HSLU Hochschule Luzern

OCRithmetic

Optical Character Recognition Calculator and Translator

Table of contents

01

Introduction

OCR?

What?

Optical **C**haracter **R**ecognition

Processing and analysis of visual data – images containing text

Why?

Transforms text into a machine-readable format

- Search
- Edit

Efficiency and productivity

Preservation

Calculator

WHY build an OCR Calculator?

- Bridge the gap between physical and digital mathematics.
- Educational support

Translator

WHY build an OCR Translator?

- Digitization of handwritten documents
- Enhanced searchability
- Language translation

Our Approach

CNN

- Calculator
- Translator

Pre-Existing Solutions

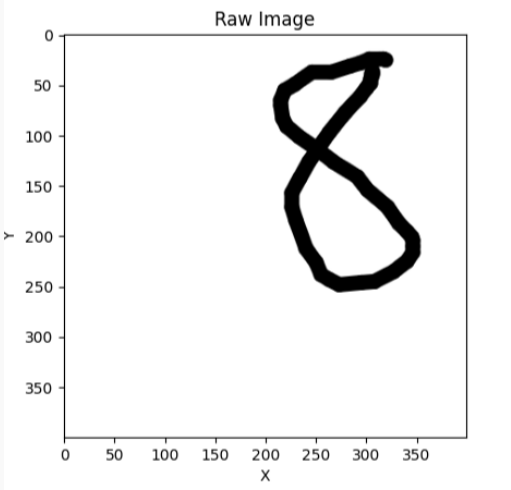
- Tesseract
- EasyOCR
- PaddleOCR
- TrOCR

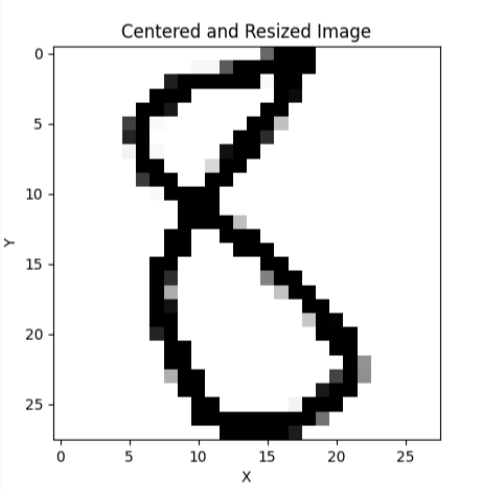
02

Calculator - I

Dataset

Classes = 0,1,2,3,4,5,6,7,8,9,+,-,*,/,=,x



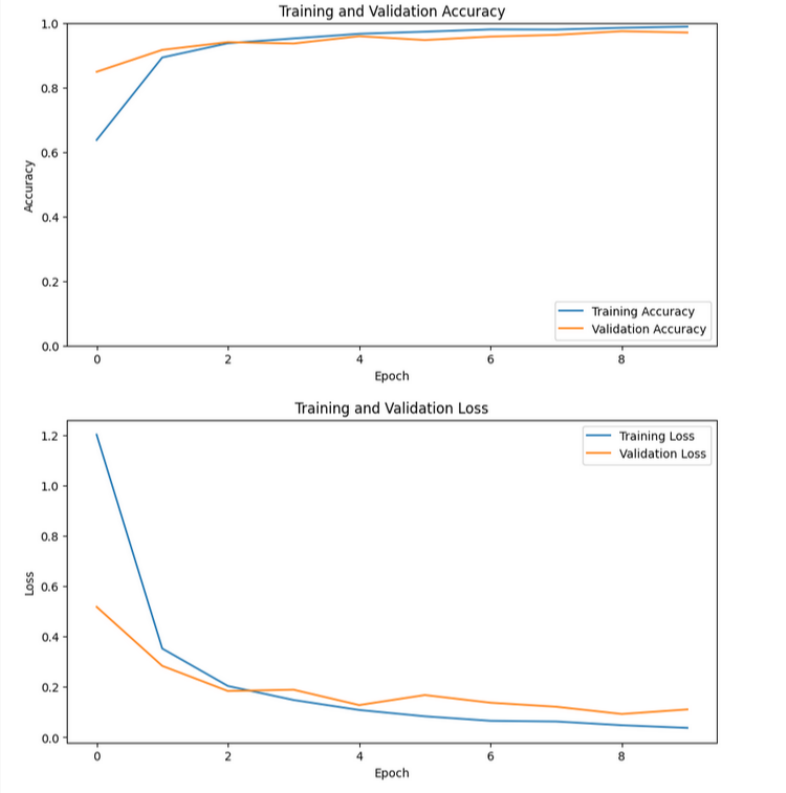


CNN - Definition

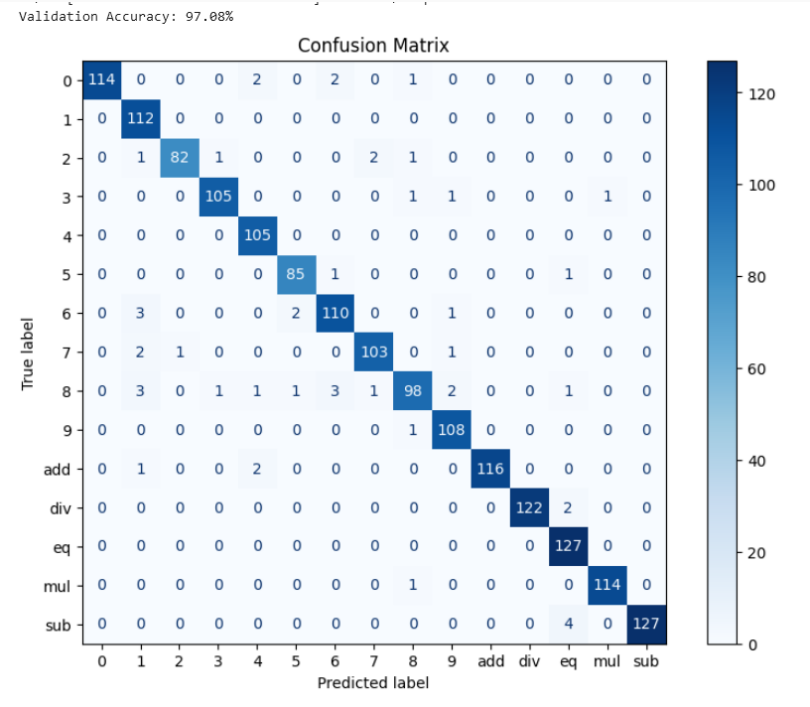
- 100 pictures per class
- Training/test split
- 10 epochs

```
def build_model(input_shape, num_classes):
    model = models.Sequential([
        layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, (3, 3), activation='relu'),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, (3, 3), activation='relu'),
        layers.Flatten(),
        layers.Dense(128, activation='relu'),
        layers.Dense(num_classes, activation='softmax')
    ])
```

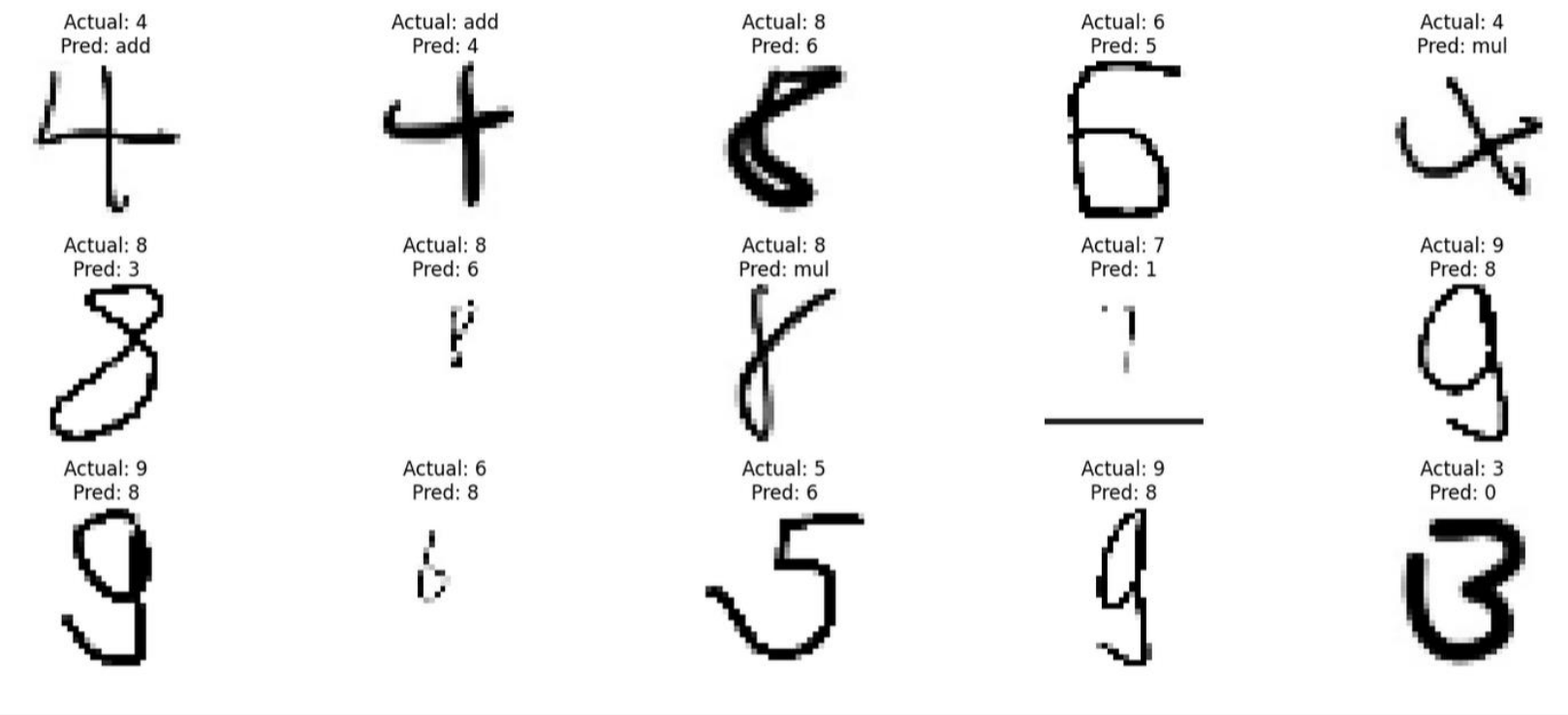
CNN Quality


Training and Validation Accuracy
Accuracy
Epoch
Training Accuracy
Validation Accuracy
Training and Validation Loss
Loss
Epoch
Training Loss
Validation Loss

