

OCRithmetic

Optical Character Recognition
Calculator and Translator

Table of contents

01

Introduction

Calculator and Translator

03

Calculator - II

Use of pre-trained models for handwritten character recognition. Demo

05

Future Work

Proposition of possible next iterations of this project

02

Calculator - I

CNN designed to recognize handwritten digits and arithmetic operators

04

Letters

CNN designed to recognize handwritten characters and perform translation

06

Discussion

01

Introduction

OCR?

What?

Optical Character Recognition

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

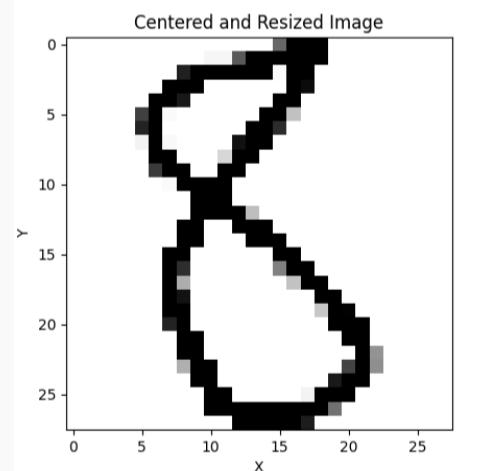
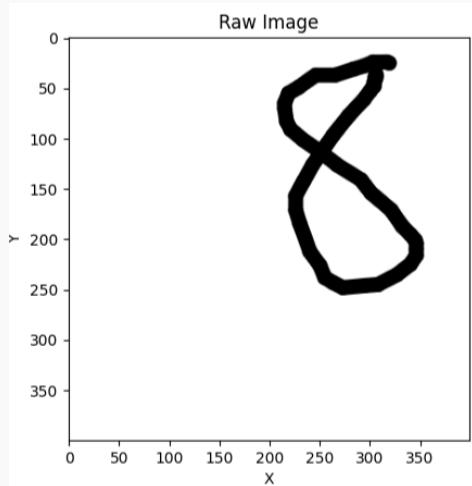
- ▷ Tesseract
- ▷ EasyOCR
- ▷ PaddleOCR
- ▷ TrOCR

02

Calculator - I

Dataset

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

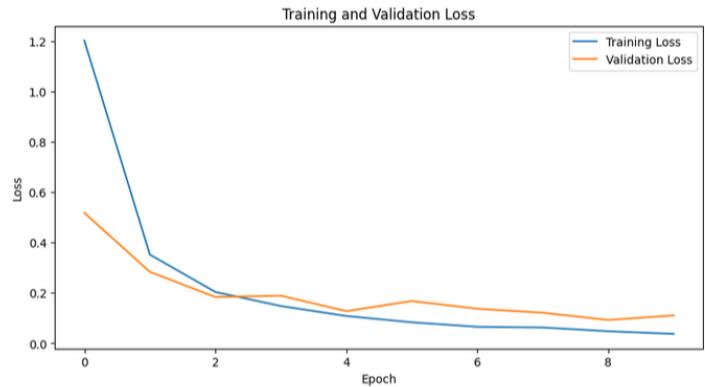
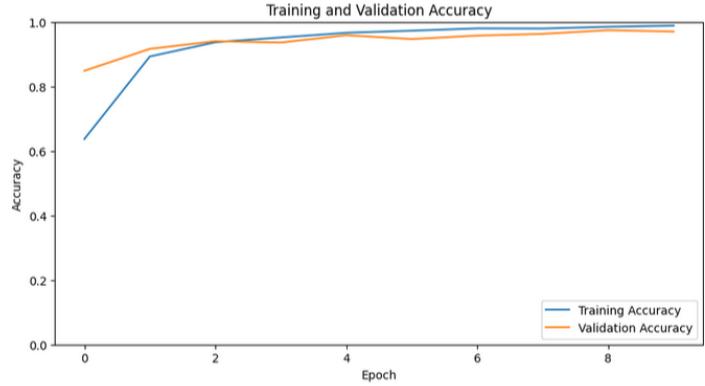


