# OCR
# Handwriting Recognition

Youssef Chaabouni, Alexandre Misrahi, and Tim Valencony

# Motivations
## Why OCR?

- Store handwritten papers:
  - Fast access
  - Fast search
  - Fast sharing
  - Analysis
- Preservation of knowledge
- Document management

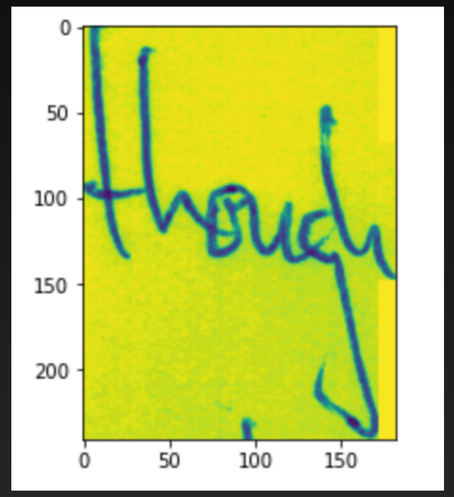
# Data

## Description & Preprocessing

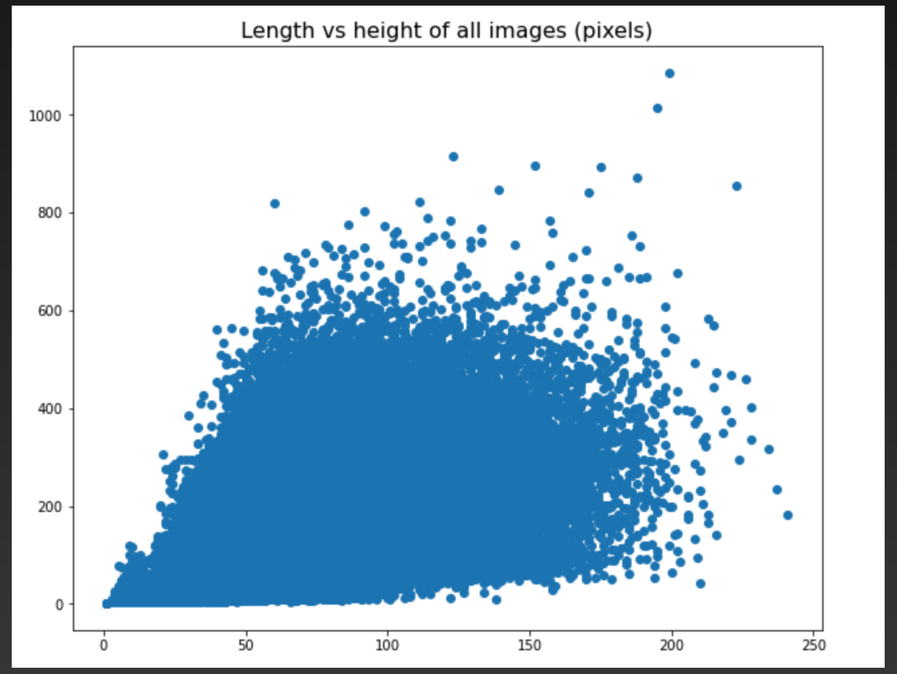
- IAM Database
- 115,320 words written by 657 different authors

# Data

## Description & Preprocessing




```
Longest word has 21 characters: ' plate-and-corrugation '
```

# Preprocessing

- Rotation (from one dataset to another)
- Reshaping (after segmentation)
- Background uniformisation (weird pixels in background in IAM database)
- Background noise addition (in character model for better generalisation)
- Edge finding (for segmentation)
- Contrast accentuation (EMNIST database)

# 1. First approach:
# *segment-and-decode*

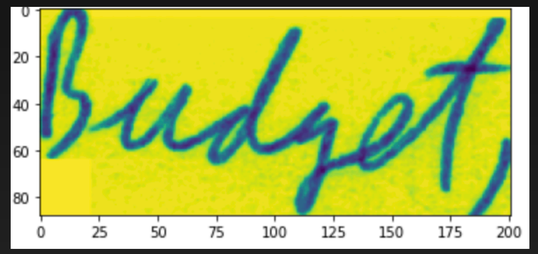
# Implementation

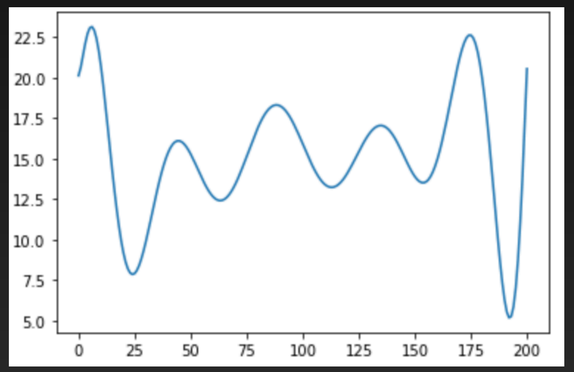
- We process as follows: segmentation of the character in the word + prediction of each character individually
- Step 1: Segmentation
- Step 2: Character prediction using CNN

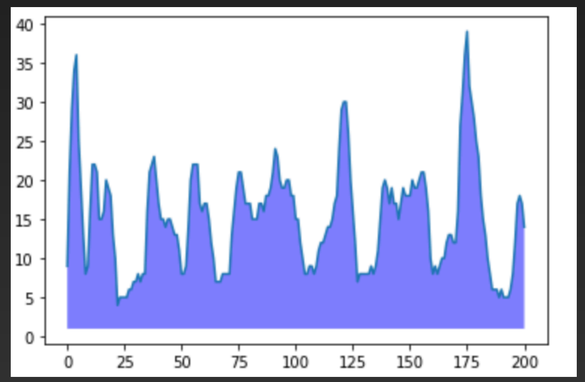
# Step 1: Segmentation

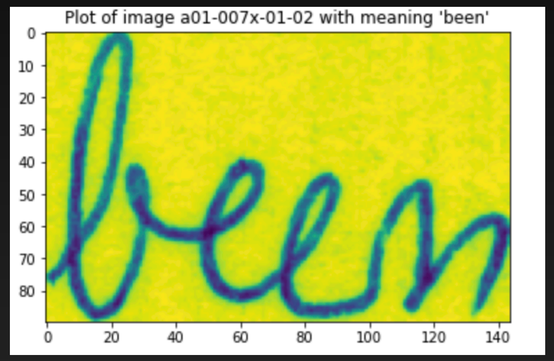
## Extracting Information from images

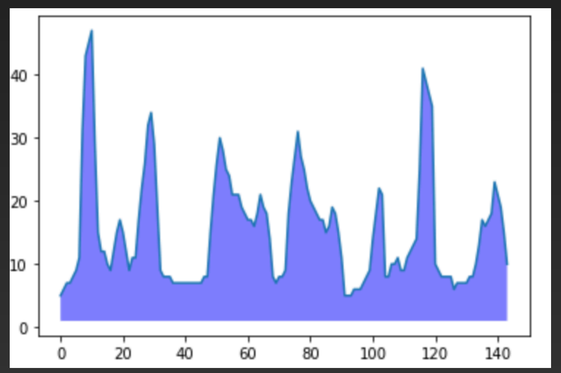
- Examples: “Budget” and “been”

