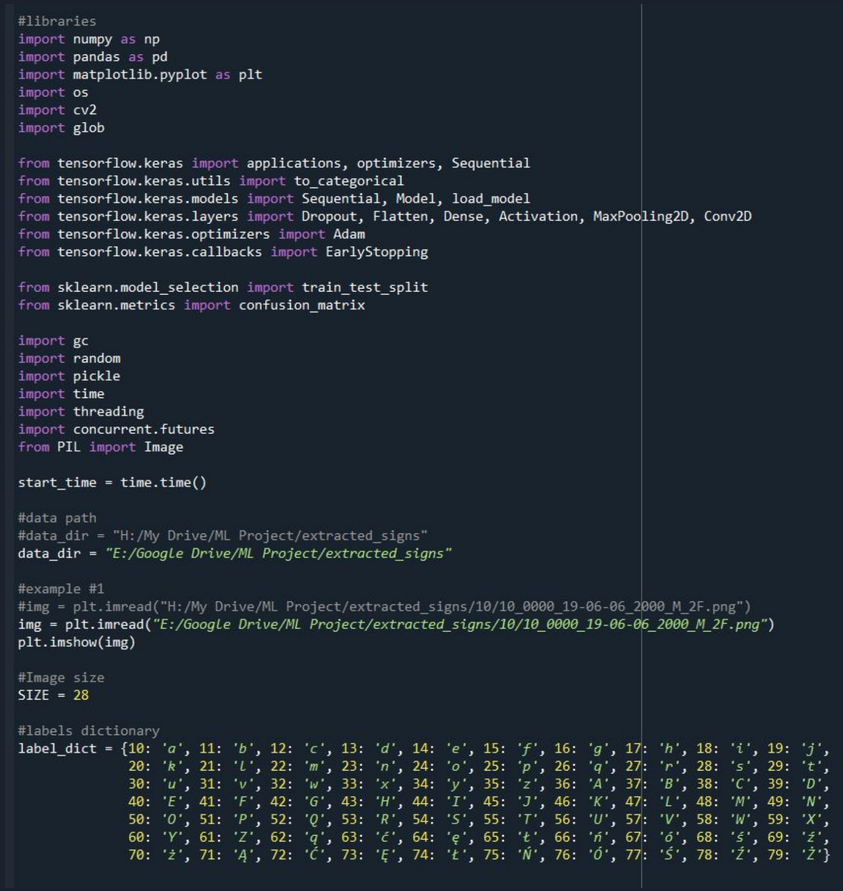
6_0132_98_M_2F.png 6_0133_98_M_2E.png 6_0134_98_M_2F.png 6_0135_98_M_2F.png 6_0136_98_M_2F.png 6_0137_9_8_2F.png 6_0138_9_8_2F.png 6_0139_9_8_2F.png 6_0140_96_M_2F.png 6_0141_96_M_2F.png 6_0142_96_M_2F.png

6_0143_98_K_2F.png 6_0144_98_M_2E.png 6_0145_98_K_2F.png 6_0146_98_K_2F.png 6_0147_96_K_2F.png 6_0148_96_K_2F.png 6_0149_96_K_2F.png 6_0150_96_M_2F.png 6_0151_96_M_2F.png 6_0152_96_M_2F.png 6_0153_95_M_2F.png

The Code

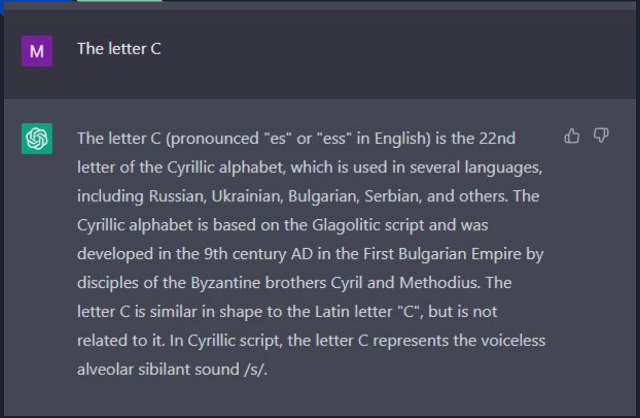
- Libraries
- Data path
- Exemplary pic before resizing
- SIZE variable
- Dictionary of labels