CNN - Definition

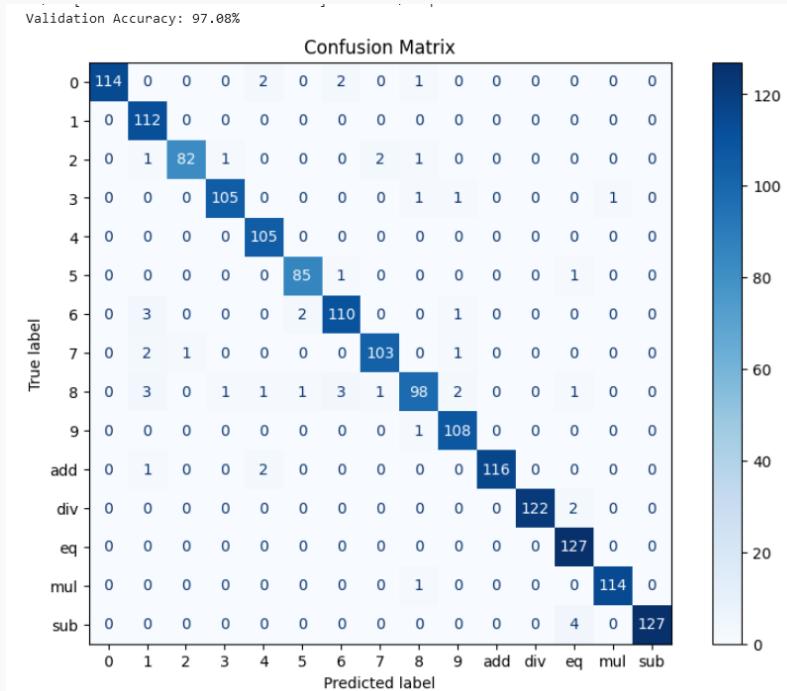
- 100 pictures per class
- Training/test split
- 10 epochs

```
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

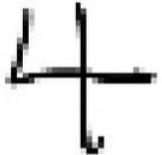


Confusion Matrix



Missclassification

Actual: 4
Pred: add



Actual: 8
Pred: 3



Actual: 9
Pred: 8



Actual: add
Pred: 4



Actual: 8
Pred: 6



Actual: 6
Pred: 8



Actual: 8
Pred: 6



Actual: 8
Pred: mul



Actual: 5
Pred: 6



Actual: 6
Pred: 5



Actual: 7
Pred: 1



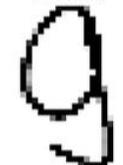
Actual: 9
Pred: 8



Actual: 4
Pred: mul



Actual: 9
Pred: 8

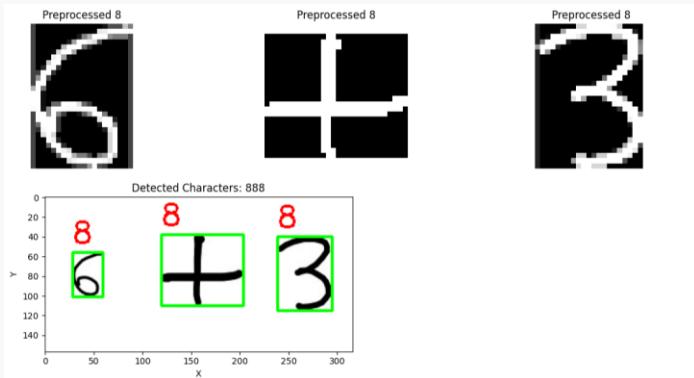
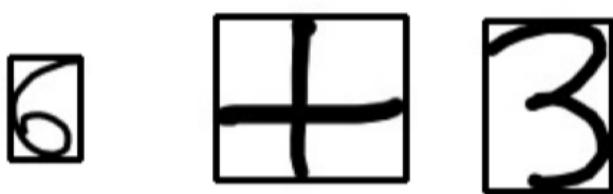


Actual: 3
Pred: 0

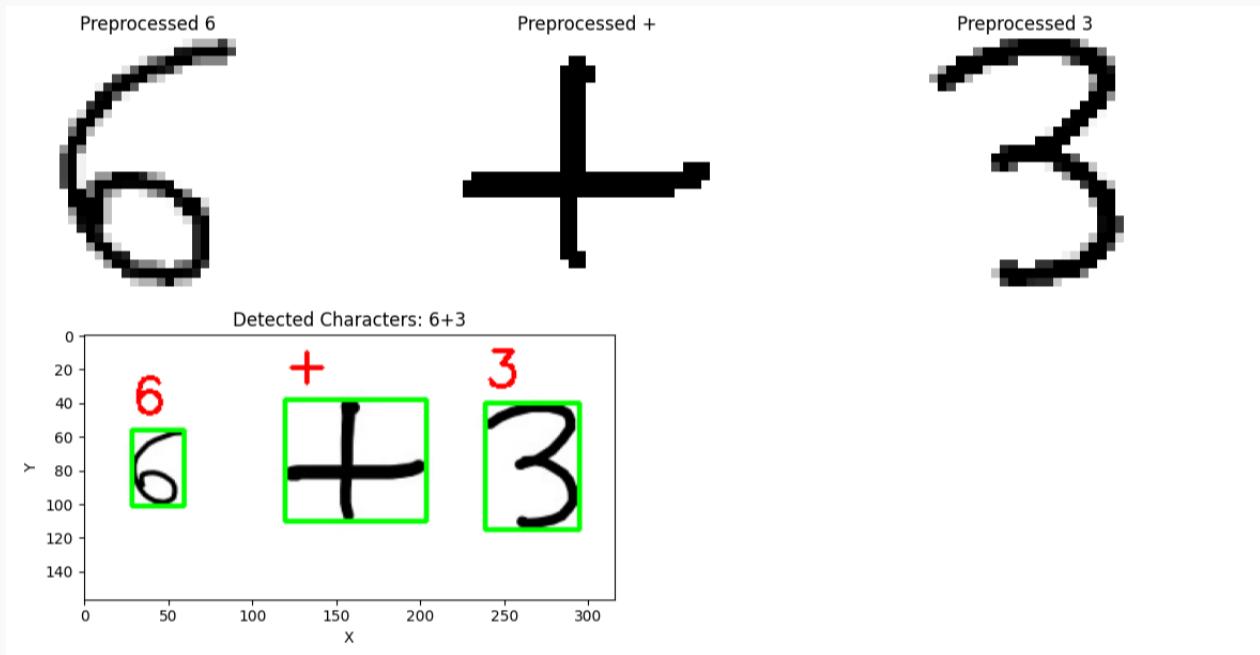


Object detection

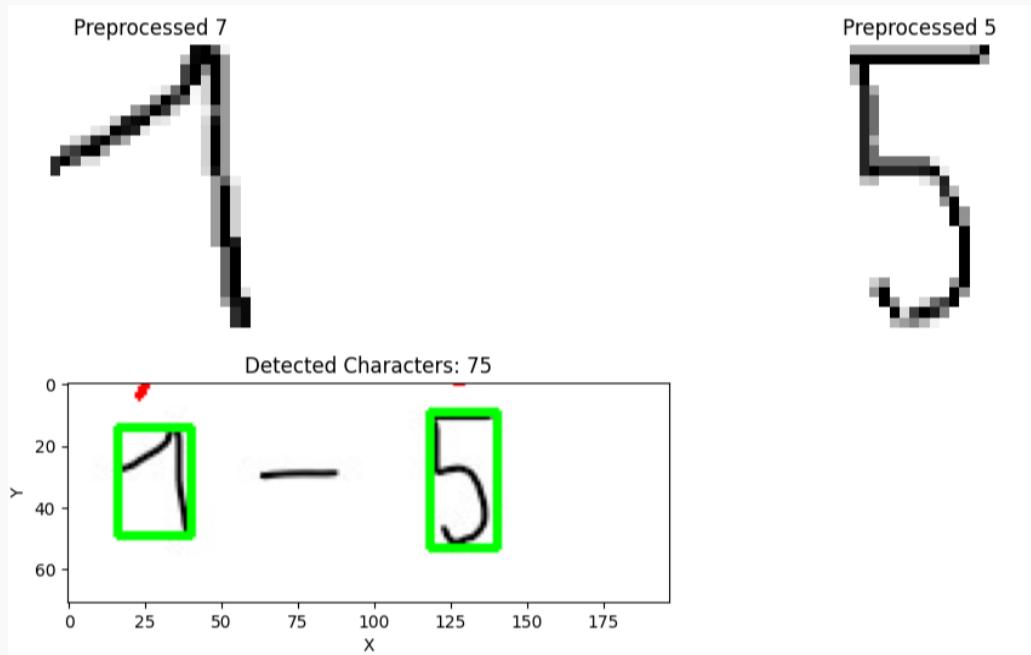
Detected Characters: 888



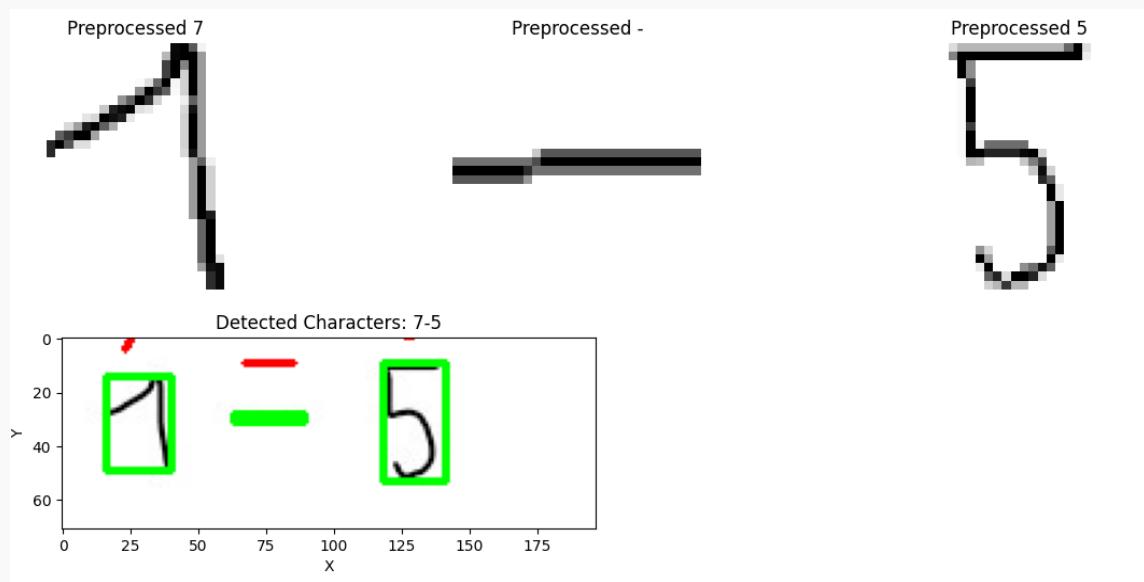
Object Detection Improvement



Detection Error