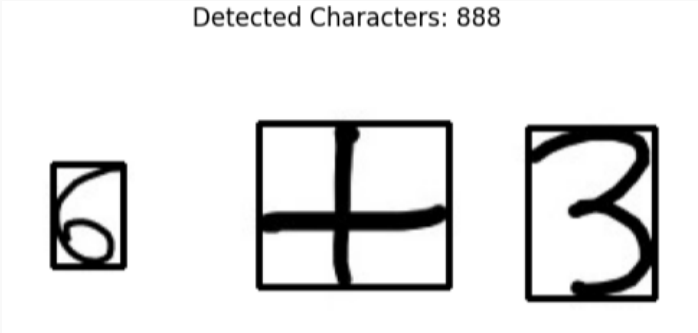
Confusion Matrix

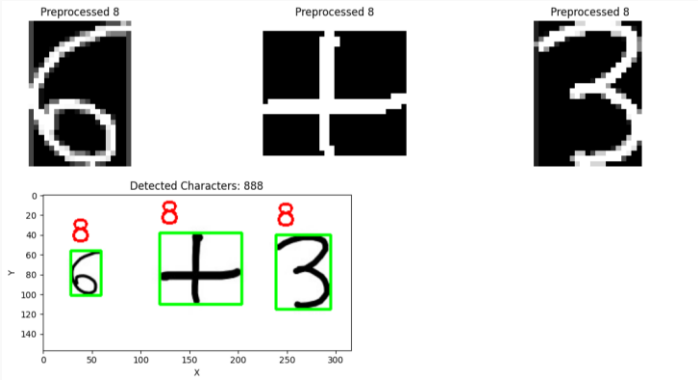


Missclassification

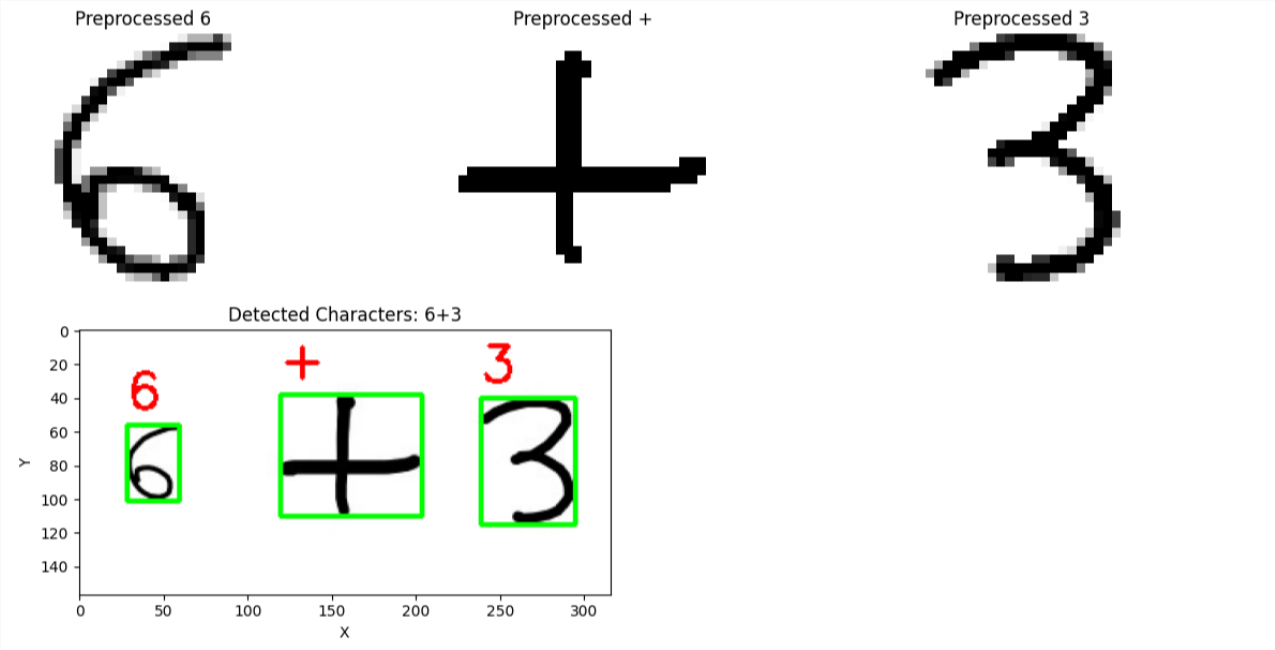


Object detection

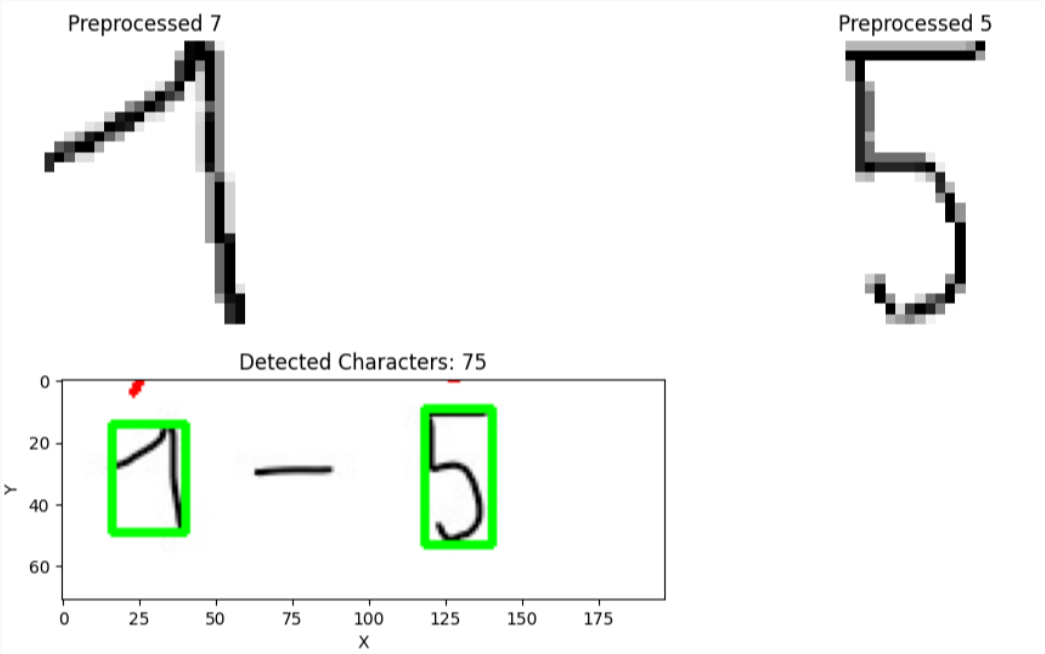




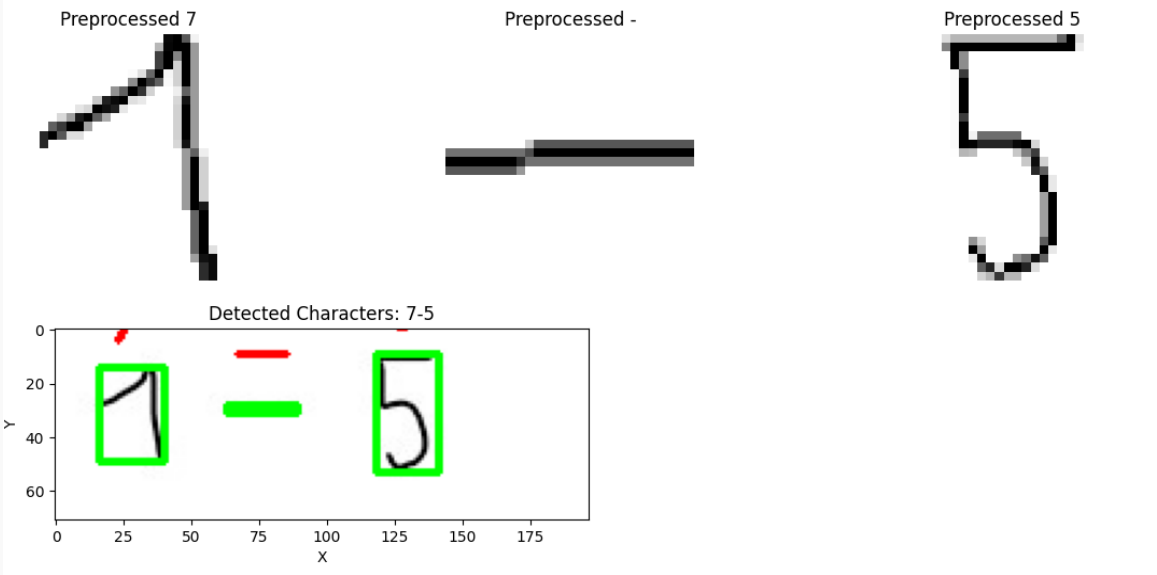
Object Detection Improvement



Detection Error

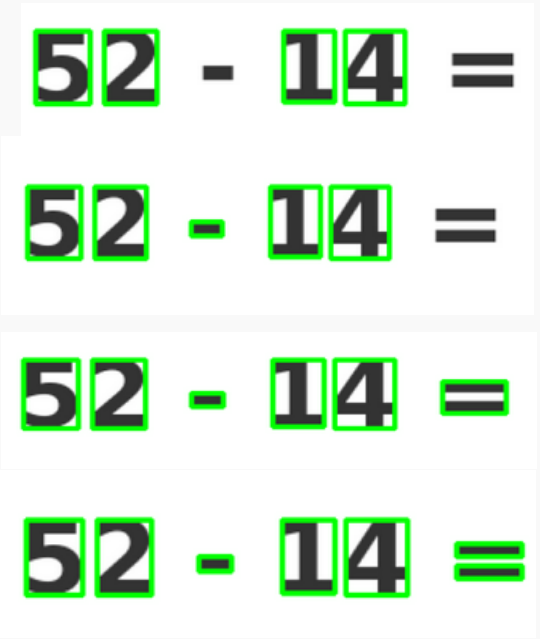


Object Detection Improvment



Object Detection Improvment

- The Detection has to be improved step by step



Learnings

- Great Dataset is needed
- Classification easy part
- Object Detection base on CV2 Contour Finding
- Calculation with fraction line
- This set up works fine with white background

03

Calculator - II

Steps

Image Loading

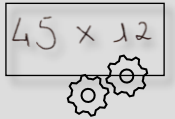
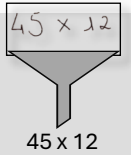
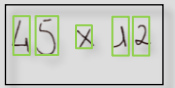


Image Preprocessing



Text Extraction



Contour Detection

Evaluate Expression

PaddleOCR

CER*: 19%

Tesseract

CER*: 38%

EasyOCR

CER*: 12%

TrOCR

CER *: 15%

* CER – Character Error Rate
Calculated for each solution, on 100 images

$$\frac{\text{Number of incorrect characters}}{\text{Total number of characters in the reference text}} \times 100\%$$

TrOCR

Transformer-based **O**ptical **C**haracter **R**ecognition

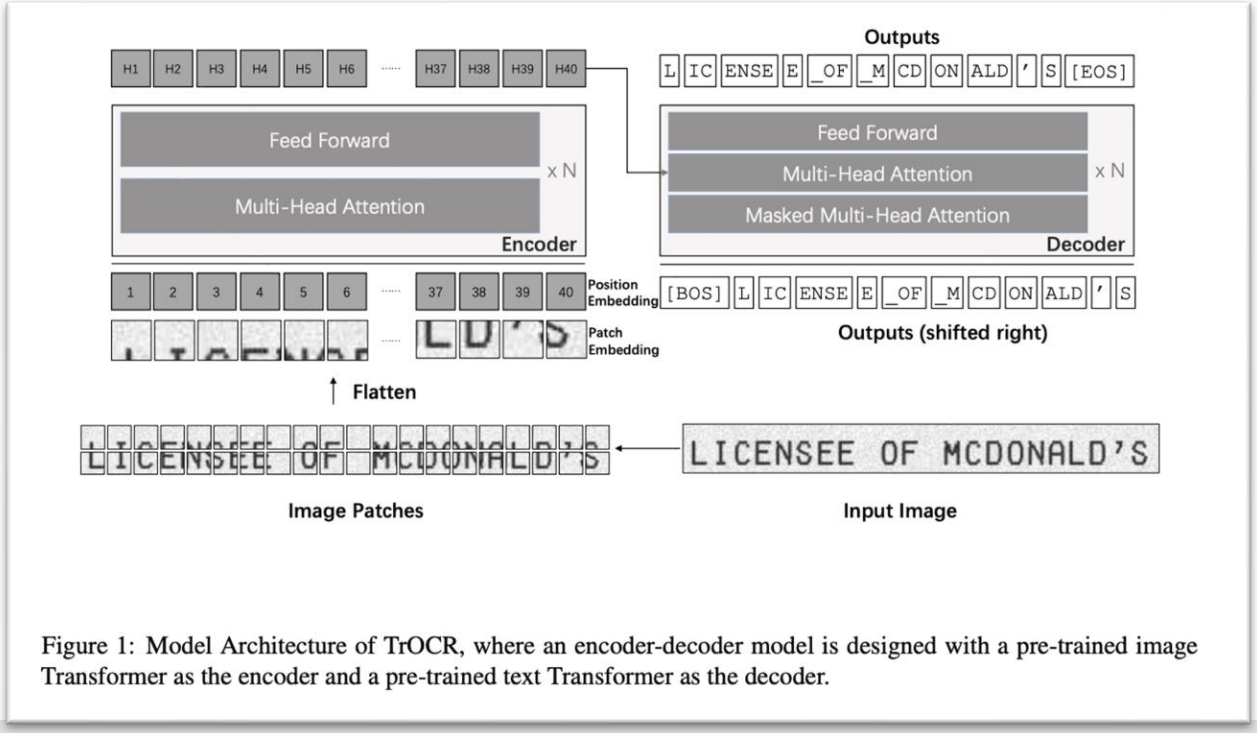


Figure 1: Model Architecture of TrOCR, where an encoder-decoder model is designed with a pre-trained image Transformer as the encoder and a pre-trained text Transformer as the decoder.