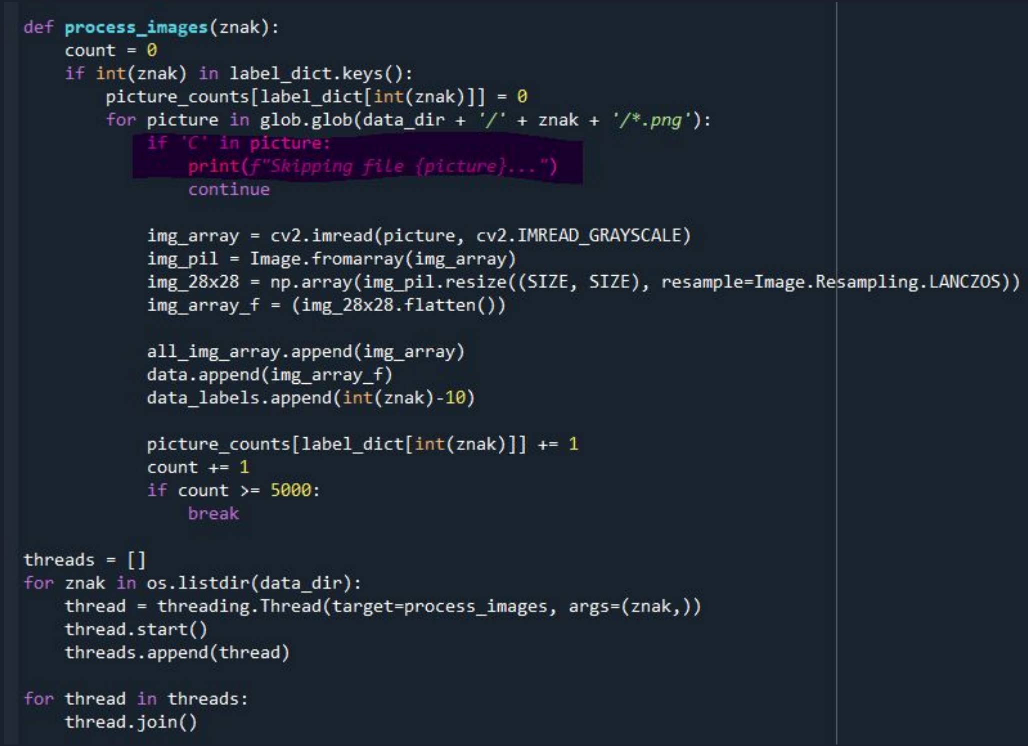
```python
#libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
import cv2
import glob

from tensorflow.keras import applications, optimizers, Sequential
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential, Model, load_model
from tensorflow.keras.layers import Dropout, Flatten, Dense, Activation, MaxPooling2D, Conv2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

import gc
import random
import pickle
import time
import threading
import concurrent.futures
from PIL import Image

start_time = time.time()

#data path
#data_dir = "H:/My Drive/ML Project/extracted_signs"
data_dir = "E:/Google Drive/ML Project/extracted_signs"

#example #1
#img = plt.imread("H:/My Drive/ML Project/extracted_signs/10/10_0000_19-06-06_2000_M_2F.png")
img = plt.imread("E:/Google Drive/ML Project/extracted_signs/10/10_0000_19-06-06_2000_M_2F.png")
plt.imshow(img)

#Image size
SIZE = 28

#labels dictionary
label_dict = {10: 'a', 11: 'b', 12: 'c', 13: 'd', 14: 'e', 15: 'f', 16: 'g', 17: 'h', 18: 'i', 19: 'j',
              20: 'k', 21: 'l', 22: 'm', 23: 'n', 24: 'o', 25: 'p', 26: 'q', 27: 'r', 28: 's', 29: 't',
              30: 'u', 31: 'v', 32: 'w', 33: 'x', 34: 'y', 35: 'z', 36: 'A', 37: 'B', 38: 'C', 39: 'D',
              40: 'E', 41: 'F', 42: 'G', 43: 'H', 44: 'I', 45: 'J', 46: 'K', 47: 'L', 48: 'M', 49: 'N',
              50: 'O', 51: 'P', 52: 'Q', 53: 'R', 54: 'S', 55: 'T', 56: 'U', 57: 'V', 58: 'W', 59: 'X',
              60: 'Y', 61: 'Z', 62: 'ą', 63: 'ć', 64: 'ę', 65: 'ł', 66: 'ń', 67: 'ó', 68: 'ś', 69: 'ź',
              70: 'ż', 71: 'Ą', 72: 'Ć', 73: 'Ę', 74: 'Ł', 75: 'Ń', 76: 'Ó', 77: 'Ś', 78: 'Ź', 79: 'Ż'}
```

The Code - image reading function



M The letter C

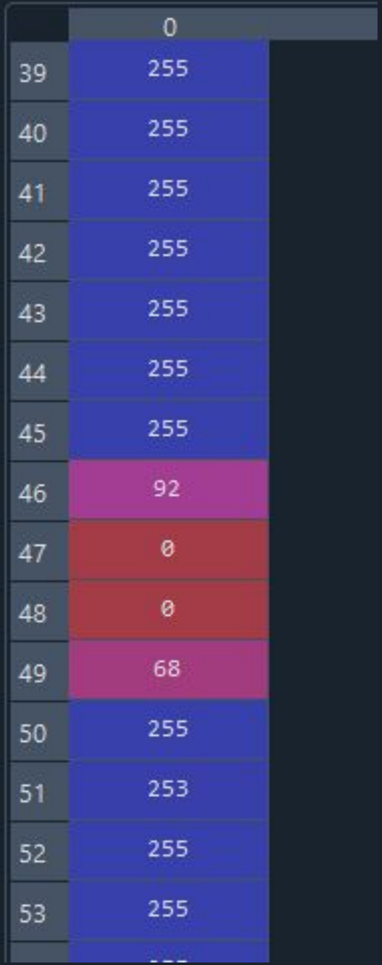
The letter C (pronounced "es" or "ess" in English) is the 22nd letter of the Cyrillic alphabet, which is used in several languages, including Russian, Ukrainian, Bulgarian, Serbian, and others. The Cyrillic alphabet is based on the Glagolitic script and was developed in the 9th century AD in the First Bulgarian Empire by disciples of the Byzantine brothers Cyril and Methodius. The letter C is similar in shape to the Latin letter "C", but is not related to it. In Cyrillic script, the letter C represents the voiceless alveolar sibilant sound /s/.



```python
def process_images(znak):
    count = 0
    if int(znak) in label_dict.keys():
        picture_counts[label_dict[int(znak)]] = 0
        for picture in glob.glob(data_dir + '/' + znak + '/*.png'):
            if 'C' in picture:
                print(f"Skipping file {picture}...")
                continue

            img_array = cv2.imread(picture, cv2.IMREAD_GRAYSCALE)
            img_pil = Image.fromarray(img_array)
            img_28x28 = np.array(img_pil.resize((SIZE, SIZE), resample=Image.Resampling.LANCZOS))
            img_array_f = (img_28x28.flatten())

            all_img_array.append(img_array)
            data.append(img_array_f)
            data_labels.append(int(znak)-10)

            picture_counts[label_dict[int(znak)]] += 1
            count += 1
            if count >= 5000:
                break

threads = []
for znak in os.listdir(data_dir):
    thread = threading.Thread(target=process_images, args=(znak,))
    thread.start()
    threads.append(thread)

for thread in threads:
    thread.join()
```