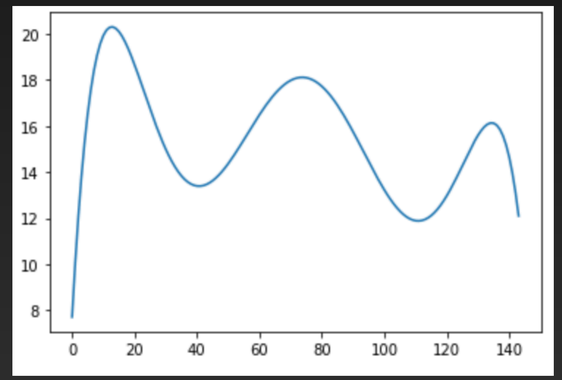








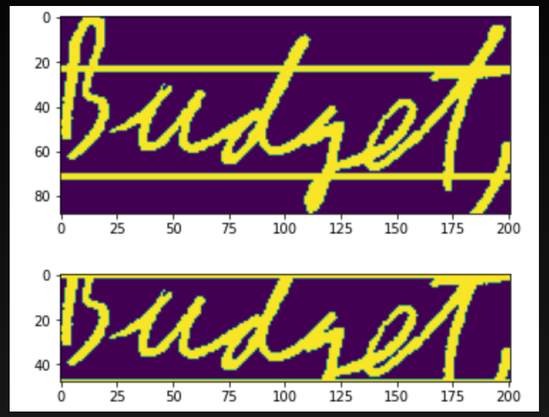



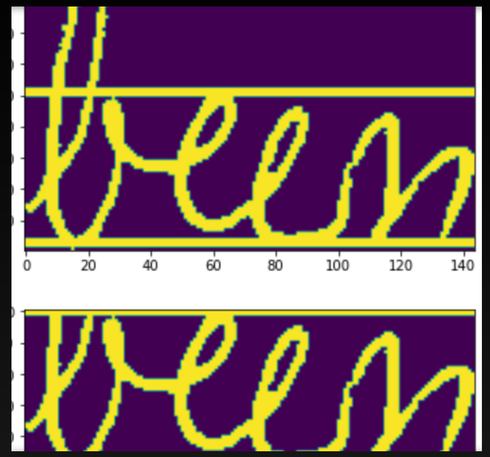


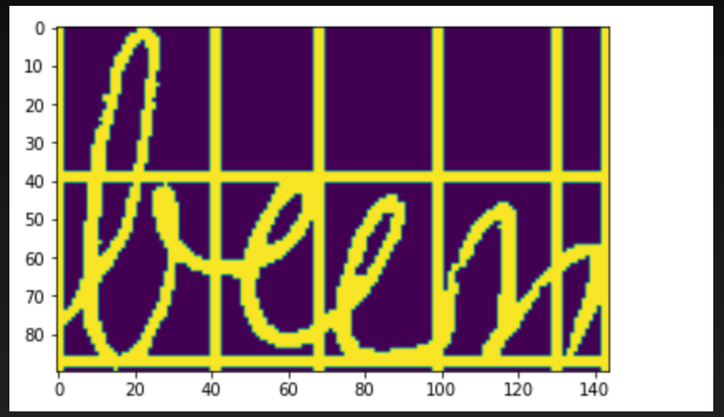
- Plots of maximum pixels per column and its fitted curve
- Find *low points* and *local minima*