Li, Minghao, et al. 2023.

TrOCR

Transformer-based **O**ptical **C**haracter **R**ecognition

Pre-Trained (stage 1)	Handwritten	Printed	Scene
Small	Small	Small	Small
Base	Base	Base	Base
Large	Large	Large	

TrOCR

Transformer-based **O**ptical **C**haracter **R**ecognition

Pre-Trained (stage 1)	Handwritten	Printed	Scene
Small	Small	Small	Small
Base	Base	Base	Base
Large	Large	Large	

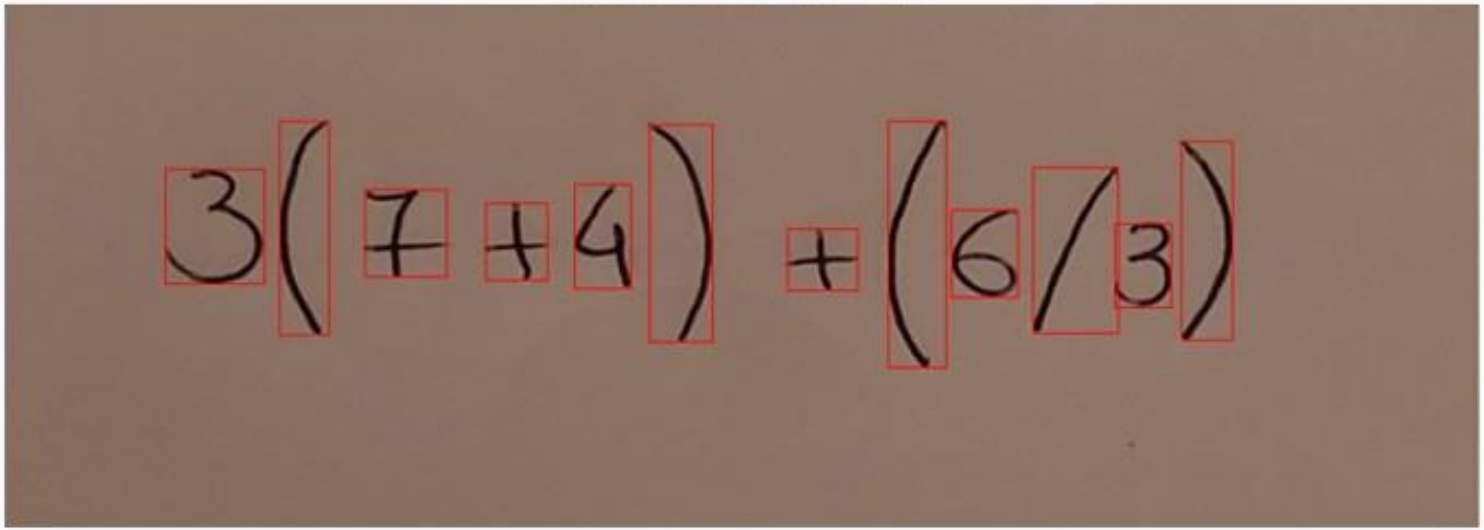
TrOCR

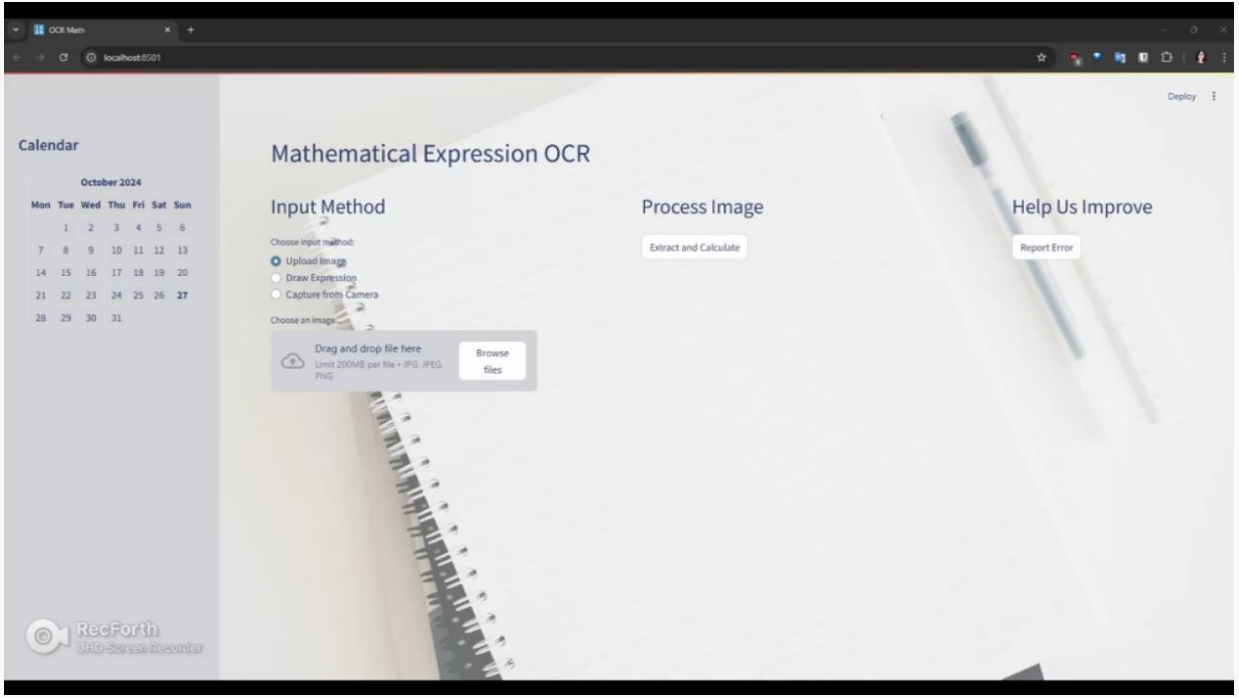
Transformer-based **O**ptical **C**haracter **R**ecognition

Pre-Trained (stage 1)	Handwritten	Printed	Scene
Small	Small	Small	Small
Base	Base	Base	Base
Large	Large	Large	

Extracted Text: 3(7+4)+(6/3)
Calculated Result: 35.0
Accuracy: 100.00%

TrOCR Result with Bounding Boxes
Extracted Text: 3(7+4)+(6/3)




OCR Math
localhost:8501
Deploy
Calendar
October 2024
Mon Tue Wed Thu Fri Sat Sun
1 2 3 4 5 6
7 8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
Mathematical Expression OCR
Input Method
Choose input method:
Upload Image
Draw Expression
Capture from Camera
Choose an image...
Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG
Browse files
Process Image
Extract and Calculate
Help Us Improve
Report Error
RecForth
UFO Screen Recorder

Demo

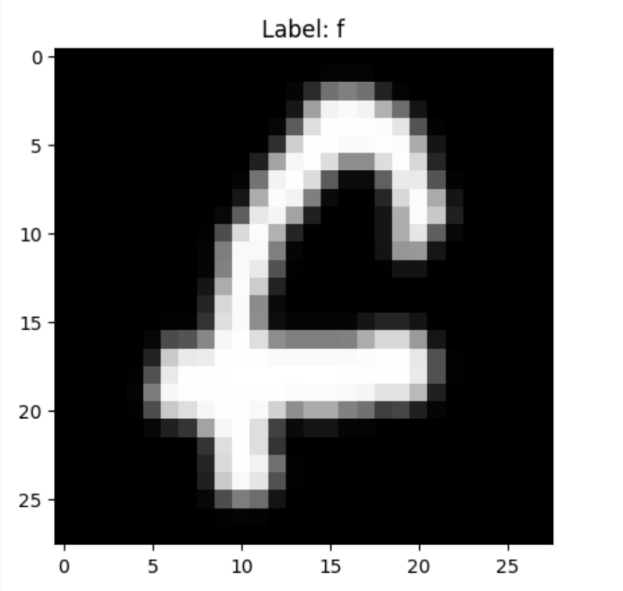
04

Letters

Pipeline

Dataset 1: Upper and Lower Case together in 26 classes

- The EMNIST Letters dataset merges a balanced set of the uppercase and lowercase letters into a single 26-class task.
- train: 88,800 test: 14,800 total: 103,600 classes: 37

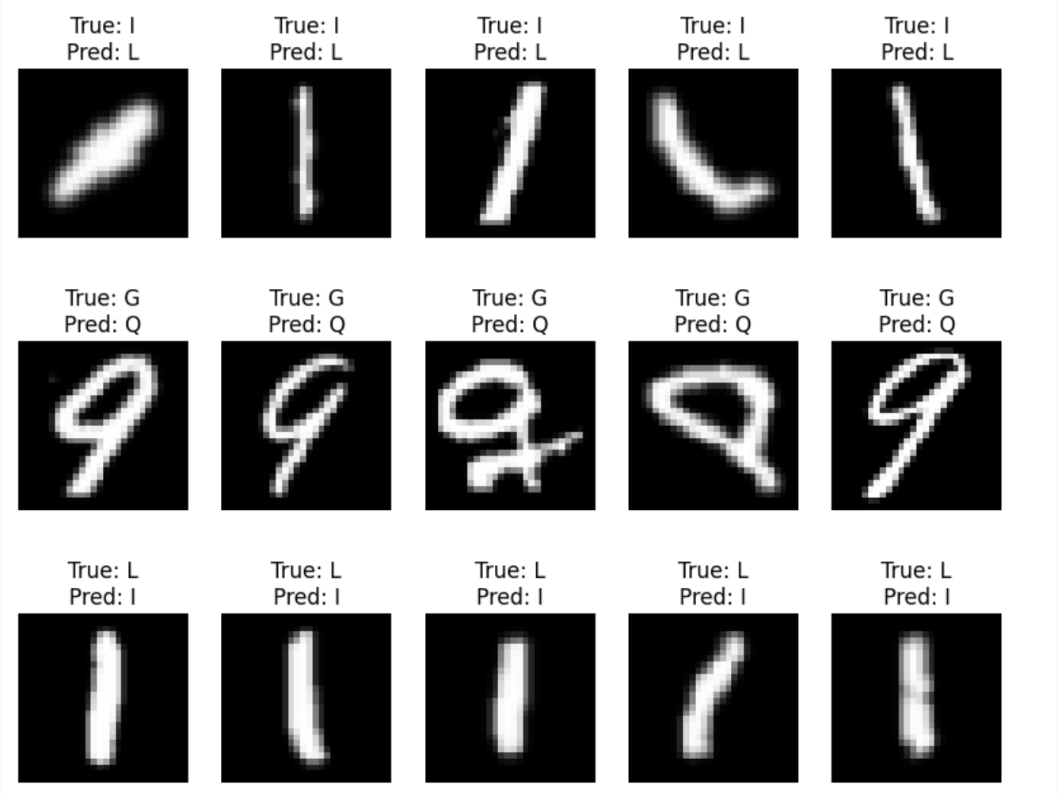


Simple architecture

```python
# Simple CNN architecture
model1 = models.Sequential([
    layers.Conv2D(16, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(26, activation='softmax')  # 26 letters
])
```

```
1388/1388 ━━━━━━━━━━━━━━━━━━━━ 15s 6ms/step - accuracy: 0.6705 - loss: 1.1120 - val_accuracy: 0.8627 - val_loss: 0.4231
Epoch 2/5
1388/1388 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - accuracy: 0.8882 - loss: 0.3467 - val_accuracy: 0.8888 - val_loss: 0.3401
Epoch 3/5
1388/1388 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9111 - loss: 0.2705 - val_accuracy: 0.8988 - val_loss: 0.3141
Epoch 4/5
1388/1388 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9211 - loss: 0.2376 - val_accuracy: 0.8950 - val_loss: 0.3166
Epoch 5/5
1388/1388 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.9290 - loss: 0.2130 - val_accuracy: 0.9060 - val_loss: 0.2868
463/463 - 1s - 3ms/step - accuracy: 0.9060 - loss: 0.2868

Test accuracy: 0.9060071706771851
```