- read in grayscale
- resize to 28x28 using LANCZOS
- convert to numpy array
- 2D into 1D

+ threading



	0
39	255
40	255
41	255
42	255
43	255
44	255
45	255
46	92
47	0
48	0
49	68
50	255
51	253
52	255
53	255

- print(picture_counts)

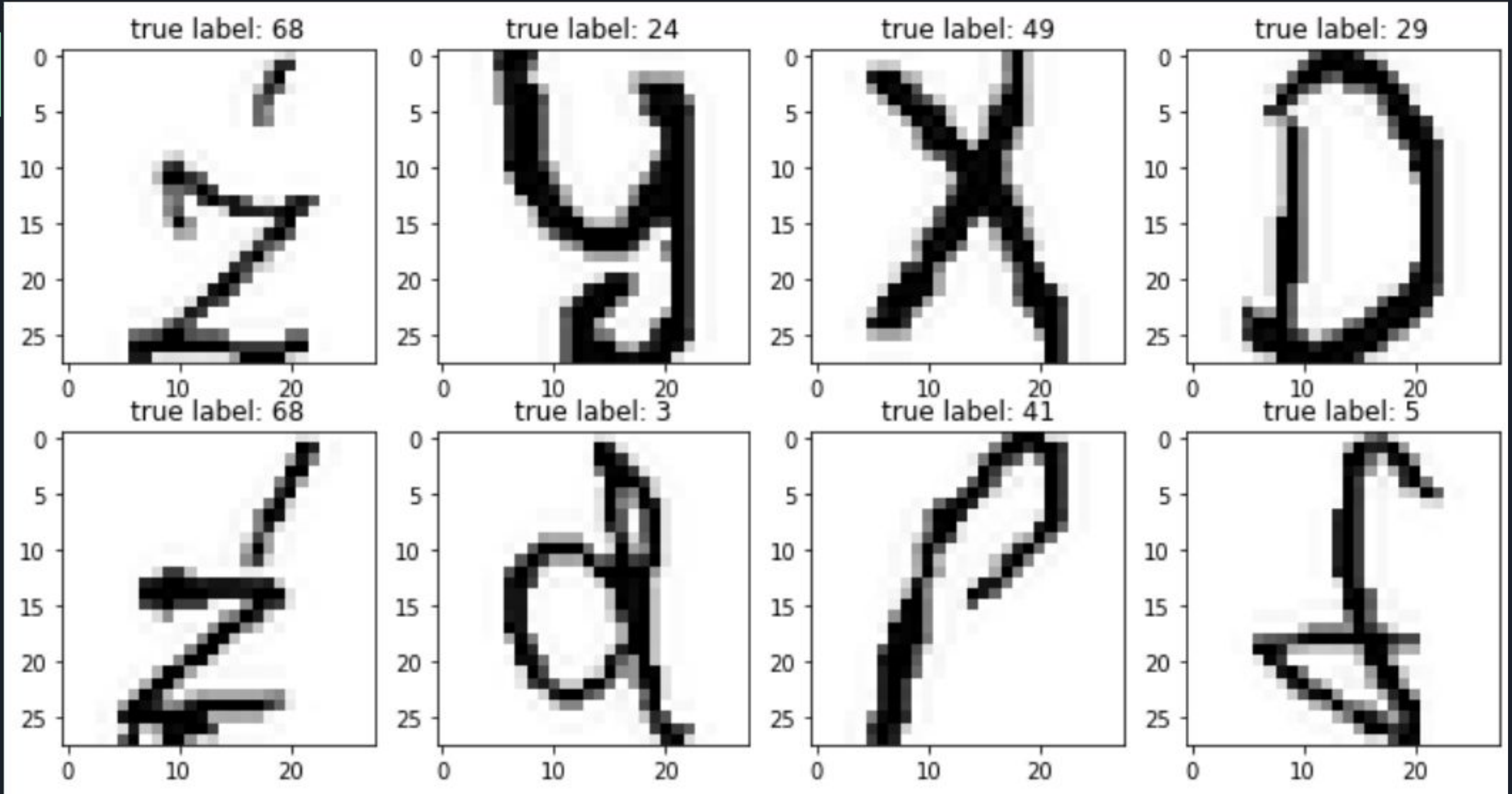
```
{'a': 5000, 'b': 5000, 'c': 5000, 'd': 5000, 'e': 5000, 'f': 5000, 'g': 5000,
'h': 5000, 'i': 5000, 'j': 5000, 'k': 5000, 'l': 5000, 'm': 5000, 'n': 5000,
'o': 5000, 'p': 5000, 'q': 5000, 'r': 5000, 's': 5000, 't': 5000, 'u': 5000,
'v': 5000, 'w': 5000, 'x': 5000, 'y': 5000, 'z': 5000, 'A': 5000, 'B': 5000,
'C': 5000, 'D': 5000, 'E': 5000, 'F': 5000, 'G': 5000, 'H': 5000, 'I':
5000, 'J': 5000, 'K': 5000, 'L': 5000, 'M': 5000, 'N': 5000, 'O': 5000,
'P': 5000, 'Q': 5000, 'R': 5000, 'S': 5000, 'T': 5000, 'U': 5000, 'V':
5000, 'W': 5000, 'X': 5000, 'Y': 5000, 'Z': 5000, 'ą': 5000, 'ć': 5000,
'ę': 5000, 'ł': 5000, 'ń': 5000, 'ó': 5000, 'ś': 5000, 'ź': 5000, 'ż': 5000,
'Ą': 5000, 'Ć': 5000, 'Ę': 5000, 'Ł': 5000, 'Ń': 5000, 'Ó': 5000, 'Ś':
5000, 'Ź': 5000, 'Ż': 5000}
```