Object Detection Improvement



Object Detection Improvement

- The Detection has to be improved step by step

$52 - 14 =$

$52 - 14 =$

$52 - 14 =$

$52 - 14 =$

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



PaddleOCR

CER*: 19%



EasyOCR

CER*: 12%



TrOCR

Tesseract

CER*: 38%

TrOCR

CER*: 15%

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

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

Steps



TEXT

45 x 12

TrOCR

Transformer-based Optical Character Recognition

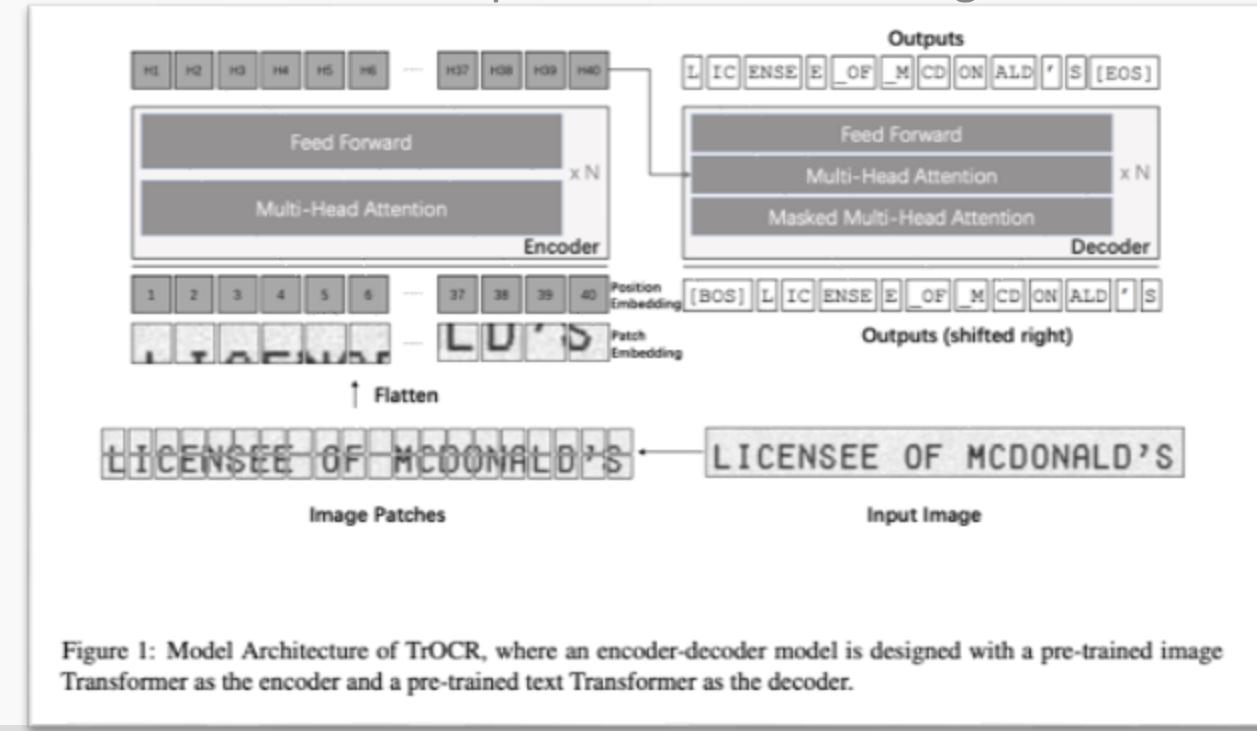


Figure 1: Model Architecture of TrOCR, where an encoder-decoder model is designed with a pre-trained image Transformer as the encoder and a pre-trained text Transformer as the decoder.

TrOCR

Transformer-based Optical Character Recognition

Pre-Trained
(stage 1)

Small
Base
Large

Handwritten

Small
Base
Large

Printed

Small
Base
Large

Scene

Small
Base