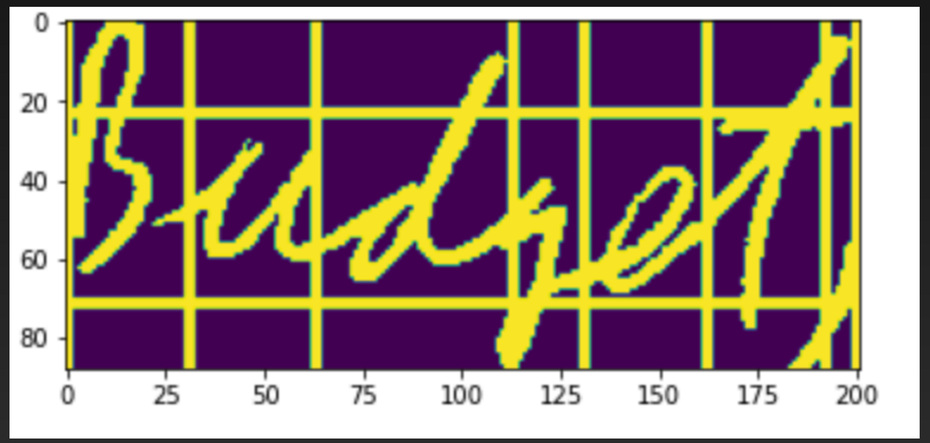
# Step1: Segmentation

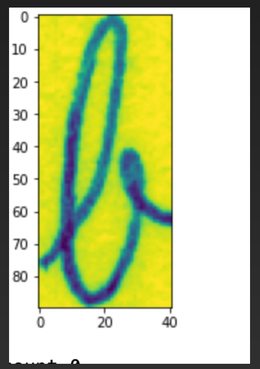
## Cutting

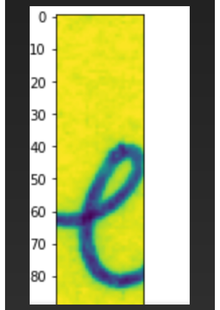


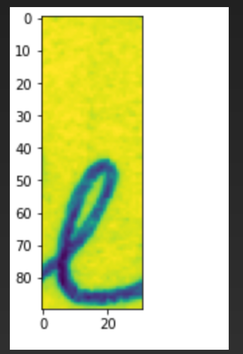


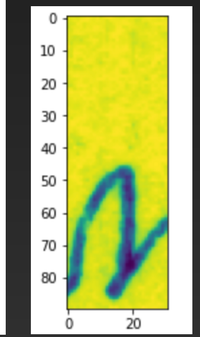


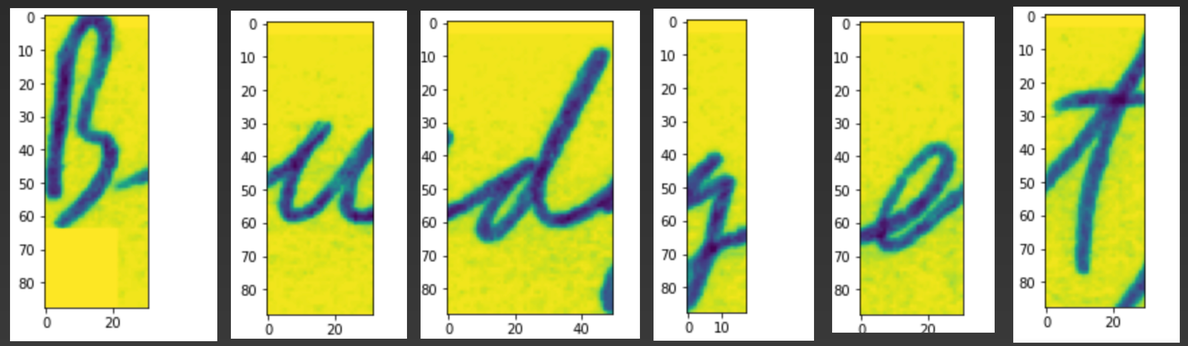






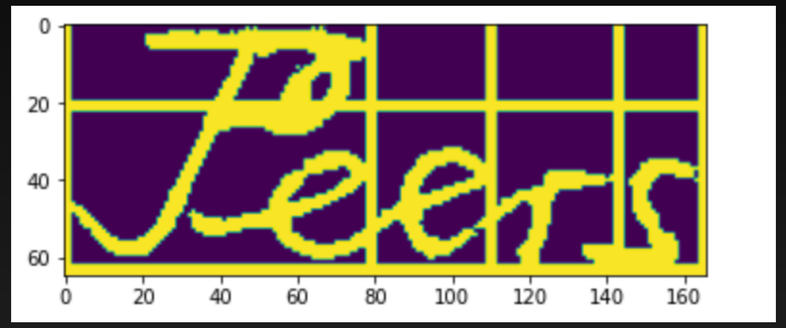


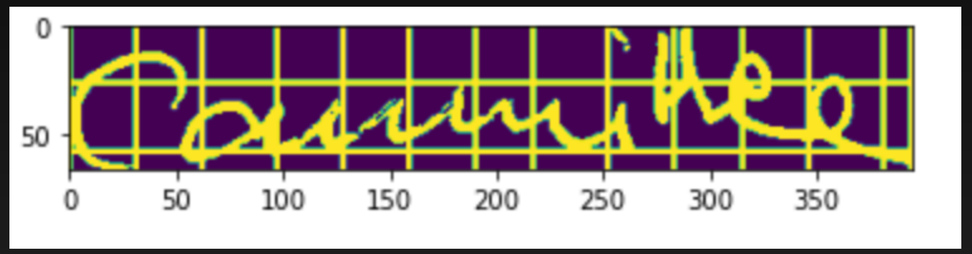


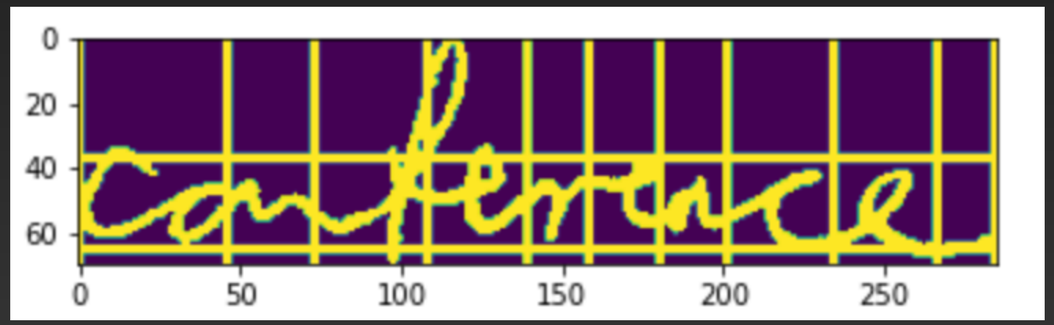


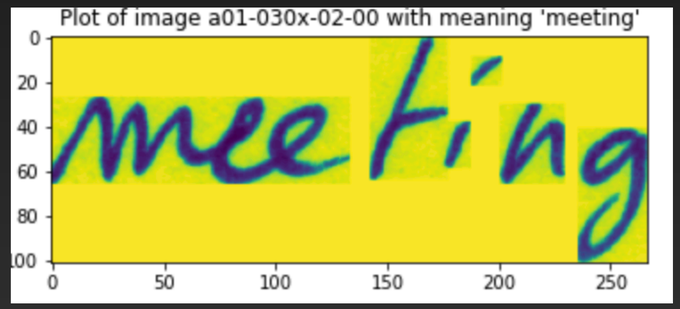
# Results

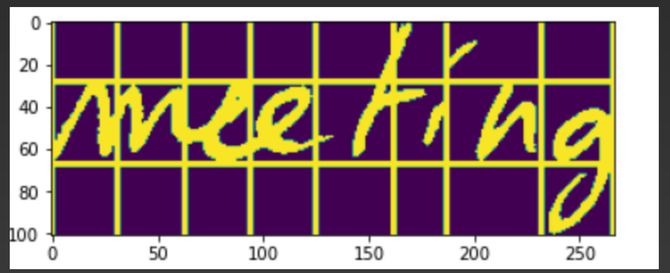
## Not very precise

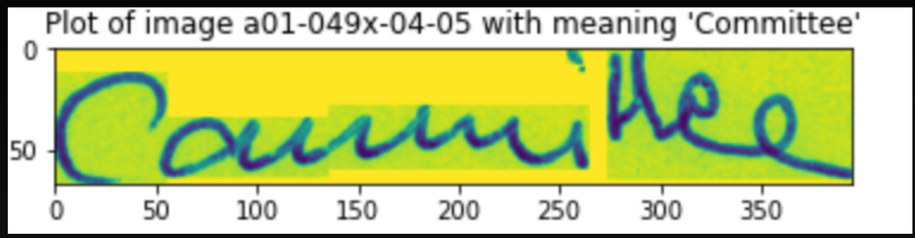






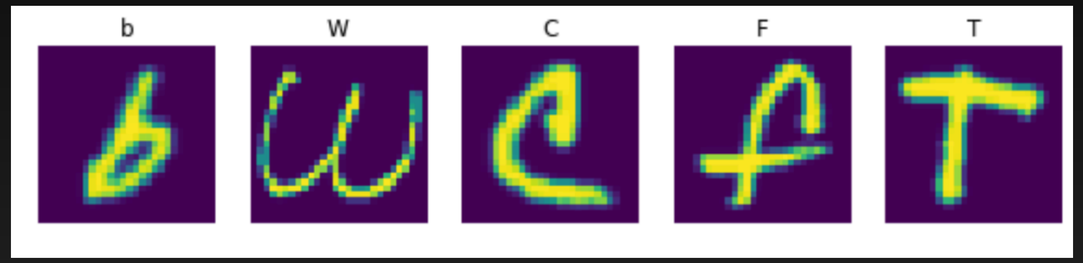


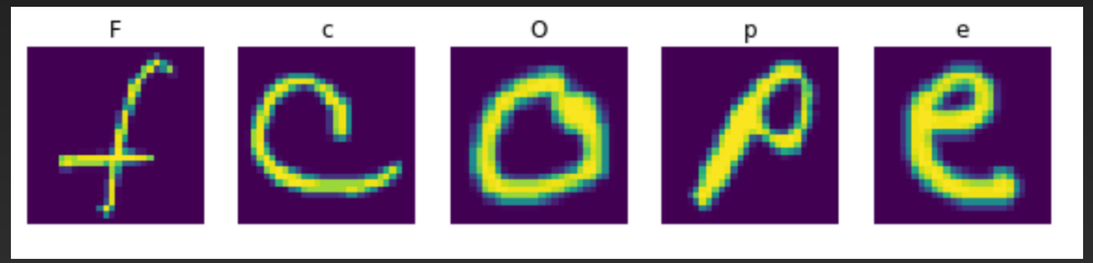





# Step2: Character Prediction

- Balanced EMNIST Dataset



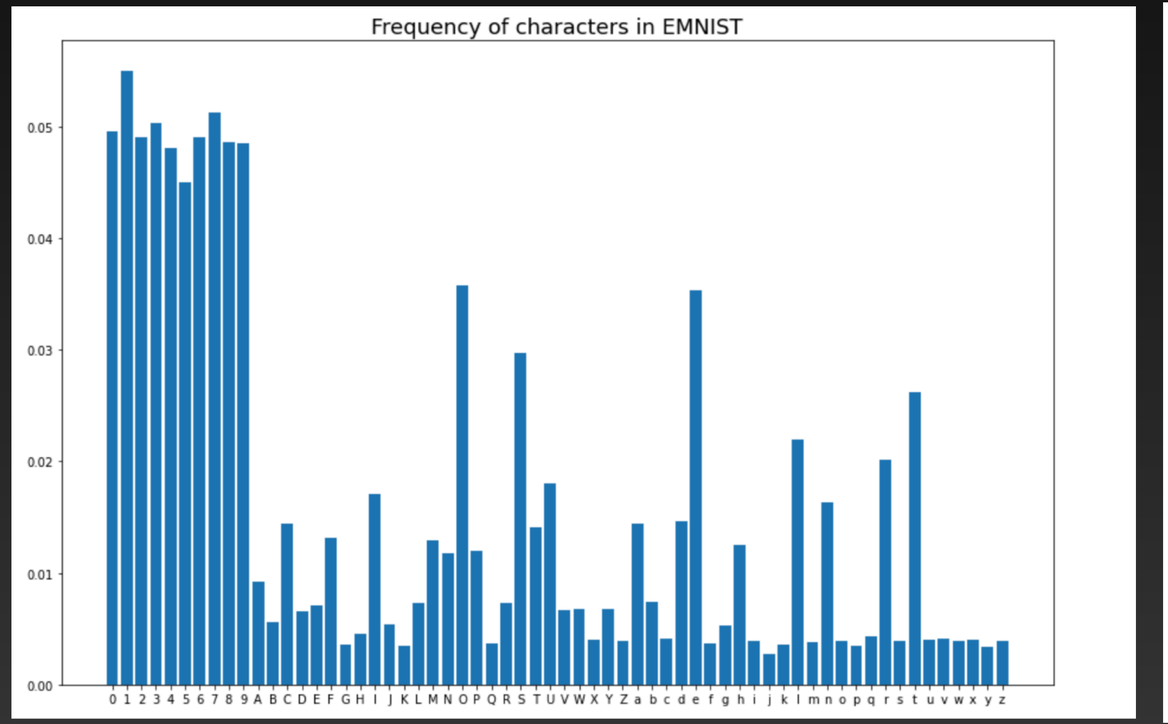



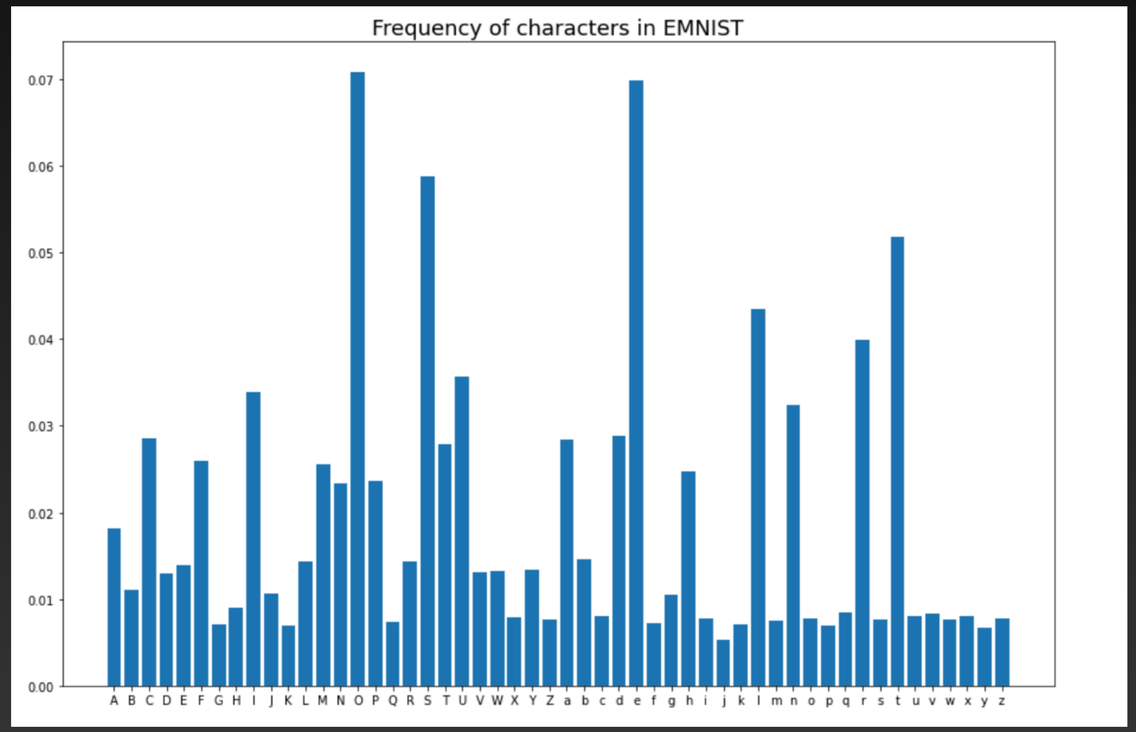

Model: "sequential"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d (Conv2D) | (None, 30, 30, 32) | 320 |
| max_pooling2d (MaxPooling2D) | (None, 15, 15, 32) | 0 |
| dropout (Dropout) | (None, 15, 15, 32) | 0 |
| conv2d_1 (Conv2D) | (None, 15, 15, 64) | 18496 |
| max_pooling2d_1 (MaxPooling2D) | (None, 7, 7, 64) | 0 |
| conv2d_2 (Conv2D) | (None, 5, 5, 128) | 73856 |
| max_pooling2d_2 (MaxPooling2D) | (None, 2, 2, 128) | 0 |
| dropout_1 (Dropout) | (None, 2, 2, 128) | 0 |
| batch_normalization (BatchNormalization) | (None, 2, 2, 128) | 512 |
| flatten (Flatten) | (None, 512) | 0 |
| dense (Dense) | (None, 128) | 65664 |
| dropout_2 (Dropout) | (None, 128) | 0 |
| batch_normalization_1 (BatchNormalization) | (None, 128) | 512 |
| dense_1 (Dense) | (None, 64) | 8256 |
| dense_2 (Dense) | (None, 52) | 3380 |