The Code

Data split into:

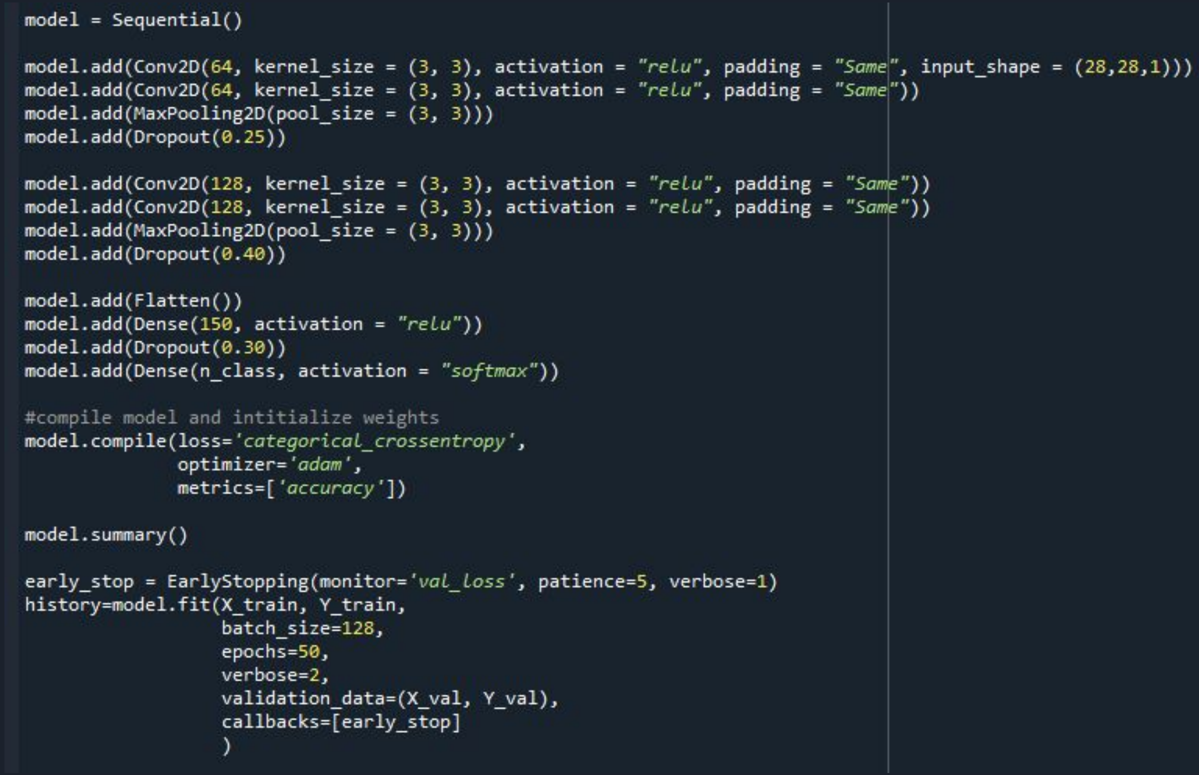
- train 60%
- test 20%
- validation 20%

1. Normalize / 255
2. One-hot encoding of the labels
3. Reshape into 4D arrays for CNN, ex. (X_train.shape[0], 28, 28, 1)


true label: 68
true label: 24
true label: 49
true label: 29
true label: 68
true label: 3
true label: 41
true label: 5
0
5
10
15
20
25
0
10
20