TrOCR

Transformer-based Optical Character Recognition

Pre-Trained
(stage 1)

Small
Base
Large

Handwritten

Small
Base
Large

Printed

Small
Base
Large

Scene

Small
Base

TrOCR

Transformer-based Optical Character Recognition

Pre-Trained
(stage 1)

Small

Base

Large

Handwritten

Small

Base

Large

Printed

Small

Base

Large

Scene

Small

Base

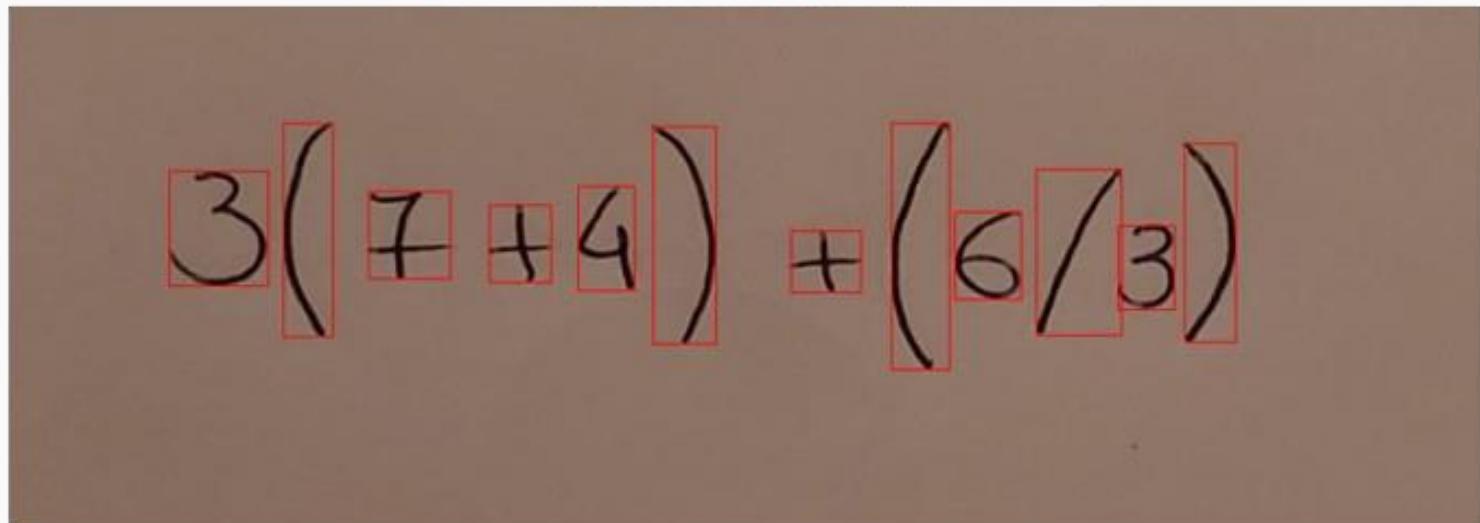
Extracted Text: $3(7+4)+(6/3)$

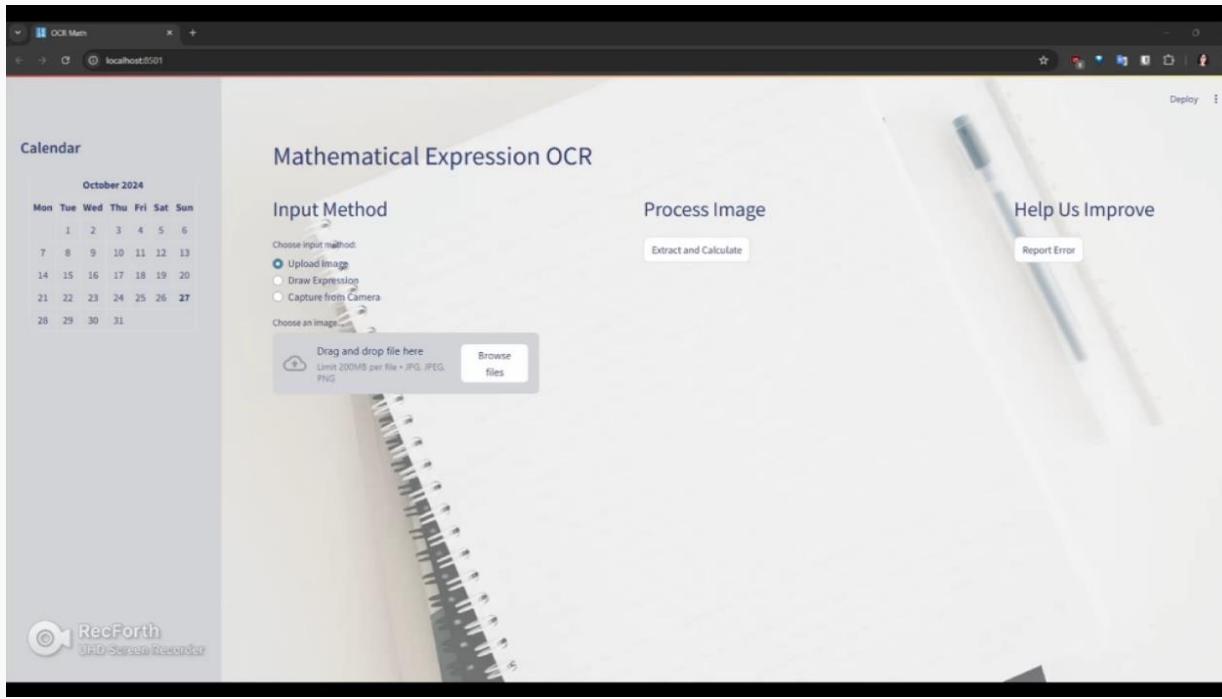
Calculated Result: 35.0

Accuracy: 100.00%

TrOCR Result with Bounding Boxes

Extracted Text: $3(7+4)+(6/3)$





Demo

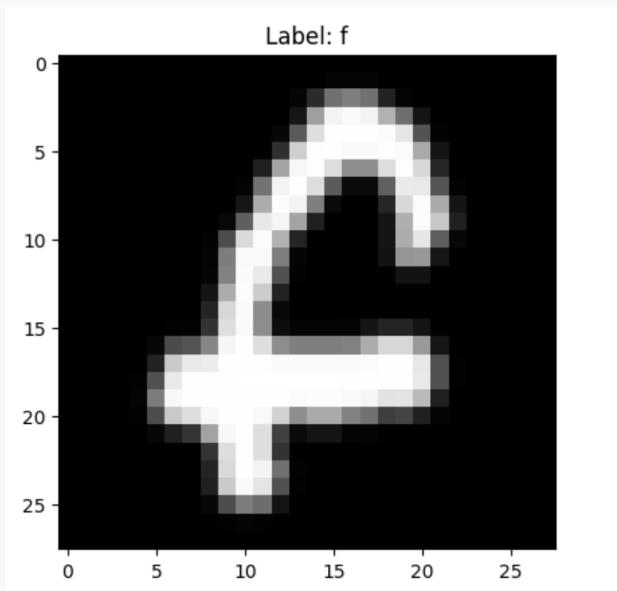
04

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

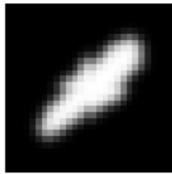
```
# Simple CNN architecture
model1 = models.Sequential([
    layers.Conv2D(16, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),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