Misclassification



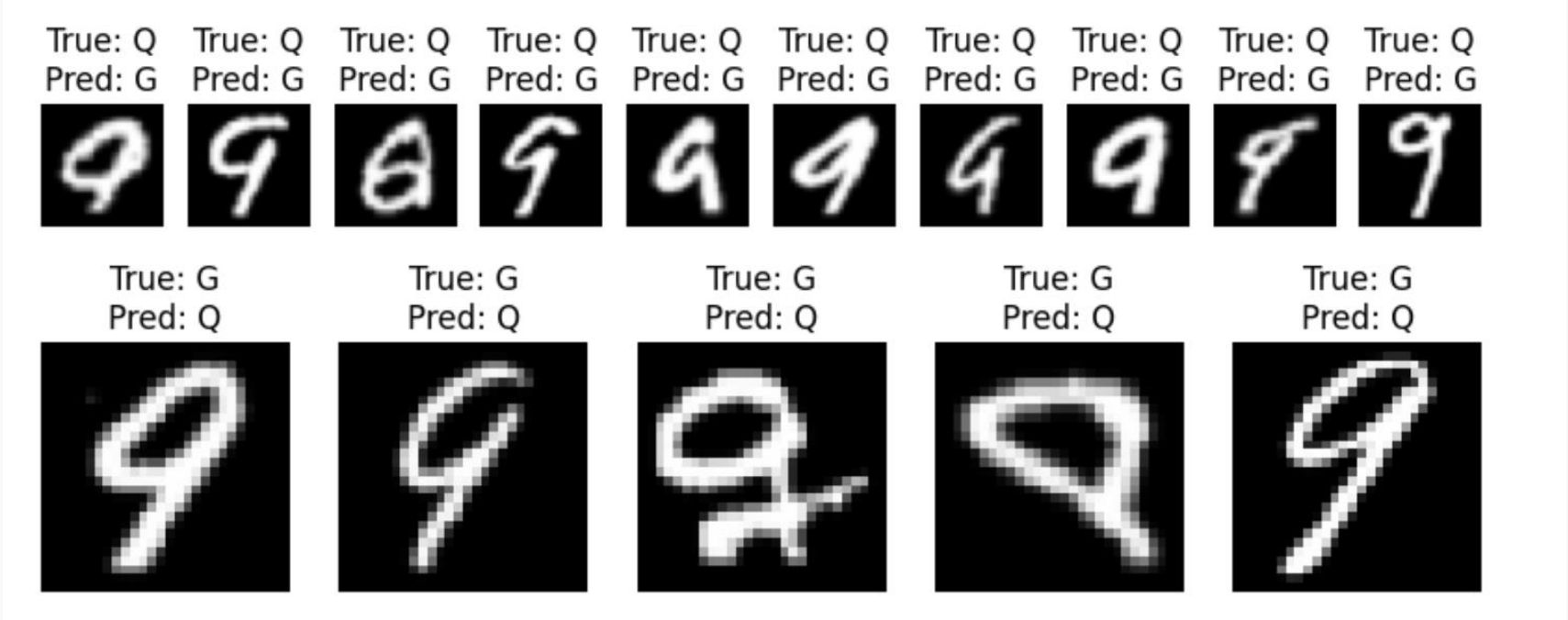
A bit more complex architecture

```
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(26, activation='softmax')  # 26 letters
])
```

```
Epoch 1/5
1388/1388 ━━━━━━━━━━━━━━━━━━━━ 16s 7ms/step - accuracy: 0.6822 - loss: 1.0708
Epoch 2/5
1388/1388 ━━━━━━━━━━━━━━━━━━━━ 12s 3ms/step - accuracy: 0.9046 - loss: 0.2864
Epoch 3/5
1388/1388 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.9240 - loss: 0.2221 -
Epoch 4/5
1388/1388 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.9346 - loss: 0.1882 -
Epoch 5/5
1388/1388 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.9406 - loss: 0.1710 -
463/463 - 1s - 2ms/step - accuracy: 0.9179 - loss: 0.2428

Test accuracy: 0.9178998470306396
```

92% — slightly better

Misclassification

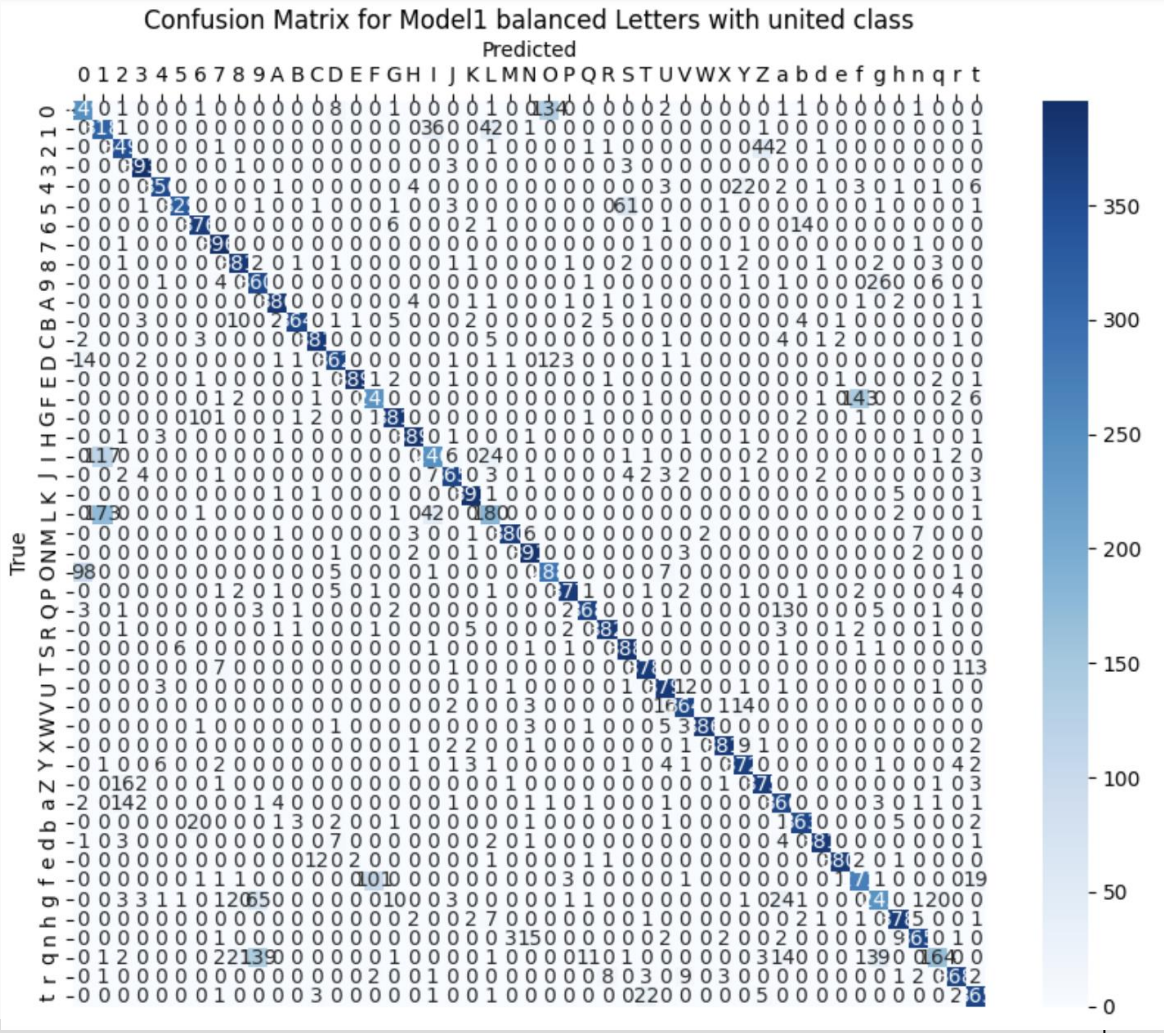


Separate labels for Upper & lower case

```
Epoch 1/5
1763/1763 ━━━━━━━━━━━━━━━━━━━━ 14s 5ms/step - accuracy: 0.6109 - loss: 1.3718
Epoch 2/5
1763/1763 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8451 - loss: 0.4459 -
Epoch 3/5
1763/1763 ━━━━━━━━━━━━━━━━━━━━ 12s 4ms/step - accuracy: 0.8649 - loss: 0.3791
Epoch 4/5
1763/1763 ━━━━━━━━━━━━━━━━━━━━ 8s 3ms/step - accuracy: 0.8744 - loss: 0.3395 -
Epoch 5/5
1763/1763 ━━━━━━━━━━━━━━━━━━━━ 13s 5ms/step - accuracy: 0.8860 - loss: 0.3098
588/588 - 2s - 3ms/step - accuracy: 0.8683 - loss: 0.3756

Test accuracy: 0.8683440685272217
```

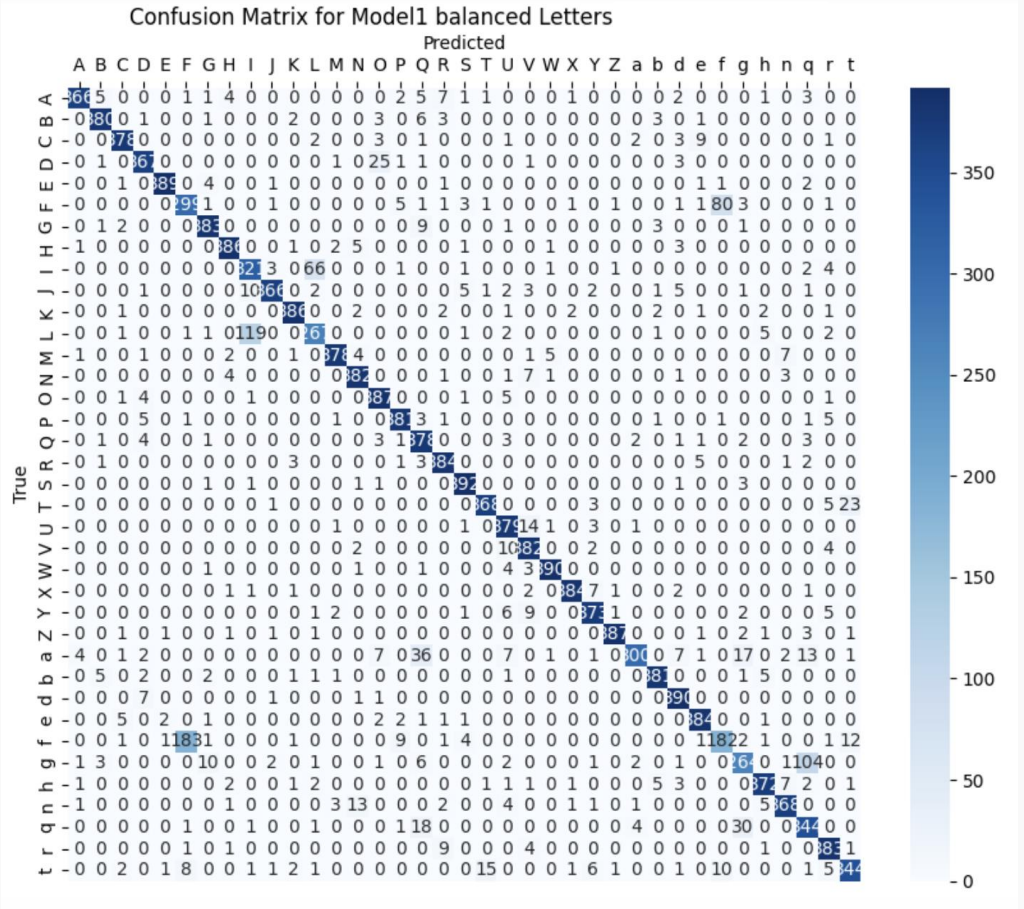
86% accuracy — why?