The Code

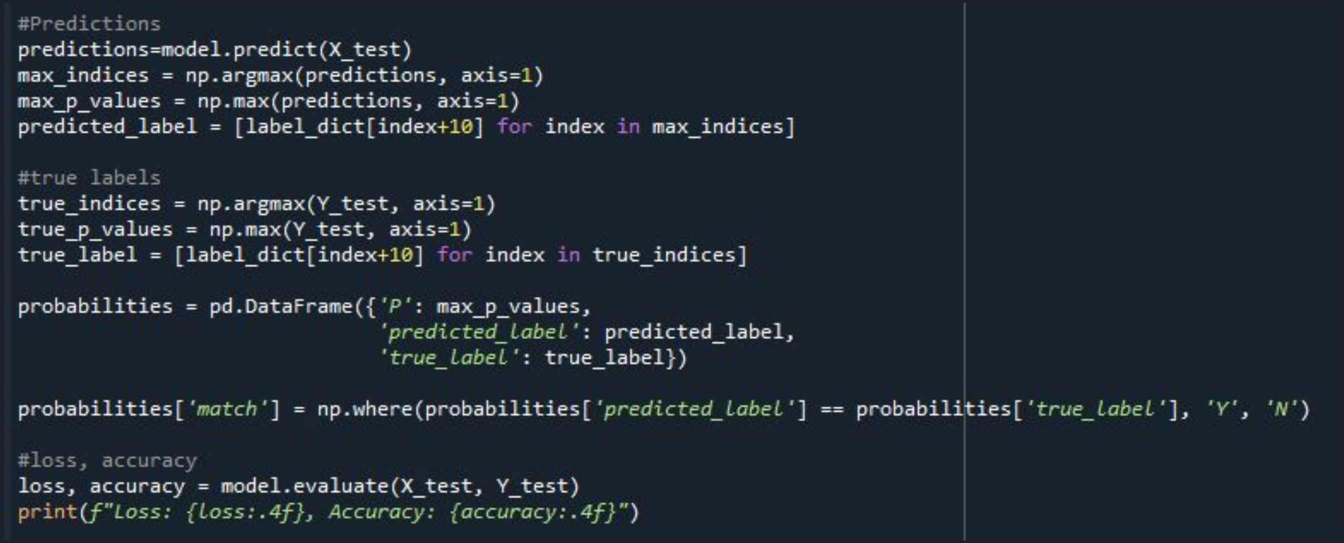
- 12 layers
- 50 epochs
- EarlyStopping



```
model = Sequential()

model.add(Conv2D(64, kernel_size = (3, 3), activation = "relu", padding = "Same", input_shape = (28,28,1)))
model.add(Conv2D(64, kernel_size = (3, 3), activation = "relu", padding = "Same"))
model.add(MaxPooling2D(pool_size = (3, 3)))
model.add(Dropout(0.25))

model.add(Conv2D(128, kernel_size = (3, 3), activation = "relu", padding = "Same"))
model.add(Conv2D(128, kernel_size = (3, 3), activation = "relu", padding = "Same"))
model.add(MaxPooling2D(pool_size = (3, 3)))
model.add(Dropout(0.40))

model.add(Flatten())
model.add(Dense(150, activation = "relu"))
model.add(Dropout(0.30))
model.add(Dense(n_class, activation = "softmax"))

#compile model and intitialize weights
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

model.summary()

early_stop = EarlyStopping(monitor='val_loss', patience=5, verbose=1)
history=model.fit(X_train, Y_train,
                  batch_size=128,
                  epochs=50,
                  verbose=2,
                  validation_data=(X_val, Y_val),
                  callbacks=[early_stop]
                  )
```