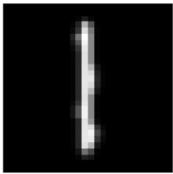
Test accuracy: 0.9060071706771851
```

Misclassification

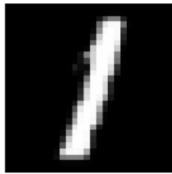
True: I
Pred: L



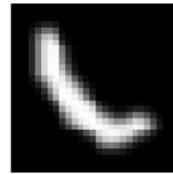
True: I
Pred: L



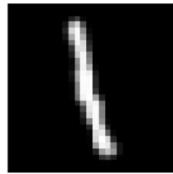
True: I
Pred: L



True: I
Pred: L



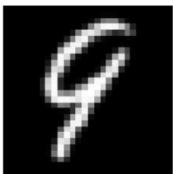
True: I
Pred: L



True: G
Pred: Q



True: G
Pred: Q



True: G
Pred: Q



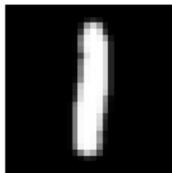
True: G
Pred: Q



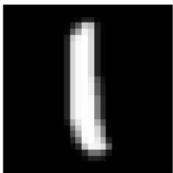
True: G
Pred: Q



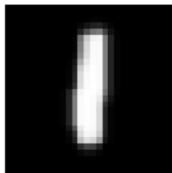
True: L
Pred: I



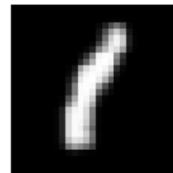
True: L
Pred: I



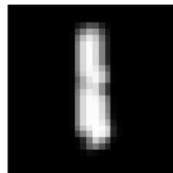
True: L
Pred: I



True: L
Pred: I



True: L
Pred: I



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

True: Q
Pred: G



True: Q
Pred: G



True: Q
Pred: G



True: Q
Pred: G



True: Q
Pred: G



True: Q
Pred: Q



True: Q
Pred: G



True: Q
Pred: G



True: Q
Pred: G



True: G
Pred: Q



True: G
Pred: Q



True: G
Pred: Q



True: G
Pred: Q



True: G
Pred: Q



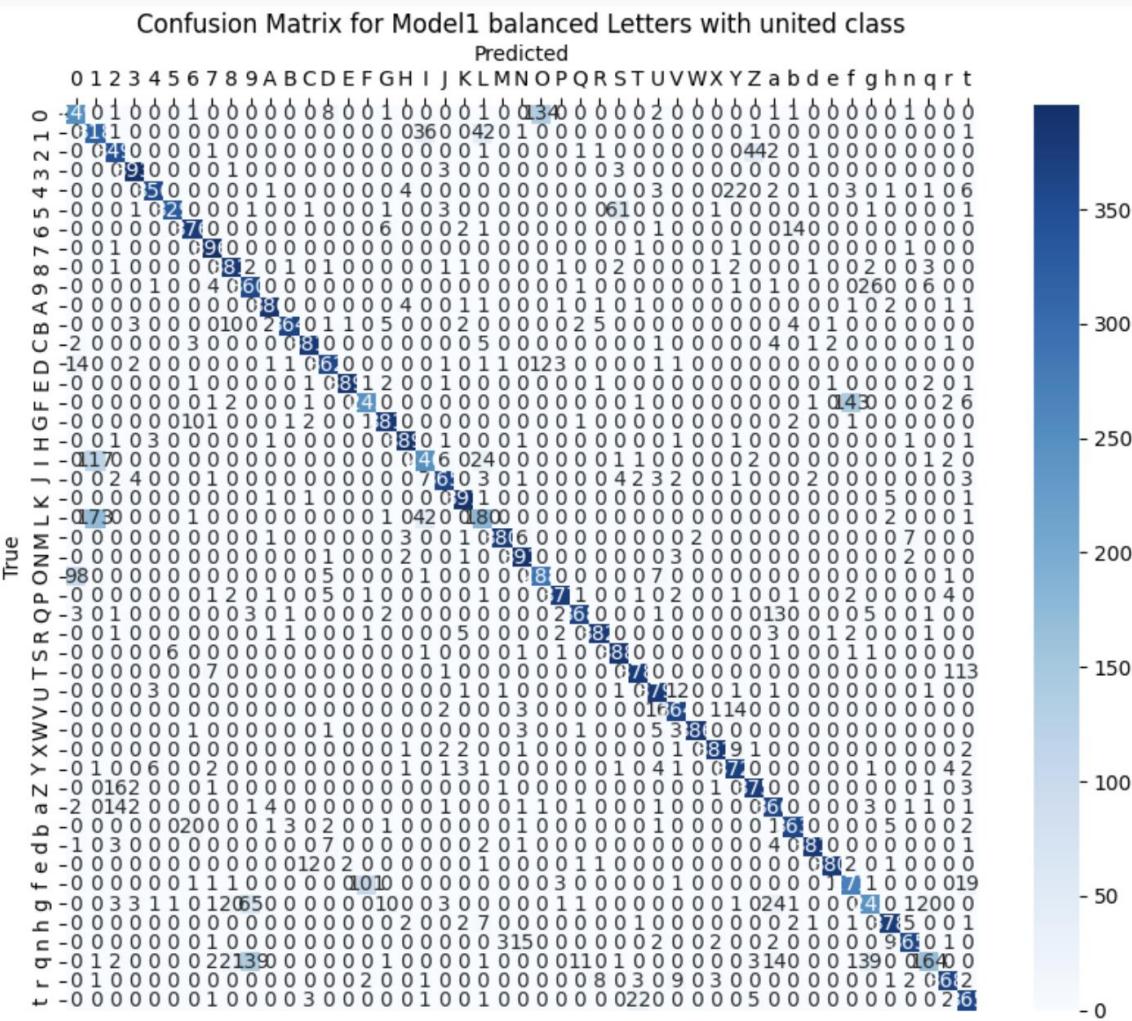
Separate labels for Upper & lower case