# Step2: Character Prediction

- Attempts at better generalisation (1) : removing digits



# Step2: Character Prediction

- Attempts at better generalisation (2) : adding noise

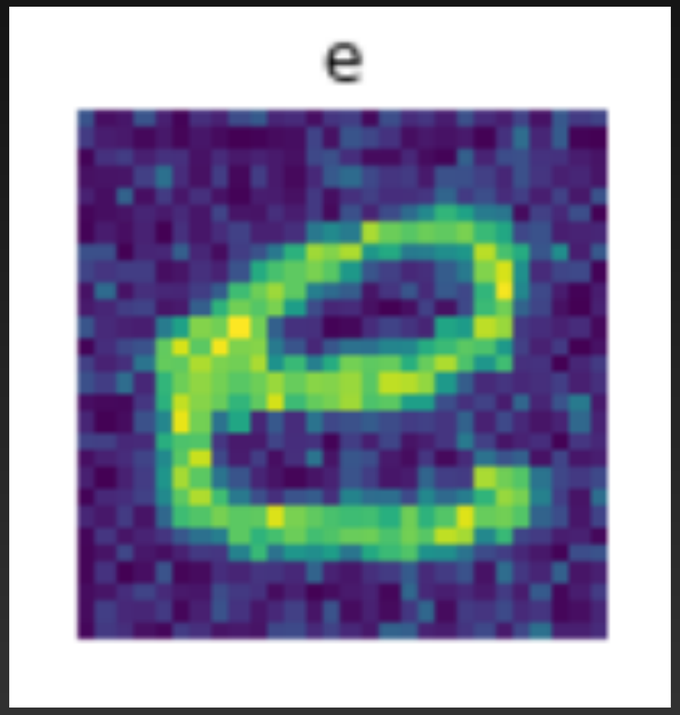



Image + Gaussian(0,0.2)

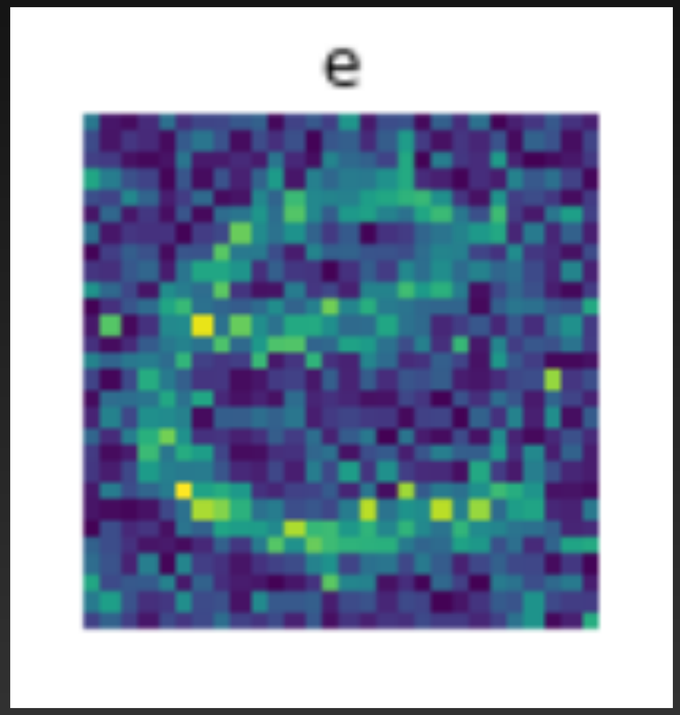



Image + Gaussian(0,0.8)

# Step2: Character Prediction

- 87% testing accuracy
- Accuracy – Generalisation tradeoff

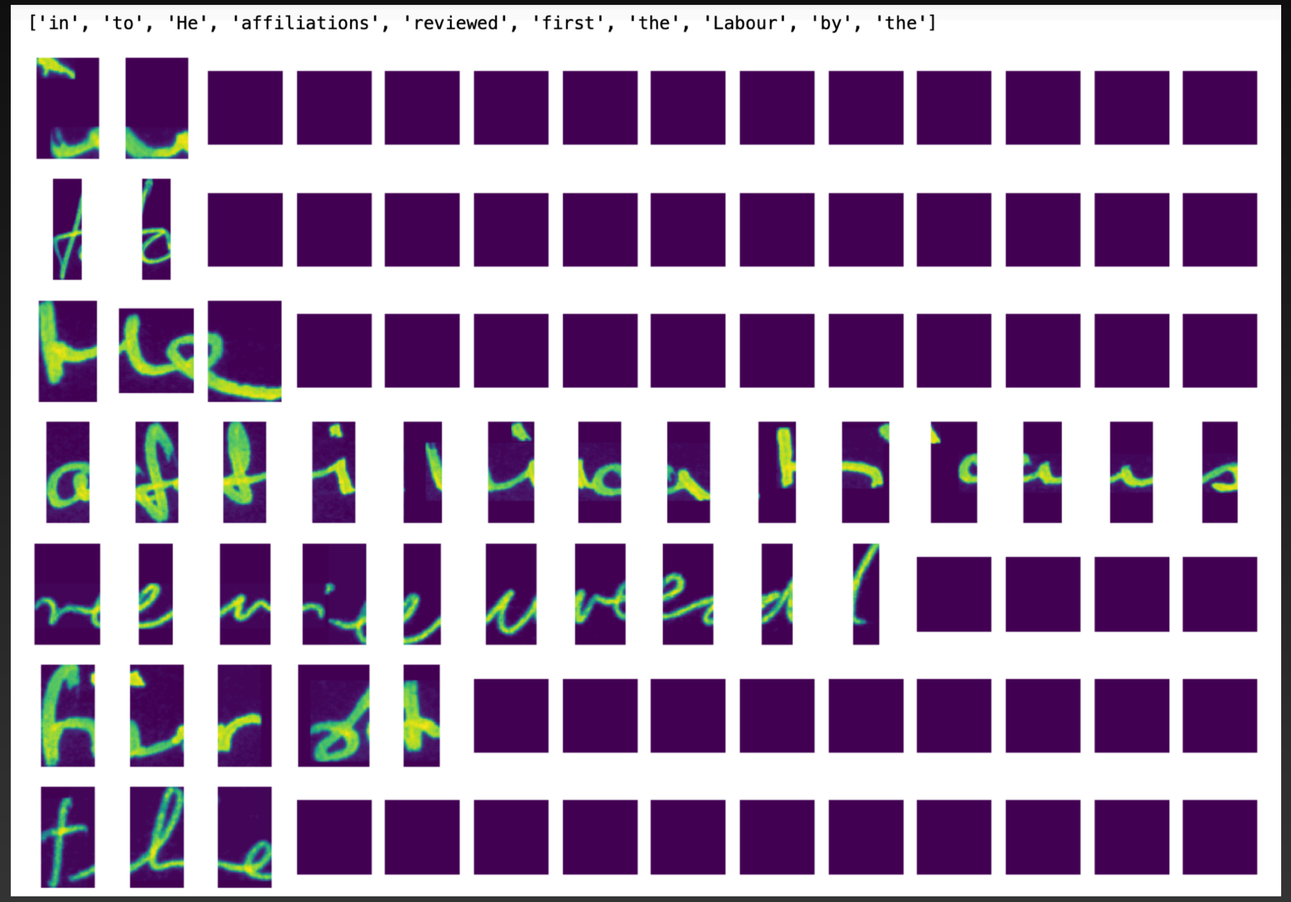
# Results on segmented characters

## Poor predictions

- Examples + Image predictions

```
'in': 'qb', 'to': 'YG', 'He': 'HWU'
```

```
'the': 'YYPY', 'Labour': 'ZUYQYYQZF', 'by': 'EYG'
```




```
'affiliations': 'GQGFYFGWYFYMPk', 'reviewed': 'HEUYEEWYYQ', 'first': 'GYFYB'
```

# Weaknesses
## Why it didn’t work

- Segmentation:
  - One handwriting per individual
  - Cutting more/less characters than there actually are

- Character Prediction using CNN:
  - Different dataset used for training
  - Attached words are harder to predict

- How to overcome this?