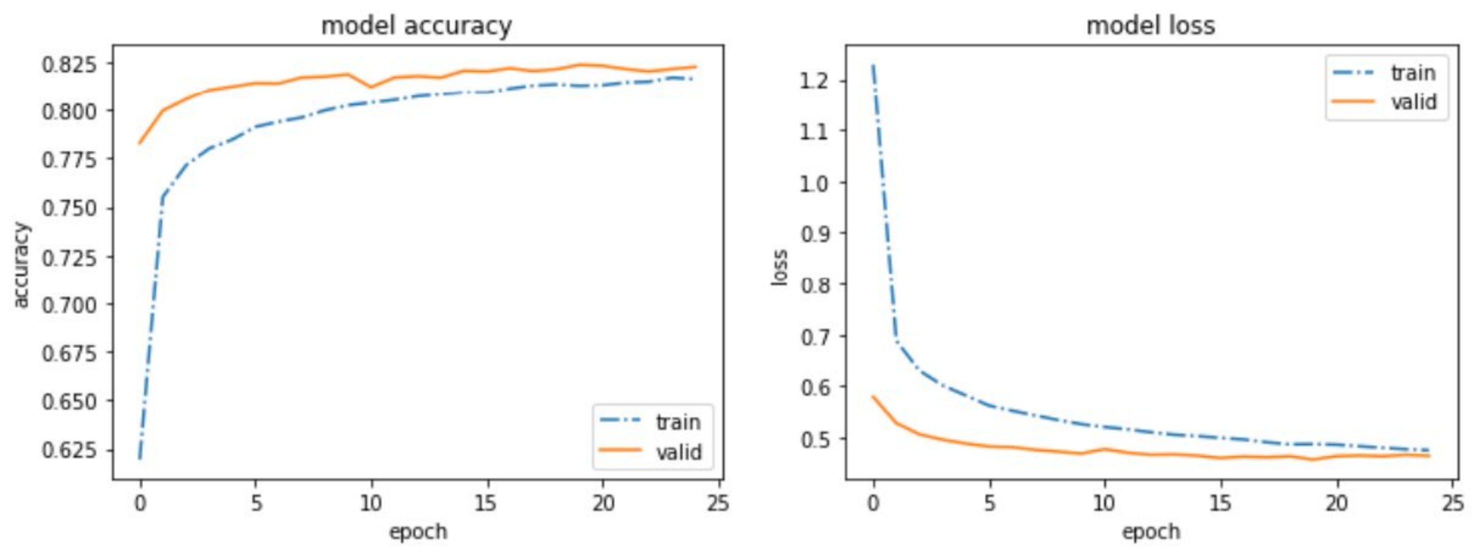
The Code

- Predictions
- Labels
- Probabilities DataFrame



```
#Predictions
predictions=model.predict(X_test)
max_indices = np.argmax(predictions, axis=1)
max_p_values = np.max(predictions, axis=1)
predicted_label = [label_dict[index+10] for index in max_indices]

#true labels
true_indices = np.argmax(Y_test, axis=1)
true_p_values = np.max(Y_test, axis=1)
true_label = [label_dict[index+10] for index in true_indices]

probabilities = pd.DataFrame({'P': max_p_values,
                              'predicted_label': predicted_label,
                              'true_label': true_label})

probabilities['match'] = np.where(probabilities['predicted_label'] == probabilities['true_label'], 'Y', 'N')

#loss, accuracy
loss, accuracy = model.evaluate(X_test, Y_test)
print(f"Loss: {loss:.4f}, Accuracy: {accuracy:.4f}")
```

The Results



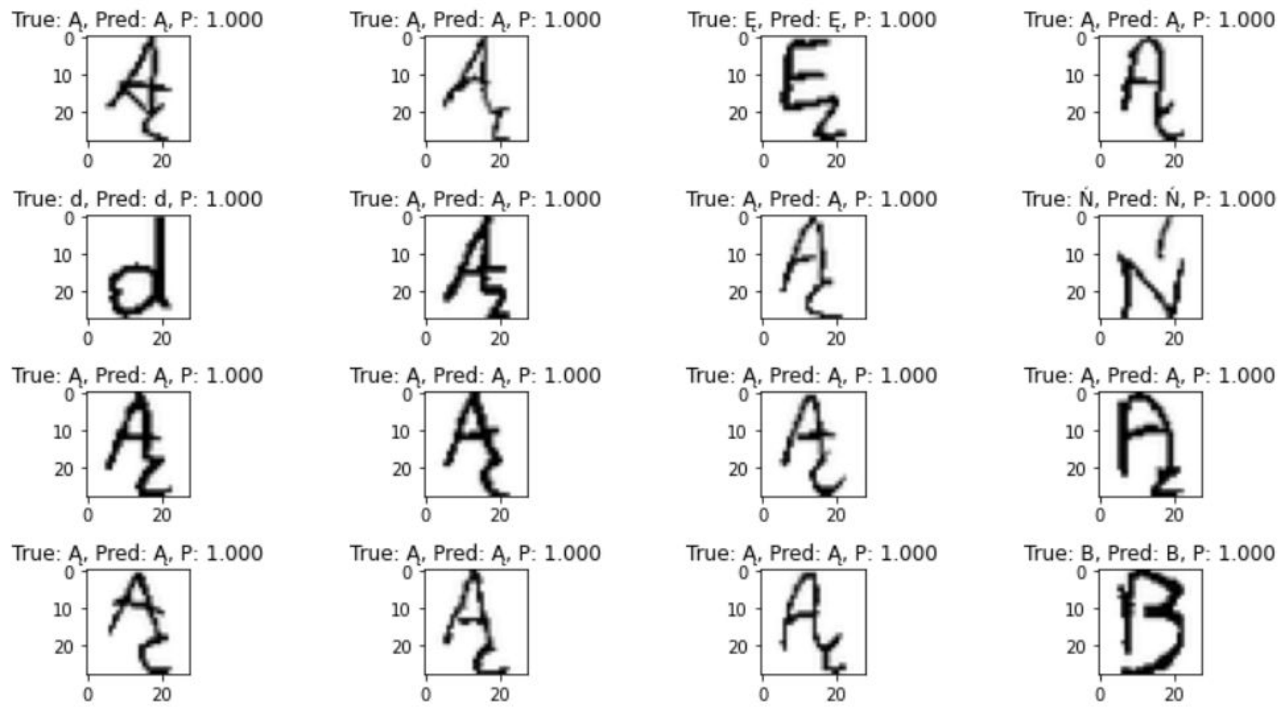
EarlyStopping: Epoche 25 - Validation loss hat sich in den letzten 5 Epochen nicht verbessert.

The Results

Dataset size	Running Time	Loss	Accuracy	Epochs
100 Bilder pro Klasse	2 min 10 sec	0.7215	0.7407	23
500 Bilder pro Klasse	11 min 58 sec	0.5587	0.7857	24
5000 Bilder pro Klasse	2 h 6 min 41 sec	0.4606	0.8226	25