Confusion Matrix letters + digits



Confusion Matrix letters

Accuracy – 90%



Improvements

More epochs

Learning rate

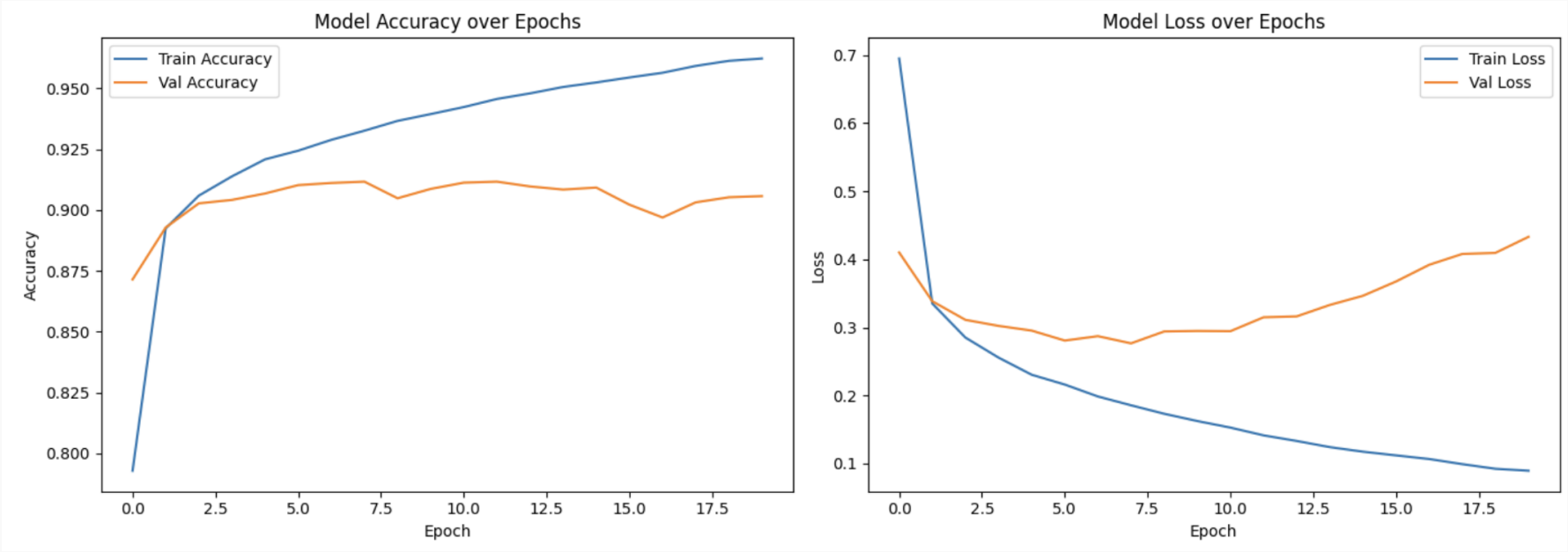
Augmentation of data

Another optimizer

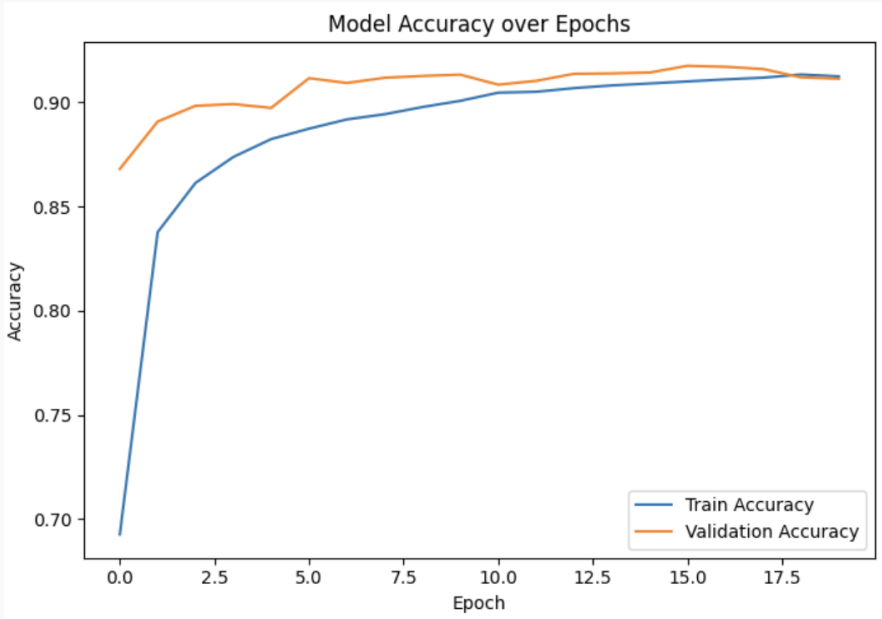
Early stopping

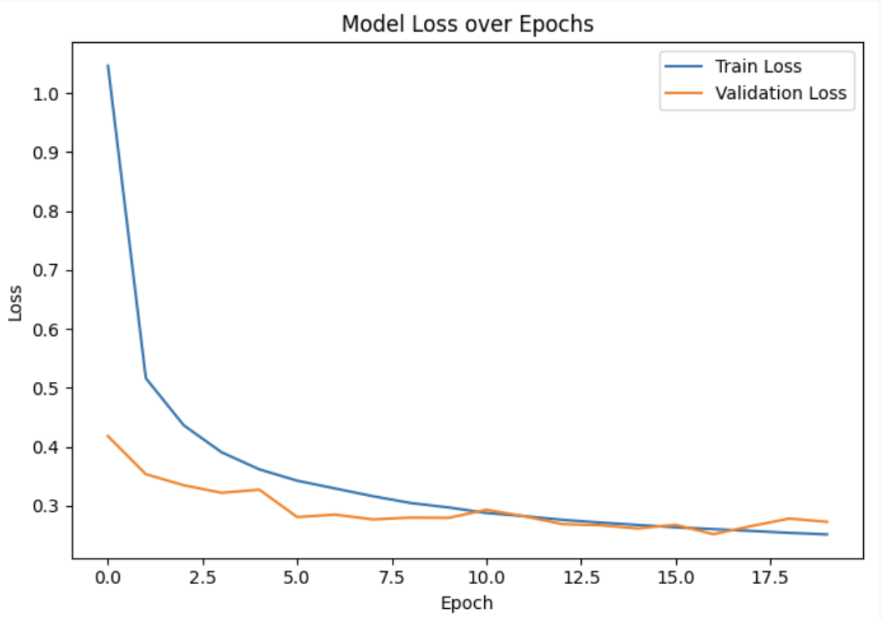
More complex model architecture

More epochs


Model Accuracy over Epochs
Train Accuracy
Val Accuracy
Accuracy
Epoch
0.800
0.825
0.850
0.875
0.900
0.925
0.950
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Model Loss over Epochs
Train Loss
Val Loss
Loss
Epoch
0.1
0.2
0.3
0.4
0.5
0.6
0.7

Data augmentation

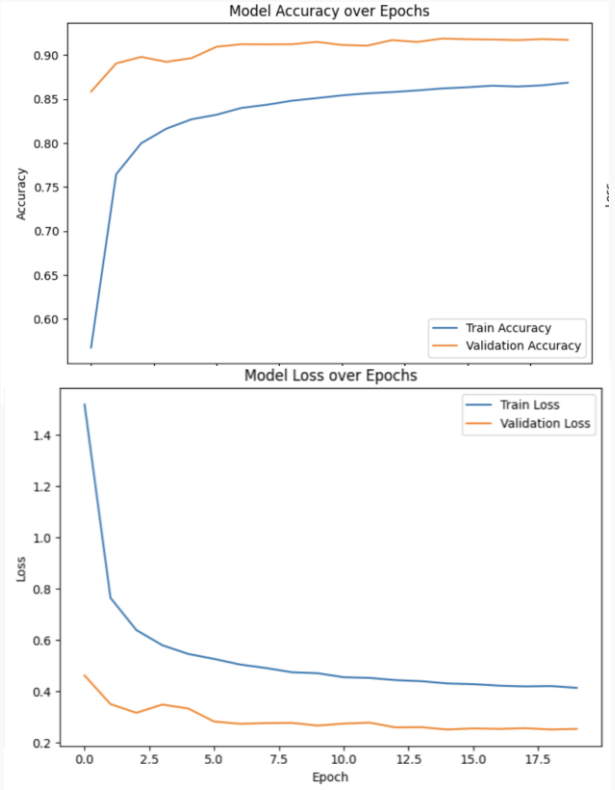




More complex model

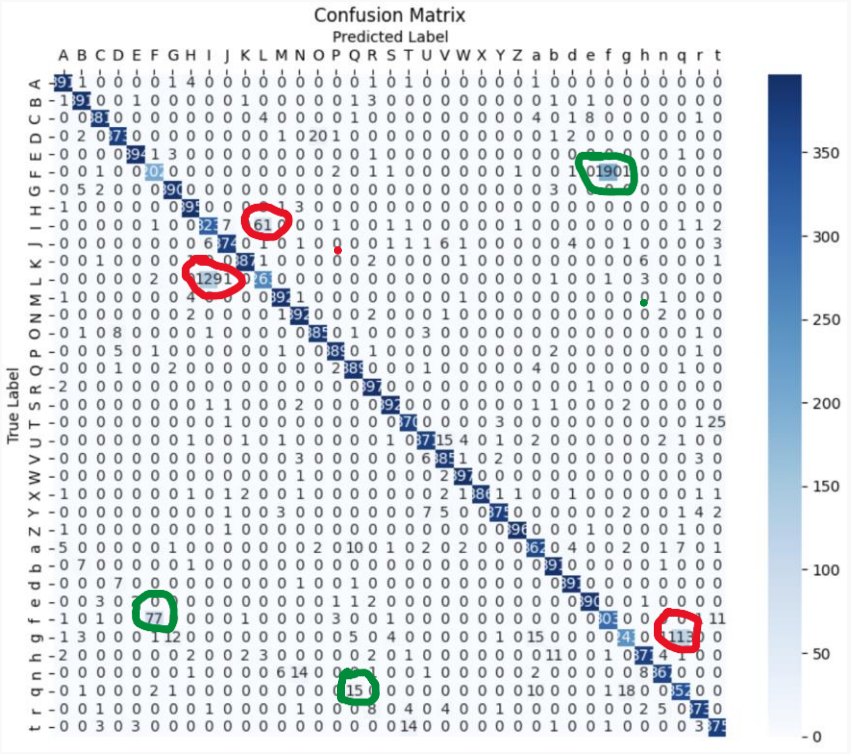
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
batch_normalization (BatchNormalization)	(None, 26, 26, 32)	128
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
dropout (Dropout)	(None, 13, 13, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 11, 11, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0
dropout_1 (Dropout)	(None, 5, 5, 64)	0
conv2d_2 (Conv2D)	(None, 3, 3, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 3, 3, 128)	512
flatten (Flatten)	(None, 1152)	0
dropout_2 (Dropout)	(None, 1152)	0
dense (Dense)	(None, 128)	147,584
batch_normalization_3 (BatchNormalization)	(None, 128)	512
dropout_3 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 37)	4,773

Total params: 246,439 (962.66 KB)
Trainable params: 245,733 (959.89 KB)
Non-trainable params: 704 (2.75 KB)
Optimizer params: 2 (12.00 B)