# 2. The real deal:
# Connectionist Temporal Classification

# Connectionist Temporal Classification (CTC)

Annotate the images at each horizontal position

Try all the possible alignments of the ground truth label into the image

# The Model

We use a Neural Network to:

1. Extract the sequence features
2. Propagate information through the sequence
3. Tries to align the ground truth label into the recognized character blocks

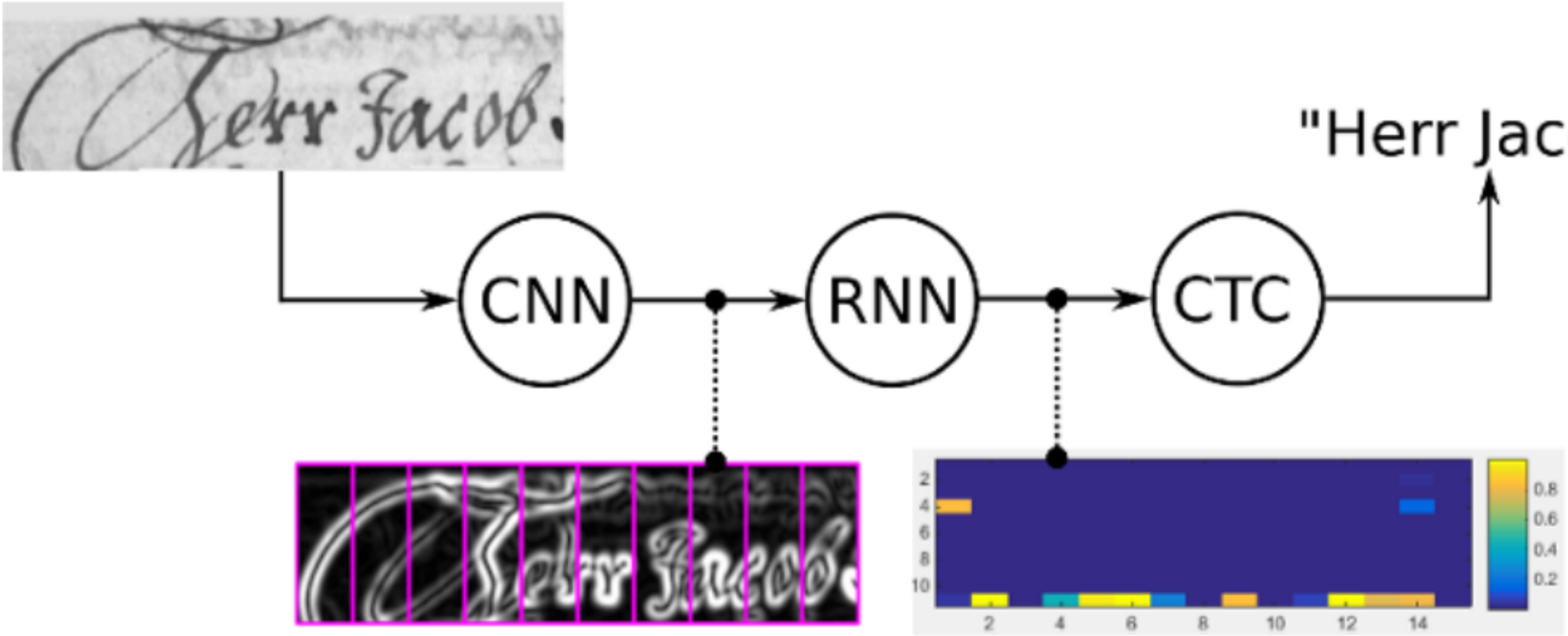
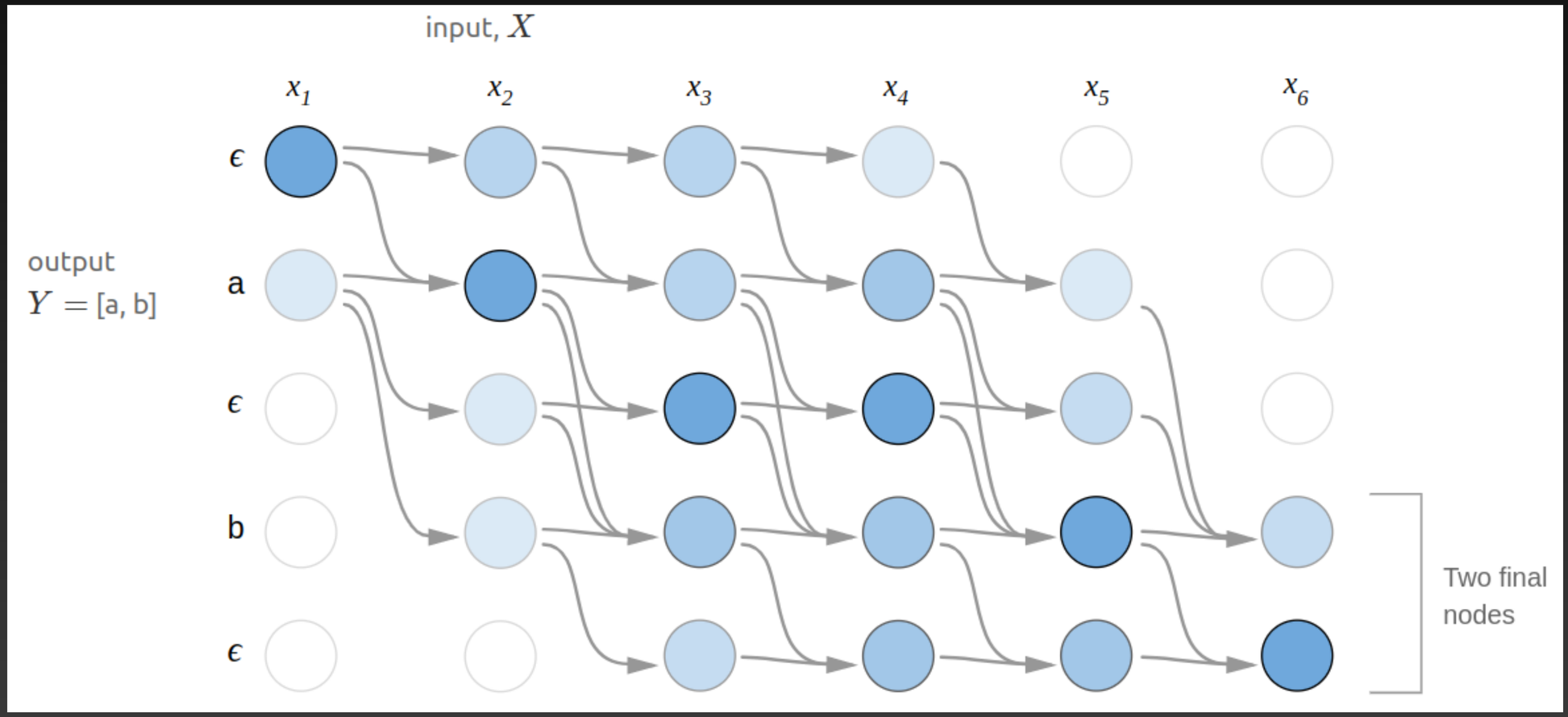