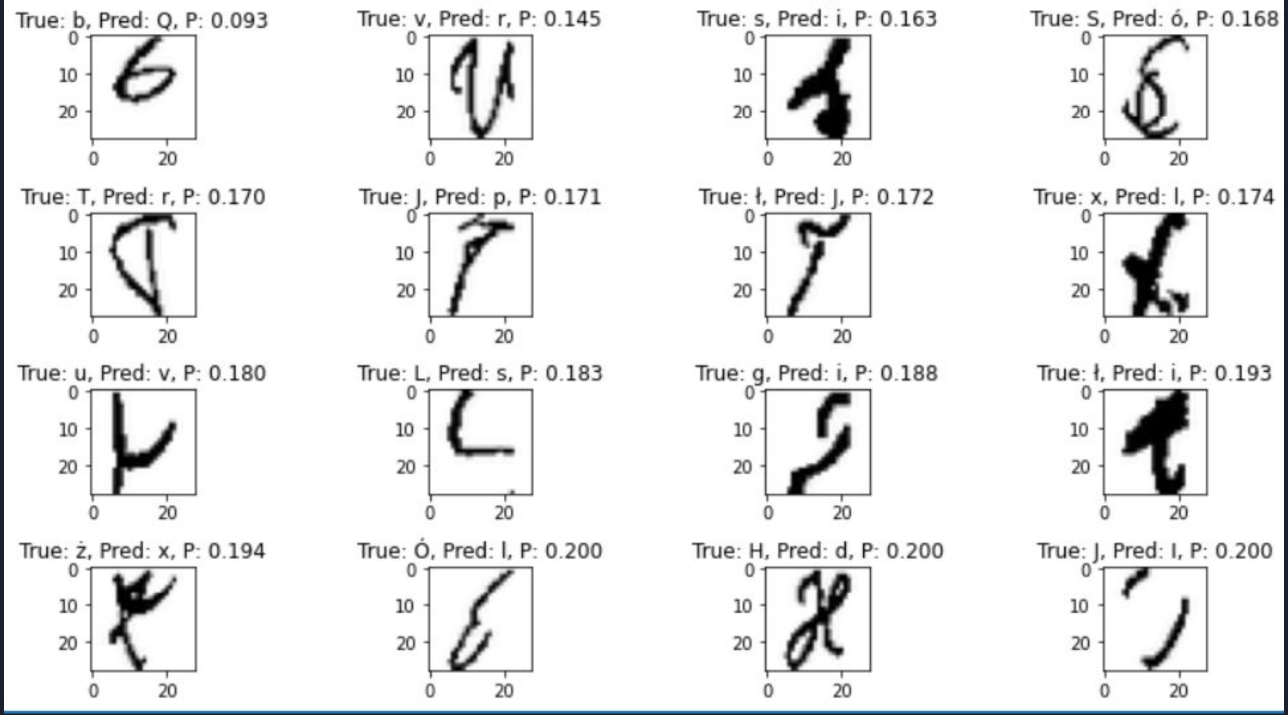
The Results

Richtige Predictions mit der höchsten Wahrscheinlichkeiten



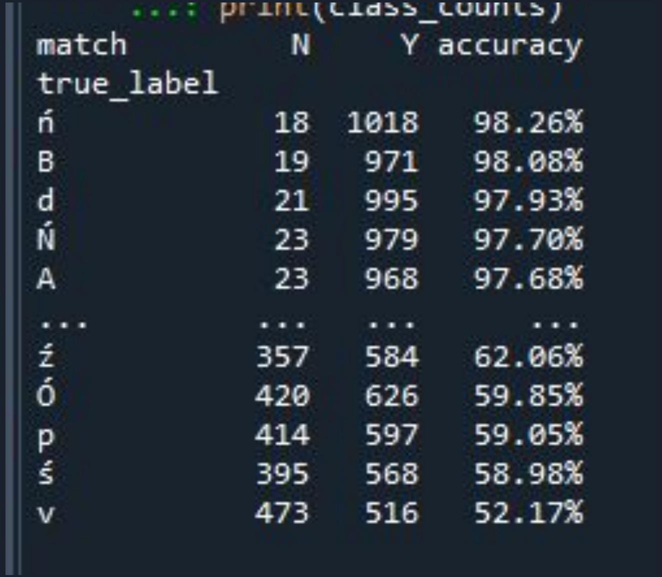
The Results

Falsche Predictions mit der niedrigsten Wahrscheinlichkeiten



The Results

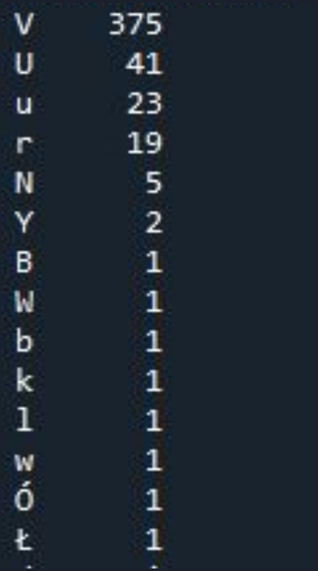
Top 5, Worst 5



```
   ...: print(class_counts)
match          N     Y accuracy
true_label
ń             18  1018   98.26%
B             19   971   98.08%
d             21   995   97.93%
Ń             23   979   97.70%
A             23   968   97.68%
...          ...   ...      ...
ź            357   584   62.06%
Ó            420   626   59.85%
p            414   597   59.05%
ś            395   568   58.98%
v            473   516   52.17%
```

The Results

Wrong predictions for letter “v”

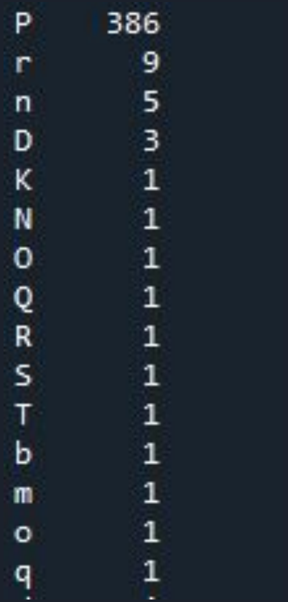


```
V    375
U     41
u     23
r     19
N      5
Y      2
B      1
W      1
b      1
k      1
l      1
w      1
Ó      1
Ł      1
```