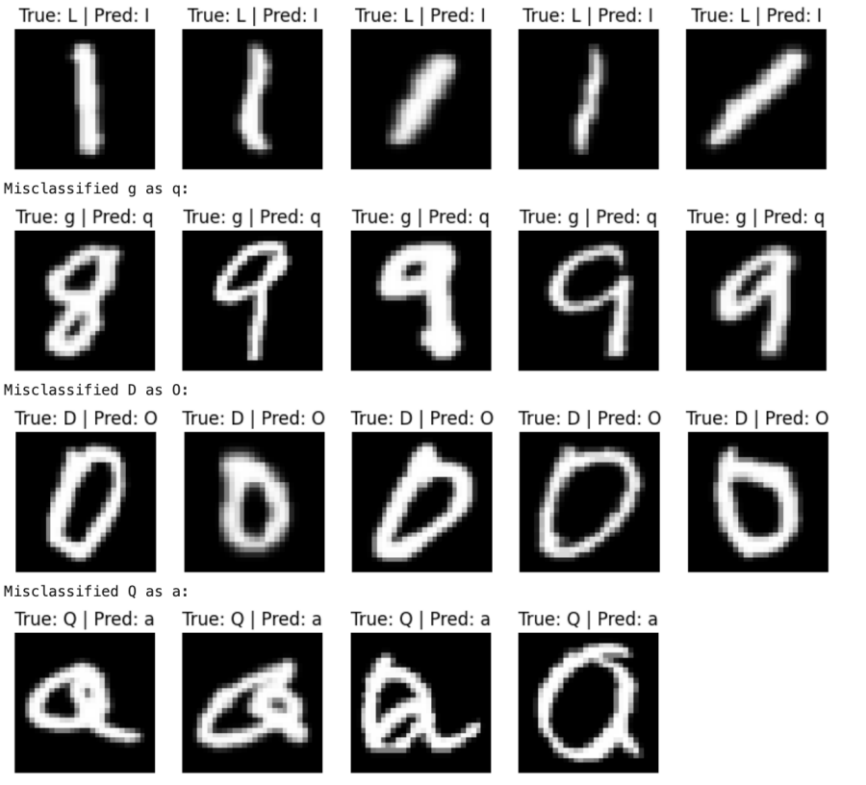


Putting all together

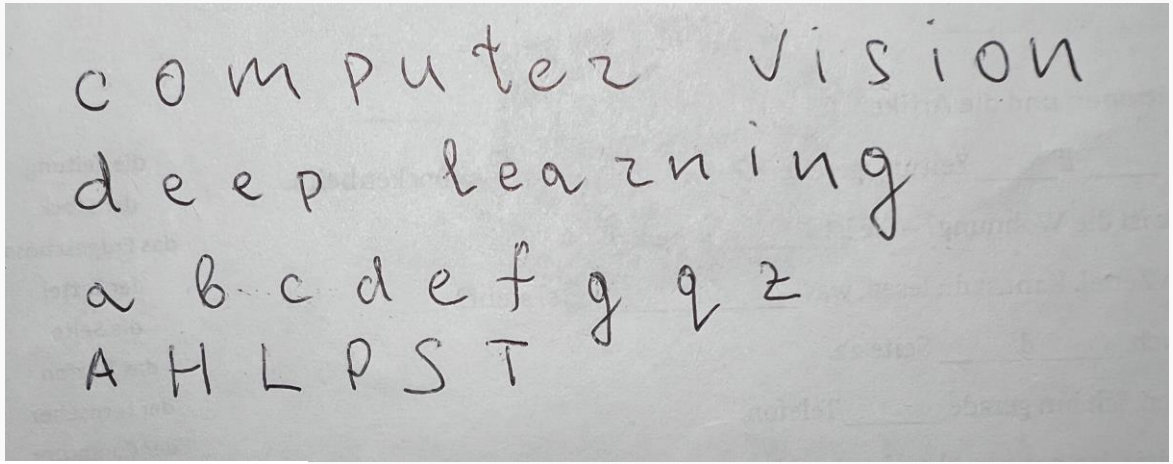
	precision	recall	f1-score	support
A	0.96	0.98	0.97	400
B	0.95	0.98	0.96	400
C	0.97	0.95	0.96	400
D	0.95	0.93	0.94	400
E	0.98	0.98	0.98	400
F	0.70	0.51	0.59	400
G	0.95	0.97	0.96	400
H	0.96	0.99	0.97	400
I	0.70	0.81	0.75	400
J	0.97	0.94	0.95	400
K	0.98	0.97	0.97	400
L	0.79	0.66	0.72	400
M	0.97	0.98	0.97	400
N	0.93	0.98	0.96	400
O	0.95	0.96	0.95	400
P	0.97	0.97	0.97	400
Q	0.92	0.97	0.94	400
R	0.94	0.99	0.97	400
S	0.98	0.98	0.98	400
T	0.94	0.93	0.93	400
U	0.95	0.93	0.94	400
V	0.92	0.96	0.94	400
W	0.97	0.99	0.98	400
X	1.00	0.96	0.98	400
Y	0.98	0.94	0.96	400
Z	0.99	0.99	0.99	400
a	0.90	0.91	0.90	400
b	0.95	0.98	0.96	400
d	0.97	0.98	0.97	400
e	0.97	0.97	0.97	400
f	0.61	0.76	0.68	400
g	0.90	0.61	0.73	399
h	0.95	0.93	0.94	400
n	0.96	0.92	0.94	400
q	0.73	0.88	0.80	400
r	0.96	0.93	0.94	400
t	0.89	0.94	0.91	400
accuracy			0.92	14799