```
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%

Confusion Matrix for Model1 balanced Letters

Improvements

More epochs

Learning rate

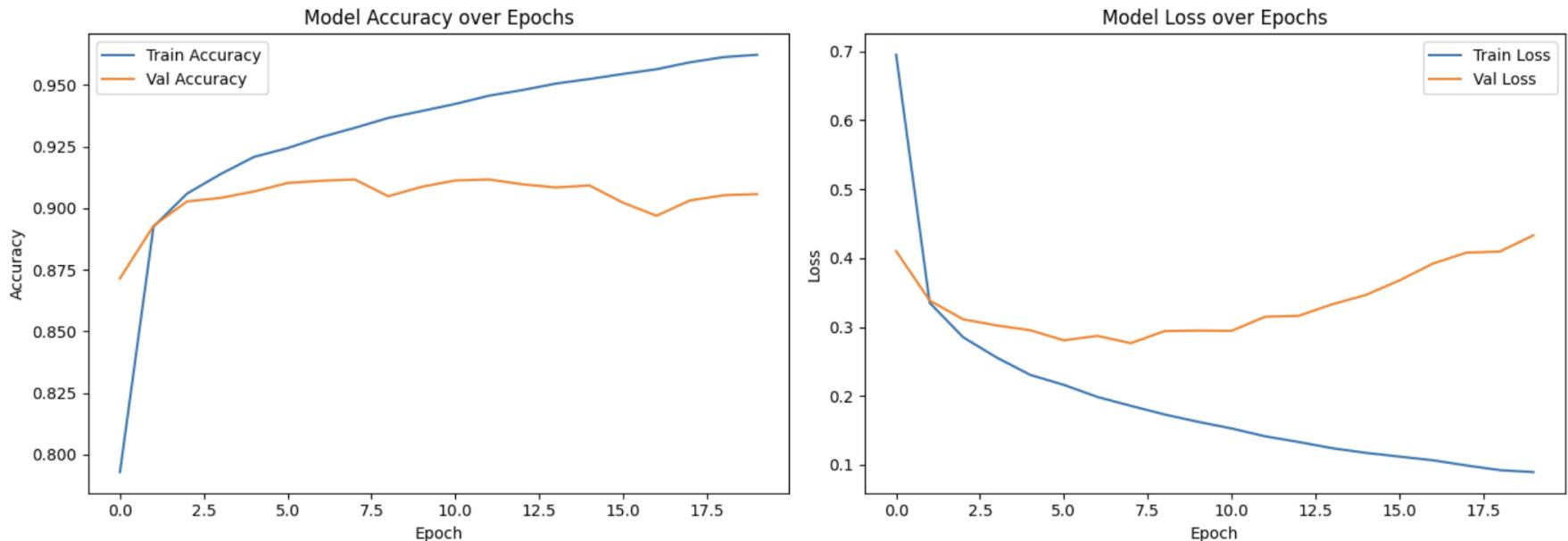
Augmentation of data

Another optimizer

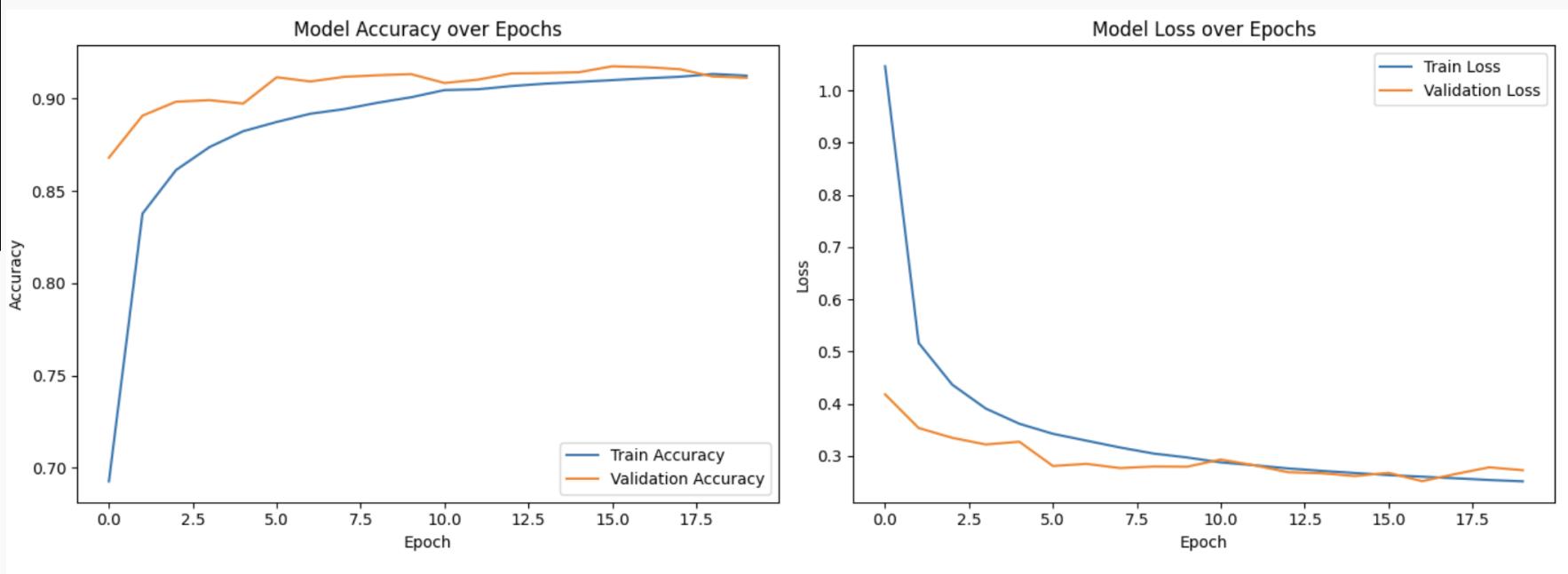
Early stopping

More complex model
architecture

More epochs



Data augmentation



More complex model

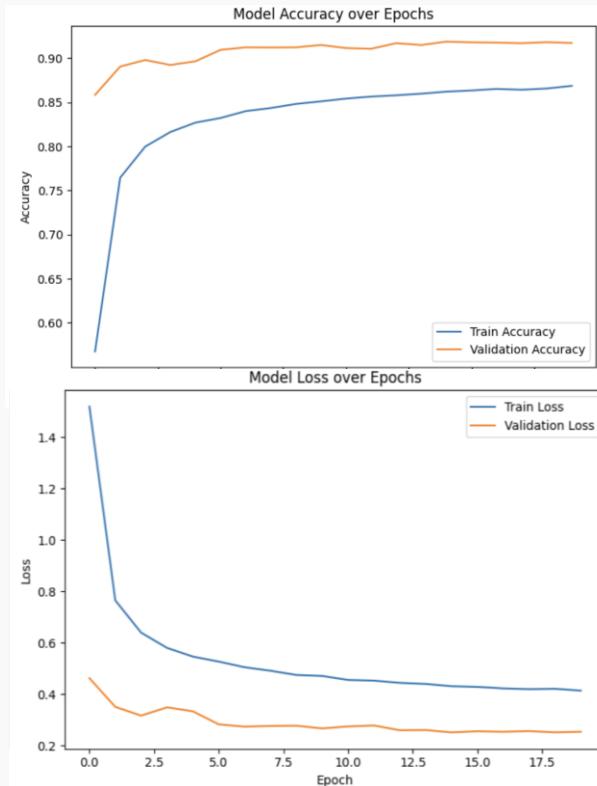
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
batch_normalization (BatchNormalization)	(None, 26, 26, 32)	128
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
dropout (Dropout)	(None, 13, 13, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	18,496
batch_normalization_1 (BatchNormalization)	(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)

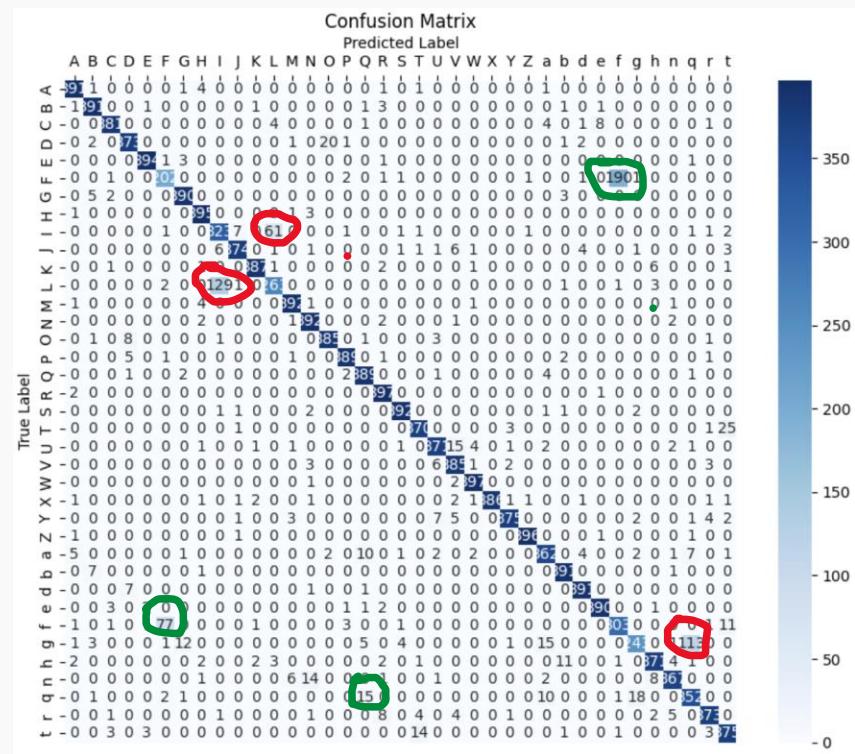
Non-trainable params: 704 (2.75 KB)

Optimizer params: 2 (12.00 B)



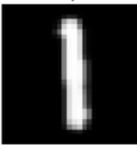
Putting all together

	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

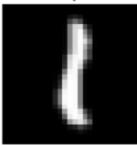


Remained misclassifications

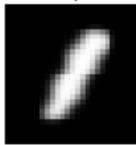
True: L | Pred: I



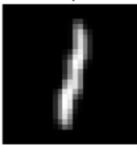
True: L | Pred: I



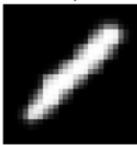
True: L | Pred: I



True: L | Pred: I



True: L | Pred: I



Misclassified g as q:

True: g | Pred: q



True: g | Pred: q



True: g | Pred: q



True: g | Pred: q



True: g | Pred: q



Misclassified D as 0:

True: D | Pred: O



True: D | Pred: O



True: D | Pred: O



True: D | Pred: O



True: D | Pred: O



Misclassified Q as a:

True: Q | Pred: a



True: Q | Pred: a



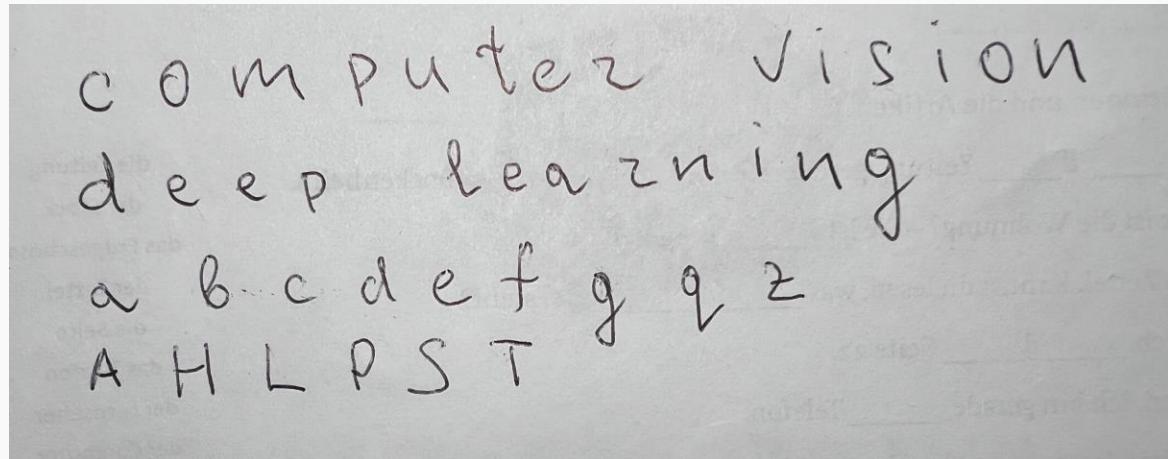
True: Q | Pred: a



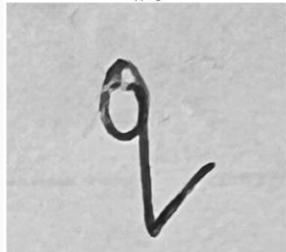
True: Q | Pred: a



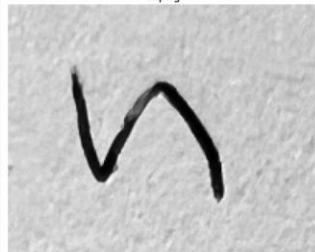
Recognition of own letters on paper



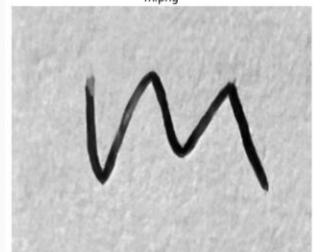
q.png



n.png



m.png



Preprocessing

1

computer vision
deep learning
a b c d e f g q z
A H L P S T

2

computer vision
deep learning
a b c d e f g q z
A H L P S T

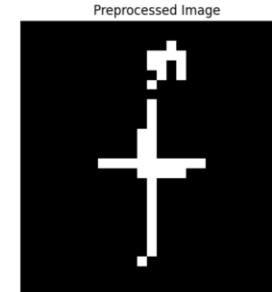
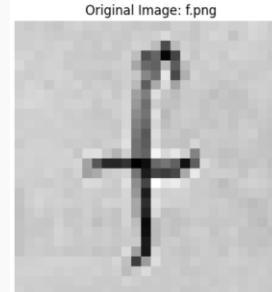
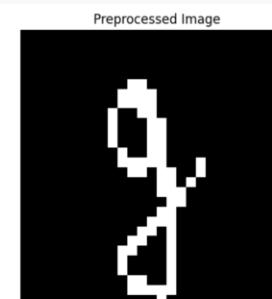
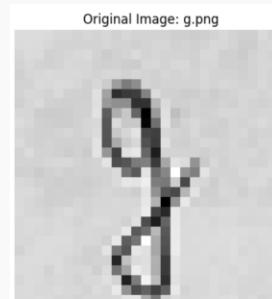
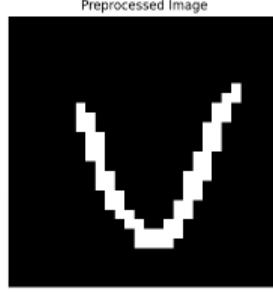
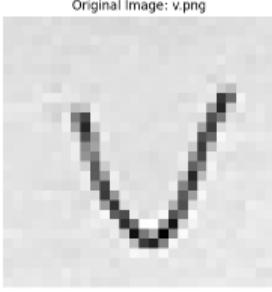
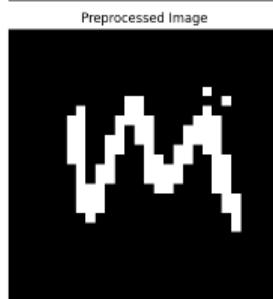
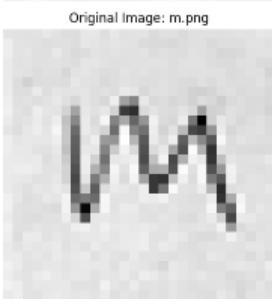
3

computer vision
deep learning
a b c d e f g q z
A H L P S T

4

computer vision
deep learning
a b c d e f g q z
A H L P S T

Letters on PAPER

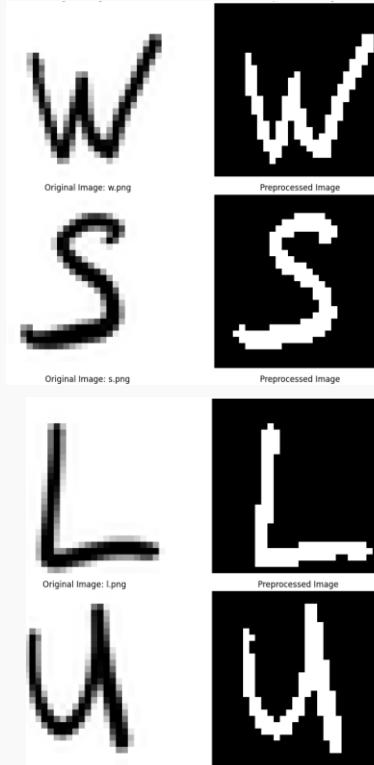


Letters on PAPER

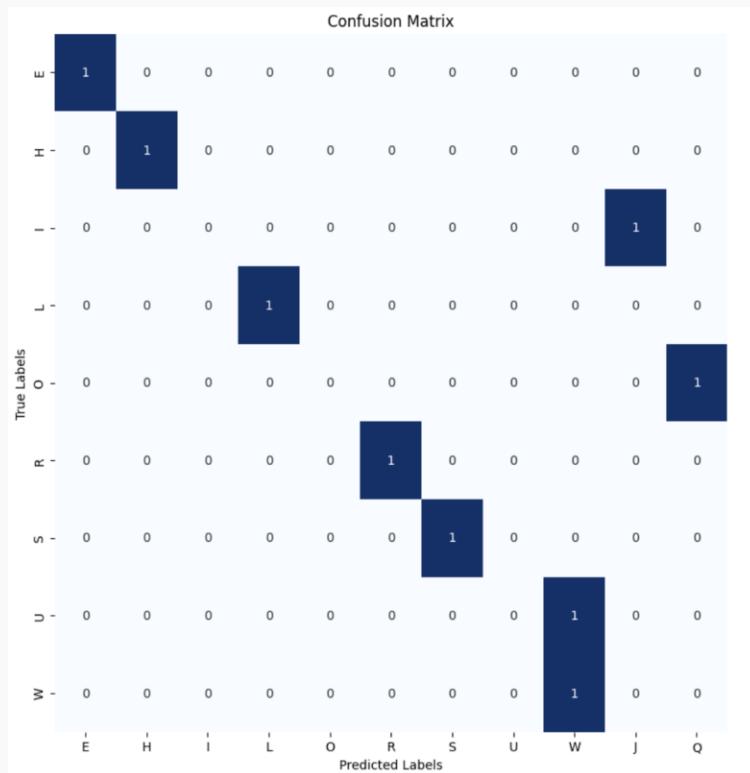
Confusion Matrix														
True Labels	e	q	F	W	E	P	B	q	G	M	V	J	S	M
E -	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G -	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F -	0	0	1	0	0	0	0	0	0	0	0	0	0	0
D -	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Z -	1	0	0	0	0	0	0	0	0	0	0	0	0	0
P -	0	0	0	0	0	1	0	0	0	0	0	0	0	0
O -	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q -	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C -	0	1	0	0	0	0	0	0	0	0	0	0	0	0
M -	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Y -	0	0	0	0	0	0	0	0	0	0	1	0	0	0
--	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S -	0	0	0	0	0	0	0	0	0	0	0	1	0	0
N -	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Recognition of own letters on iPad