Fig. 1: Overview of a NN for handwriting recognition.

# CTC loss function: The way it works

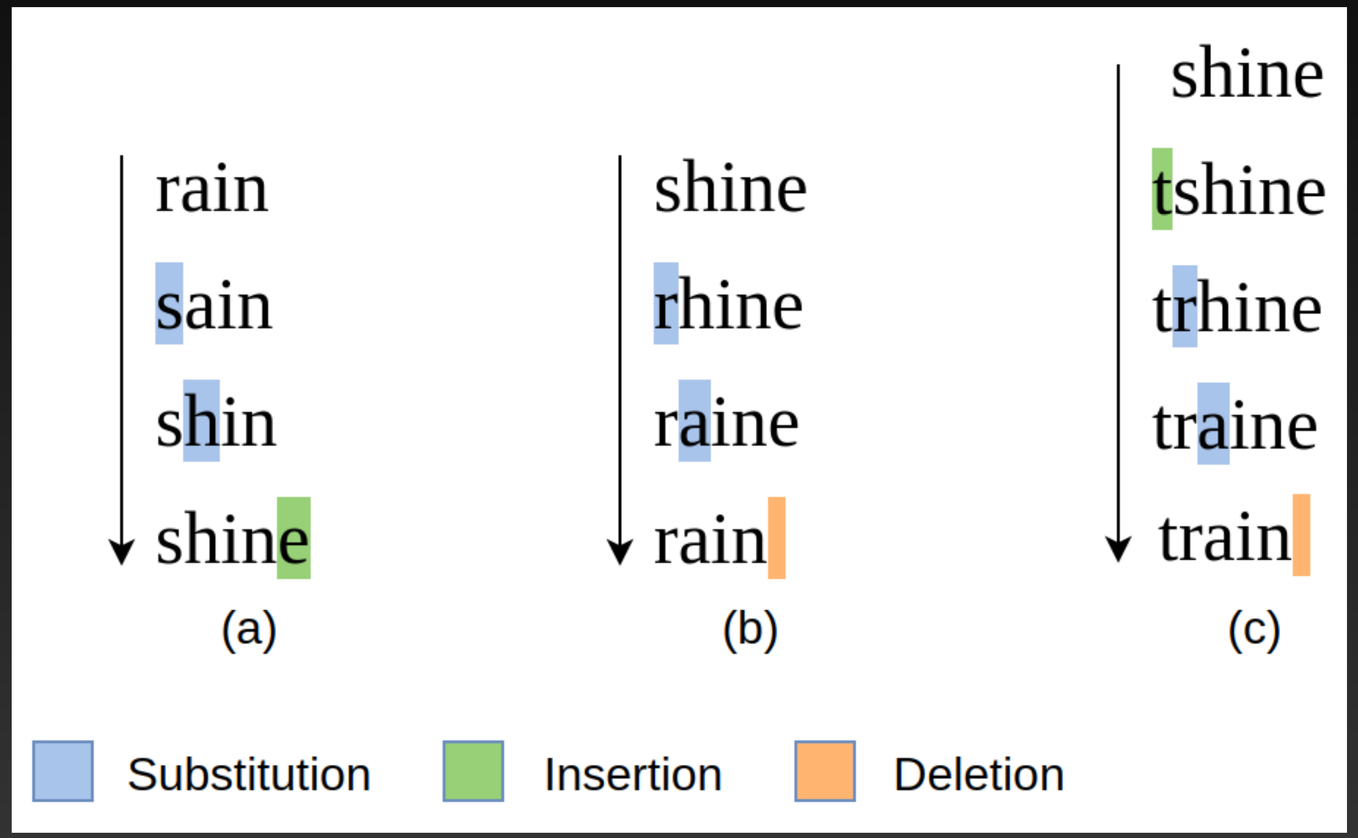
Output matrix of the NN → Sum probabilities of paths corresponding to GT label



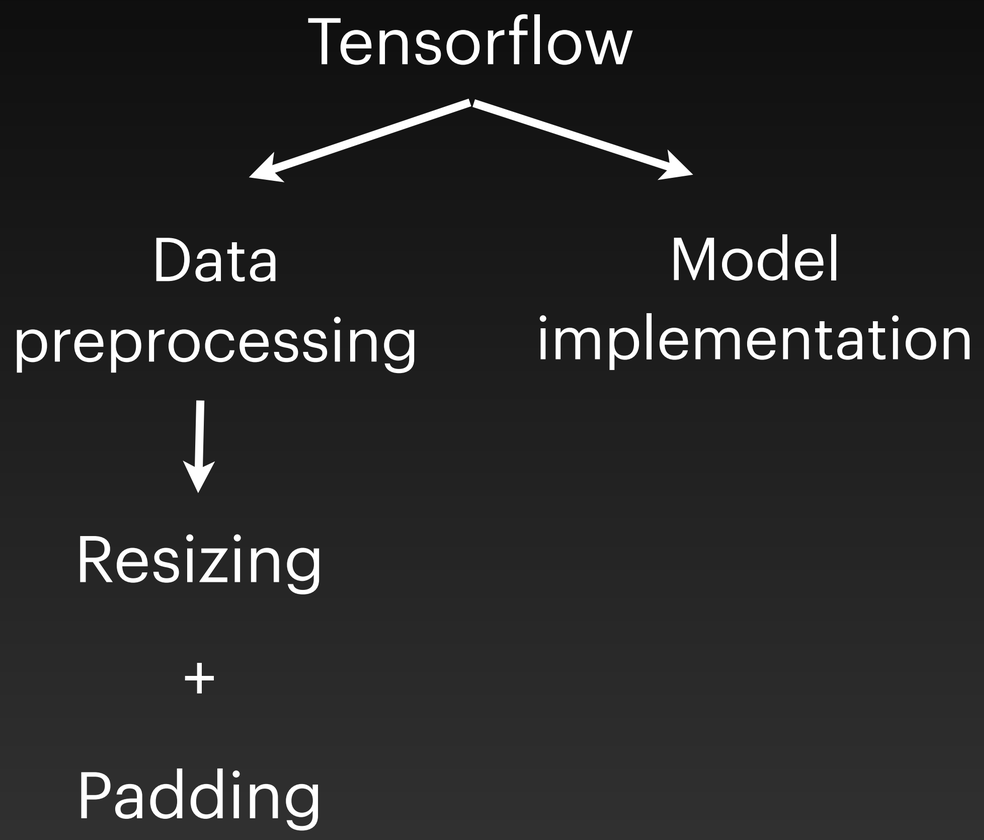
# Error Measure

## Edit Distance

Minimal number of **Substitution**, **Insertion** or **Deletion** operations to obtain a word from an other