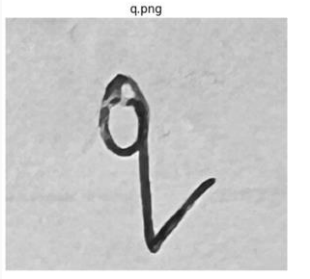


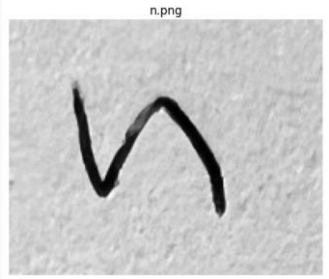
Remained misclassifications

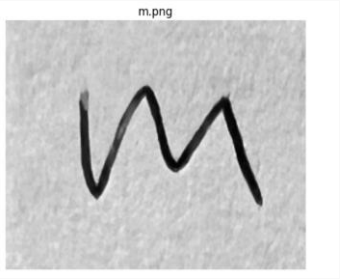


Recognition of own letters on **paper**



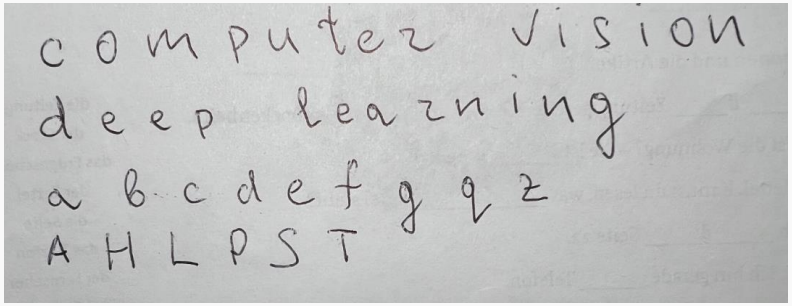




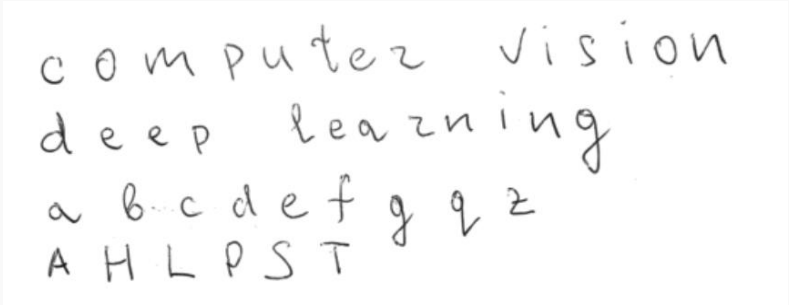


Preprocessing

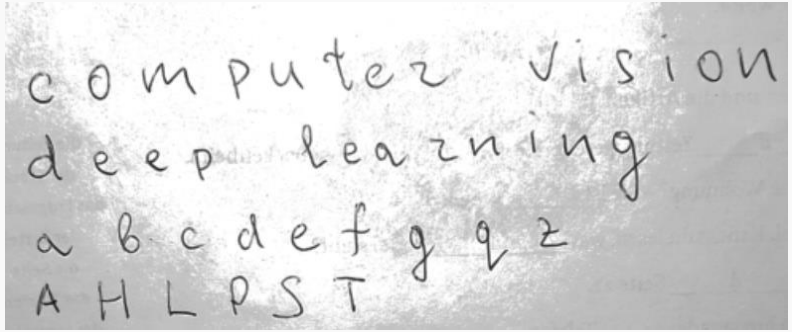
1



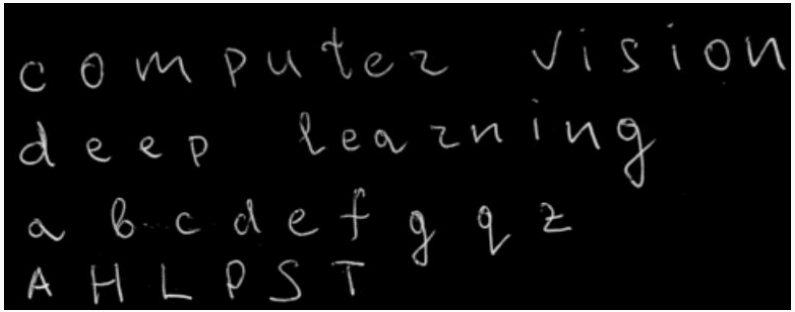
3



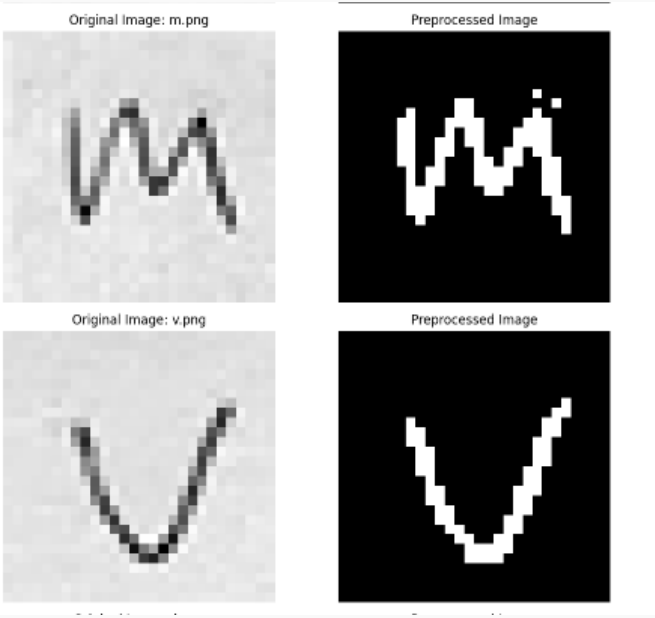
2

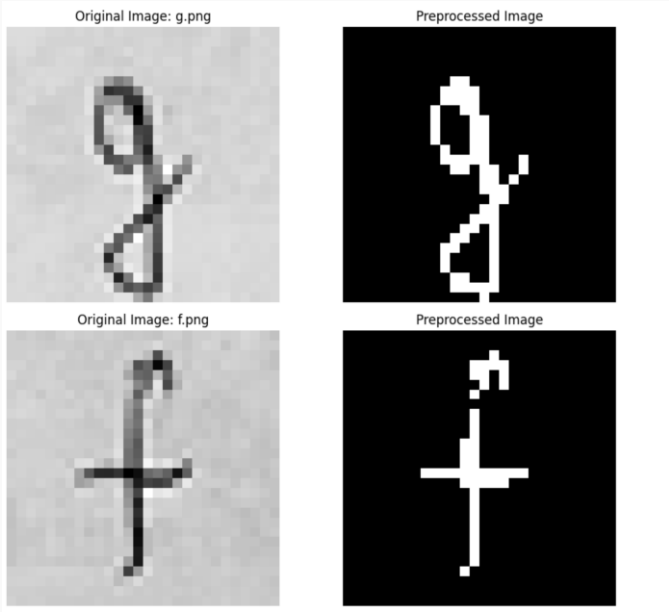


4

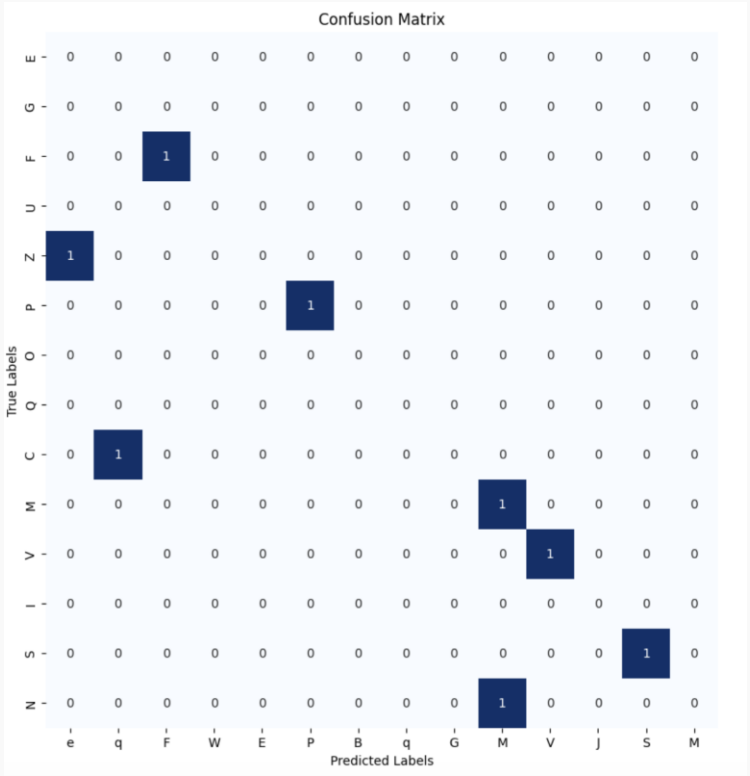


Letters on PAPER

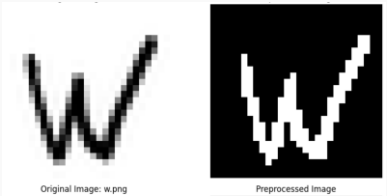


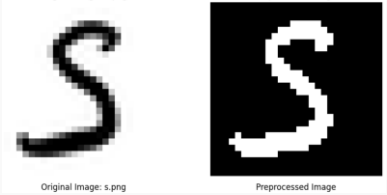


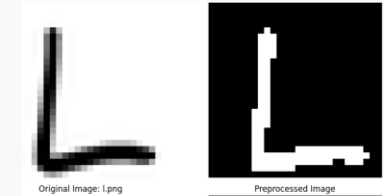
Letters on PAPER

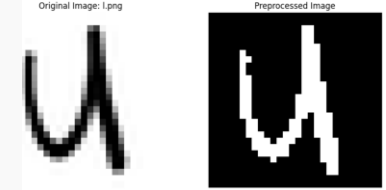


Recognition of own letters on **iPad**

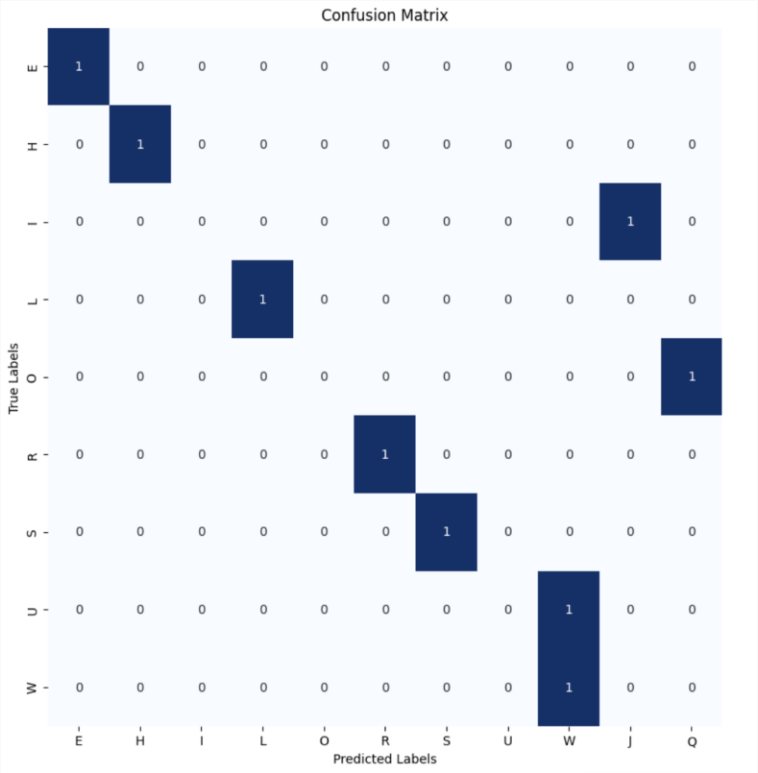




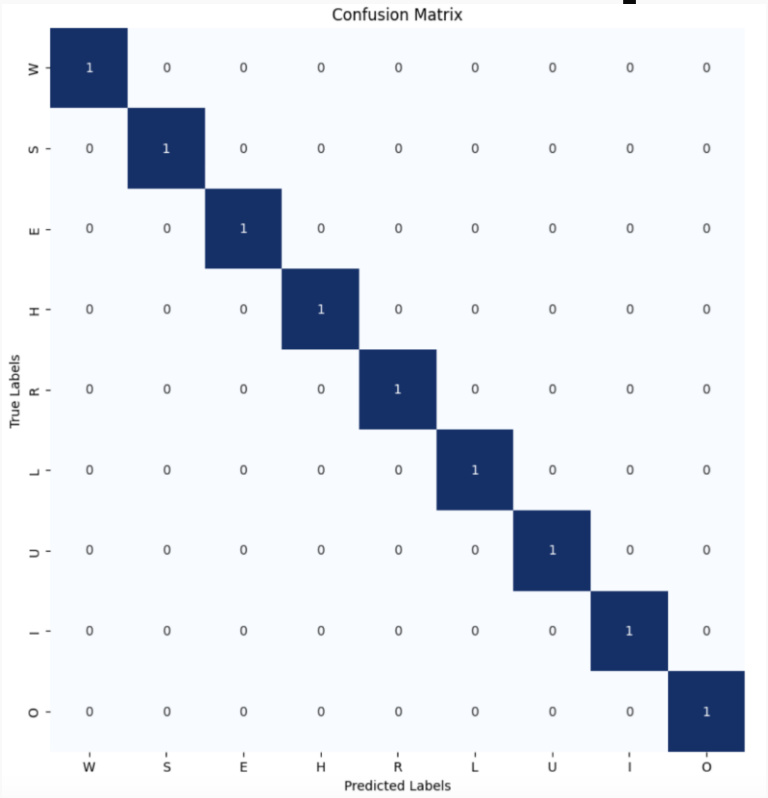




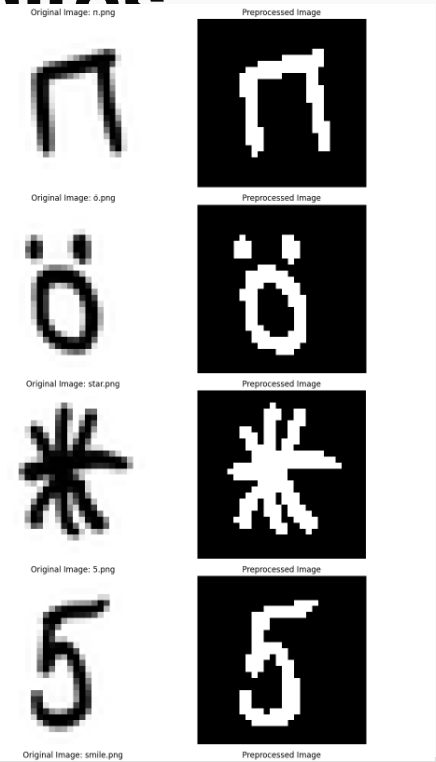
Performance with the "**best**" model

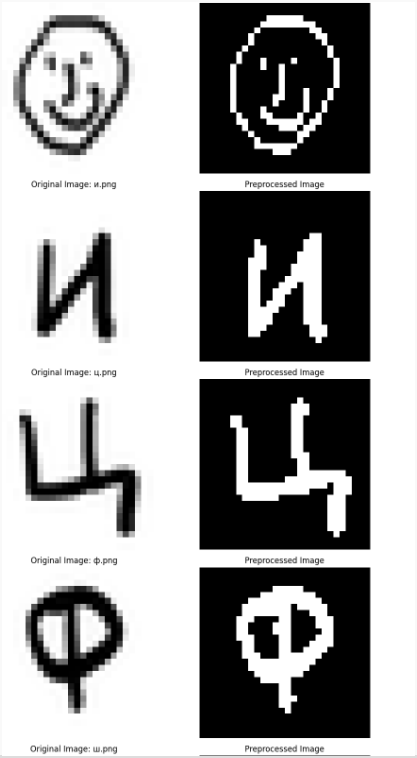


Performance with **simple** model

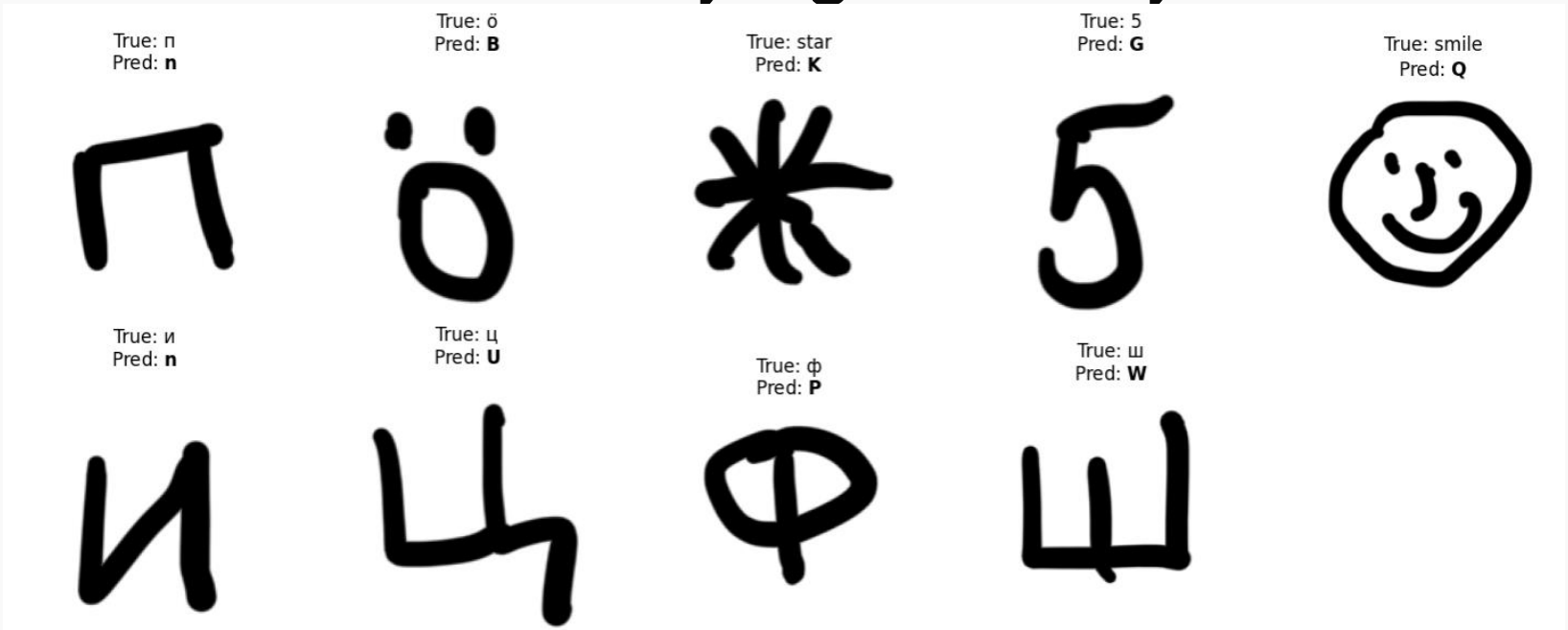


Other symbols: **Cyrillic, German, Smiles**





Result of classifying other symbols



05

Object Detection & Translation

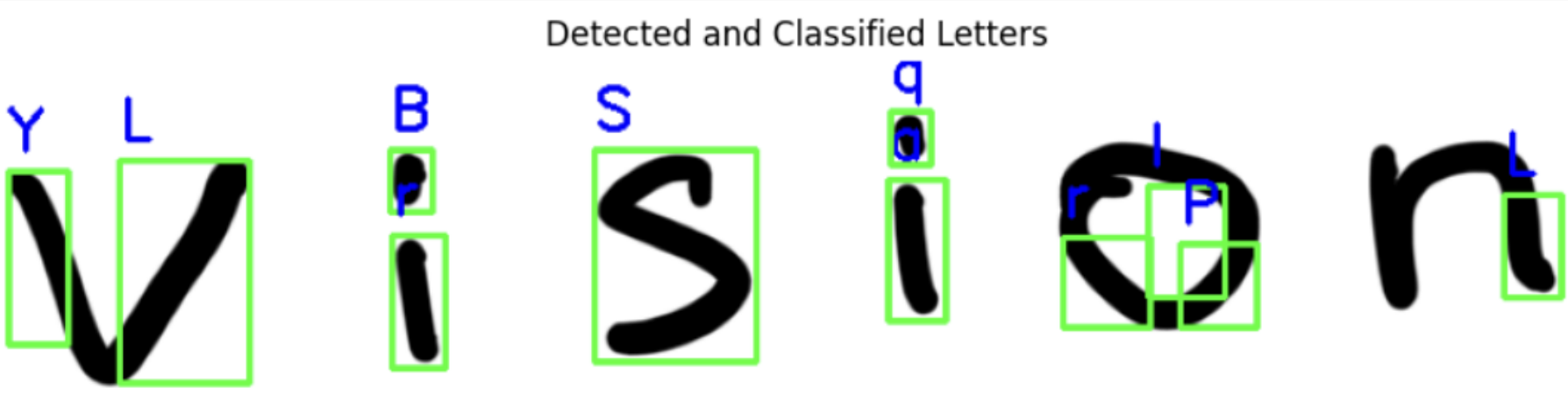
Pipeline

1. **Load and Display the Input Image**
2. **Perform Selective Search for Letter Detection**
3. **Filter and Sort Detected Regions**
4. **Preprocess Detected Letter Regions for Classification**
5. **Classify Each Detected Letter**
6. **Combine Classified Letters into a Word**
7. **Translate!**

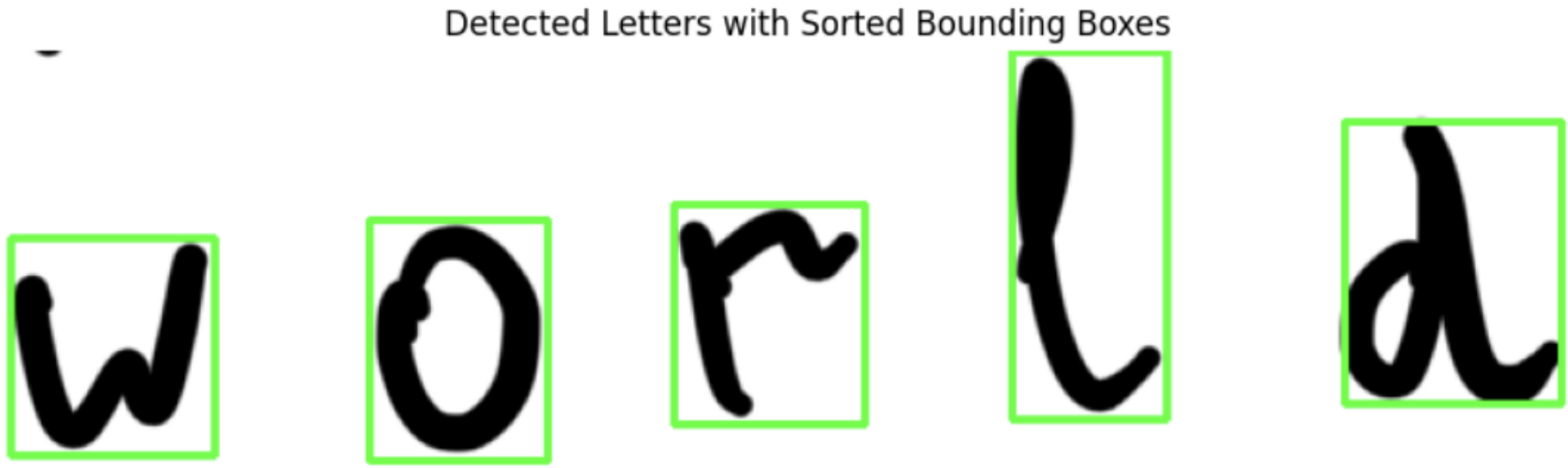
Image

Detected regions