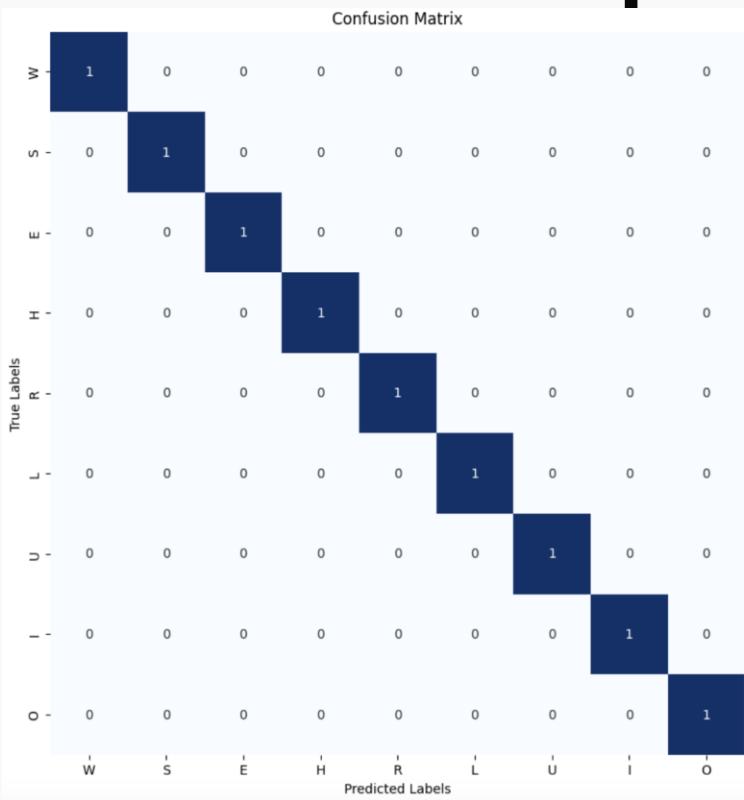
HELLO
WORLD
HSLU
hello
world
и π φ ц ѿ
ö ß
5 * !!! 😊



Performance with the "best" model



Performance with **simple** model



Other symbols: Cyrillic, German,



Result of classifying other symbols

True: н
Pred: **n**



True: ö
Pred: **B**



True: star
Pred: **K**



True: 5
Pred: **G**



True: smile
Pred: **Q**



True: и
Pred: **n**



True: ц
Pred: **U**



True: ф
Pred: **P**



True: ш
Pred: **W**



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

vision

vision

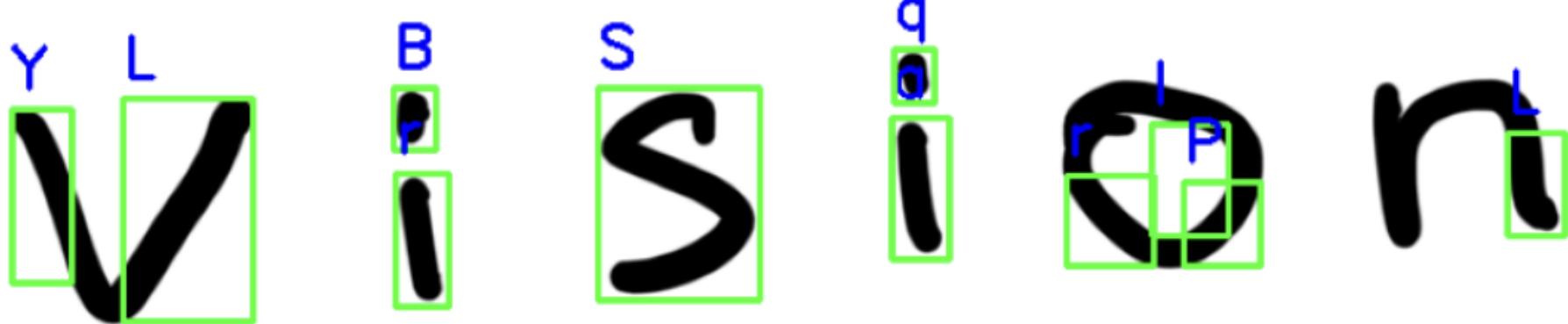
Detected regions

W i S i O n

A horizontal sequence of the letters 'W', 'i', 'S', 'i', 'O', and 'n' in a bold, black, sans-serif font. Each letter is enclosed within a thin green rectangular border, representing a detected region or bounding box. The letters are evenly spaced, and the entire word is centered horizontally.

Classified "letters"

Detected and Classified Letters



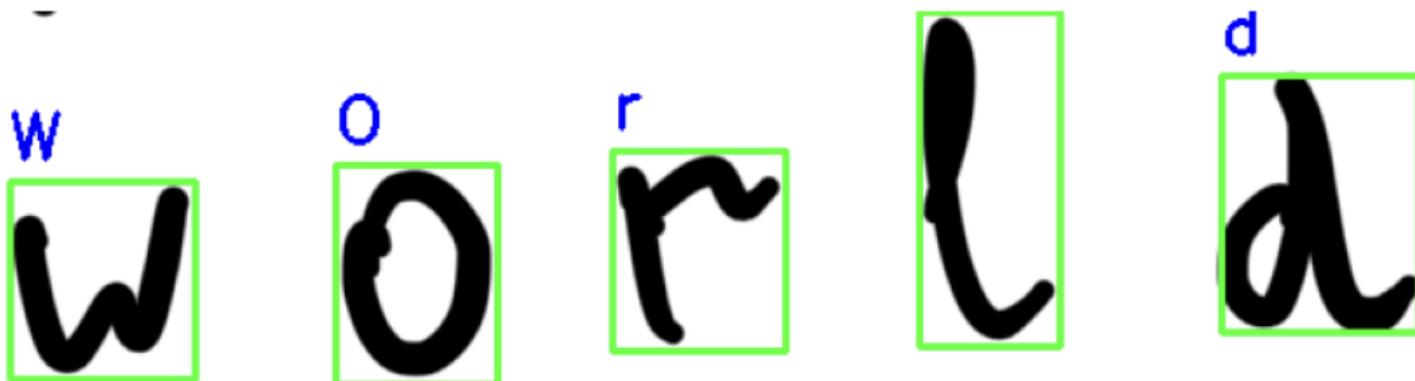
Objects detected in "world"

Detected Letters with Sorted Bounding Boxes



Classified "world"

Detected and Classified Letters



Classified "WORLD"

Detected and Classified Letters



Detected Word: W0rLd

!!!

Translation

WOrLd:

German: Welt

Ukrainian: СВІТ

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

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

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

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

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