# Implementation



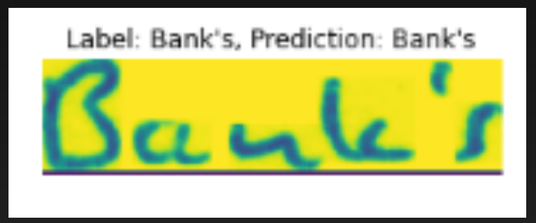

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| sample (InputLayer) | [(None, 128, 32, 1)] | 0 | [] |
| conv2d (Conv2D) | (None, 128, 32, 32) | 320 | ['sample[0][0]'] |
| max_pooling2d (MaxPooling2D) | (None, 64, 16, 32) | 0 | ['conv2d[0][0]'] |
| conv2d_1 (Conv2D) | (None, 64, 16, 64) | 18496 | ['max_pooling2d[0][0]'] |
| max_pooling2d_1 (MaxPooling2D) | (None, 32, 8, 64) | 0 | ['conv2d_1[0][0]'] |
| reshape (Reshape) | (None, 32, 512) | 0 | ['max_pooling2d_1[0][0]'] |
| reduction (Dense) | (None, 32, 64) | 32832 | ['reshape[0][0]'] |
| dropout (Dropout) | (None, 32, 64) | 0 | ['reduction[0][0]'] |
| bidirectional (Bidirectional) | (None, 32, 256) | 197632 | ['dropout[0][0]'] |
| bidirectional_1 (Bidirectional) | (None, 32, 128) | 164352 | ['bidirectional[0][0]'] |
| label (InputLayer) | [(None, None)] | 0 | [] |
| output (Dense) | (None, 32, 80) | 10320 | ['bidirectional_1[0][0]'] |
| CTC_loss (CTCLayer) | (None, 32, 80) | 0 | ['label[0][0]', 'output[0][0]'] |

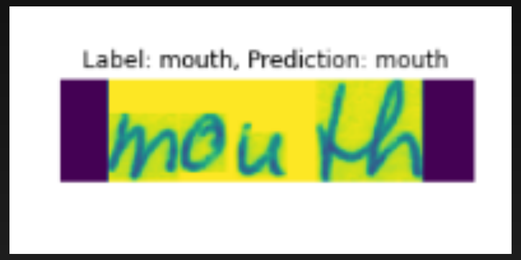
Total params: 423,952
Trainable params: 423,952
Non-trainable params: 0

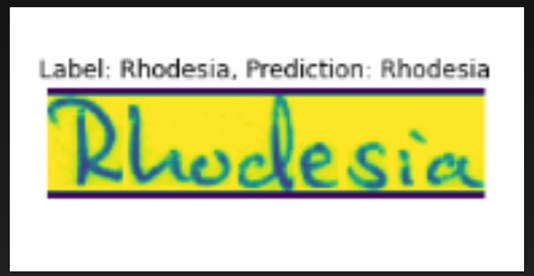
# Results

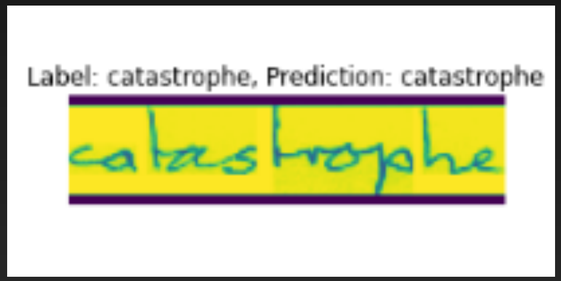
## Almost Perfect

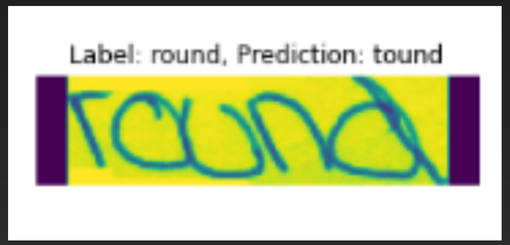


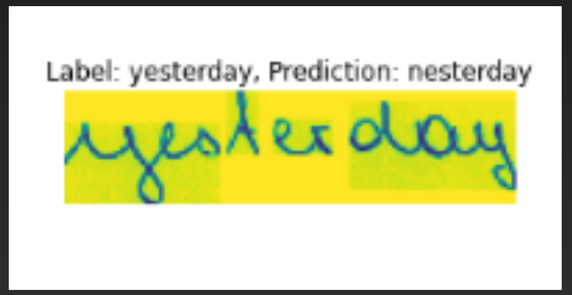


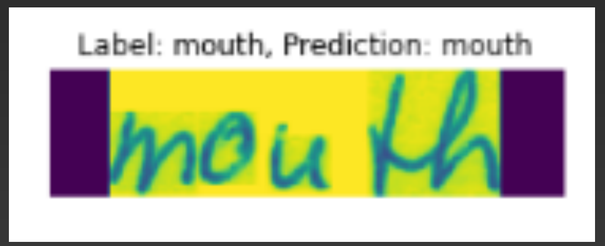



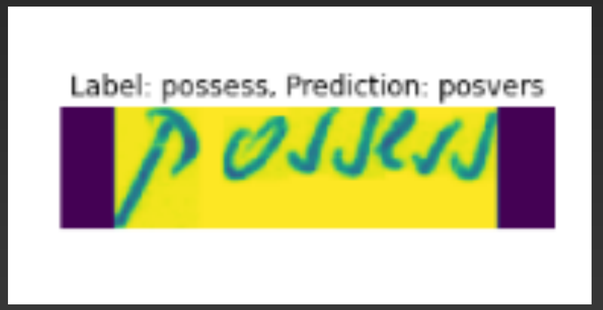



# Conclusions

- First approach failed
  - Linear segmentation is unprecise overall
  - Predicting attached handwriting is also hard
- Second approach was the right fit
  - Not single-character focused => looks at words as a whole
  - WORKS!

# Github Repository

- You can access all our work on Github: https://github.com/youssef-chaabouni/CSE204-Machine-Learning-Project

# References


- https://distill.pub/2017/ctc/
- http://vision.stanford.edu/teaching/cs231n/reports/2017/pdfs/810.pdf
- https://keras.io/examples/vision/handwriting_recognition/
- https://ai.googleblog.com/2019/03/rnn-based-handwriting-recognition-in.html