79% von falschen Predictions: “V” statt “v”

The Results

Wrong predictions for letter “p”

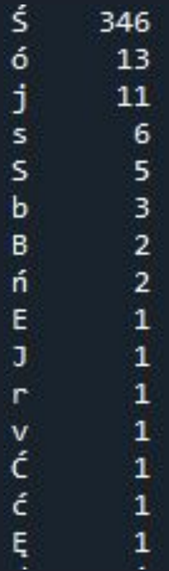


```
P    386
r      9
n      5
D      3
K      1
N      1
O      1
Q      1
R      1
S      1
T      1
b      1
m      1
o      1
q      1
```

93% von falschen Predictions: “P” statt “p”

The Results

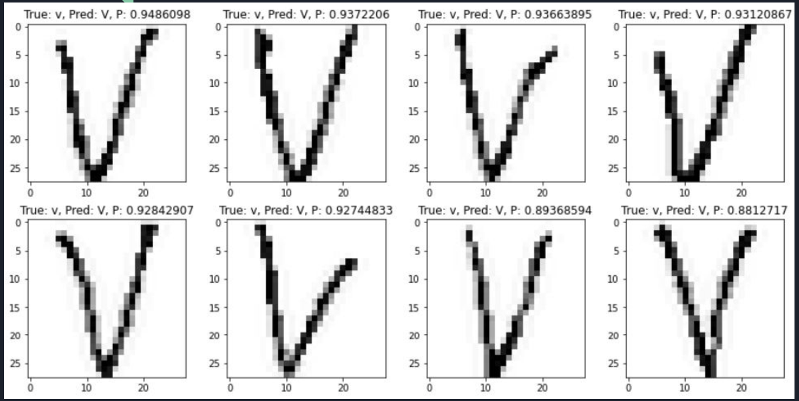
Wrong predictions for letter “ś”

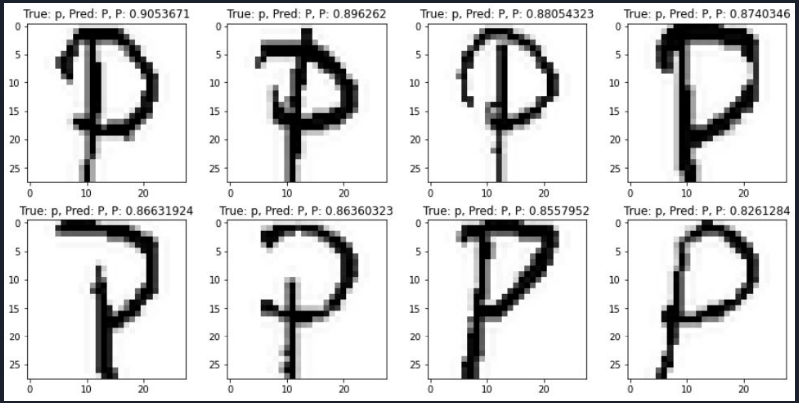


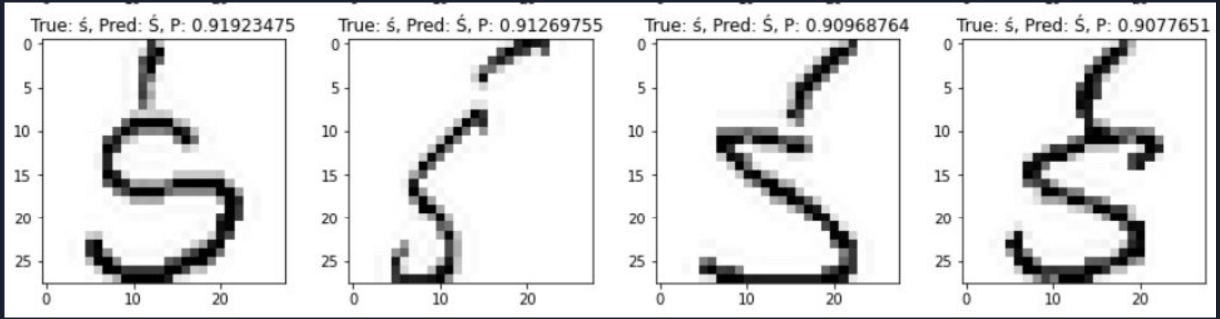
```
Ś    346
ó     13
j     11
s      6
S      5
b      3
B      2
ń      2
E      1
J      1
r      1
v      1
Ć      1
ć      1
Ę      1
```

87% von falschen Predictions: “Ś” statt “ś”

The Results - gross oder klein?







Lessons learned

Challenges:

- Das Lesen großer Datensätze in einem spezifischen Format
- Die Lernzeit des Modells
- Die Interpretation der Results und das Erkennen der Fehlerursachen

Key point:

- Data is the key: Die Qualität und Quantität der Daten können die Leistung des Modells erheblich beeinflussen

Conclusion:

- Der Aufbau eines Handwritten Character Recognition Models hat uns wertvolle Lektionen über Datenmanagement, Modelloptimierung und potenzielle Anwendungen von ML gelehrt.
- Vielleicht werden wir unsere Handschrift für die AI anpassen müssen.

Vielen Dank