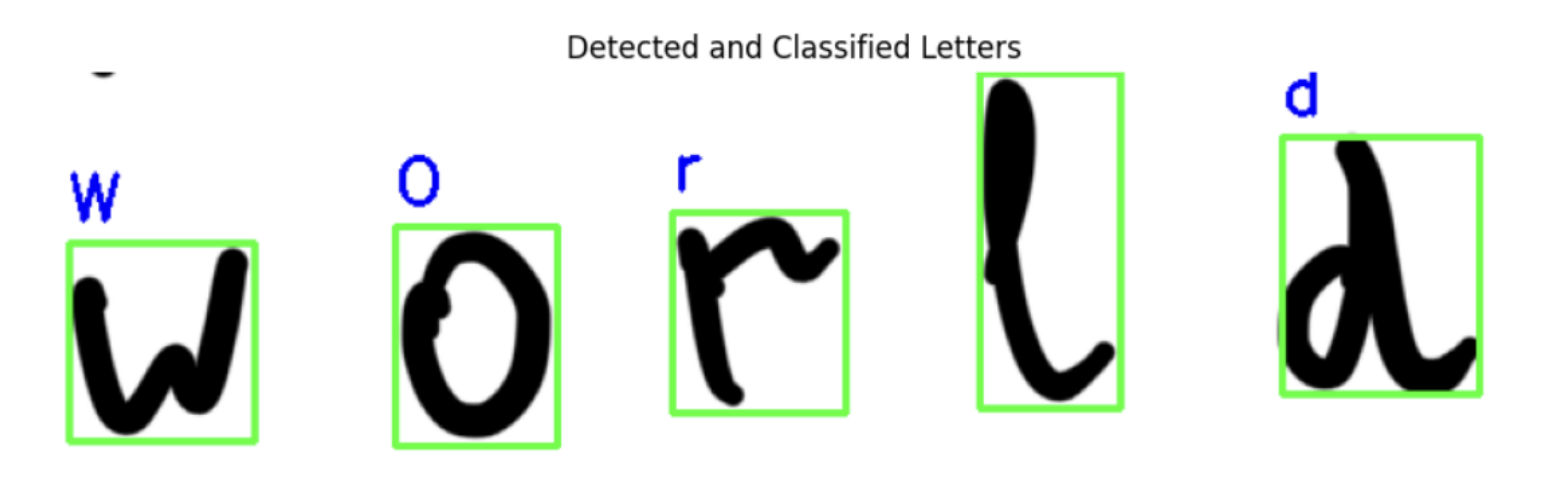
Classified "letters"



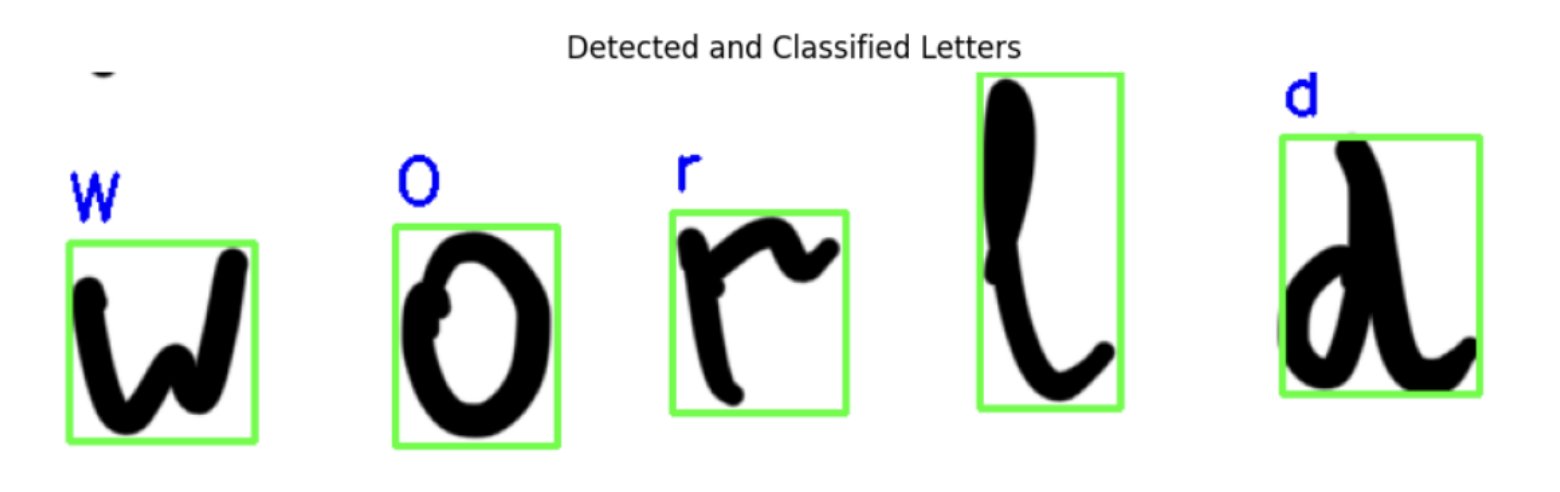
Objects detected in "world"



Classified "world"



Classified "WORLD"




```
Detected Word: WOrLd
```

!!!

Translation

```
WOrLd:
German: Welt
Ukrainian: CBIT
Spanish: Mundo
Russian: Мир
```

Translation (behind the scene)

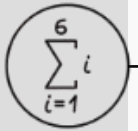
```
# !pip install googletrans==3.1.0a0   # to install on Colab
from googletrans import Translator

def translate_word(word):
    translator = Translator()
    translation_ge = translator.translate(word, src='en', dest='de')
    translation_ua = translator.translate(word, src='en', dest='uk')
    translation = translator.translate(word, src='en', dest='ru')
    translation1 = translator.translate(word, src='en', dest='es')
    return '\nGerman: ' + translation_ge.text + '\nUkrainian: ' + translation_ua.text + \
     '\nSpanish: ' + translation1.text + '\nRussian: '+ translation.text
# example usage
en_words = [detected_word]
for word in en_words:
    translation = translate_word(word)
    print(f"{word}: {translation}")
```

6 Future Work

Future Work

Update the pipeline to be able to recognize sentences not only words

Further fine-tune the TrOCR model on domain-specific datasets to improve accuracy for particular types of calculations or handwriting styles.



Enhance the system to recognize and process more complex mathematical symbols and equations, expanding its capabilities beyond basic calculations.

Real-time OCR and translation for live video streams, enabling instant calculation and translation of handwritten text.