ELL409

COURSE PROJECT PRESENTATION
HANDWRITTEN OCR

PRANAV KUMAR
2022EE31191

MOHIT DUA
2022CS51745

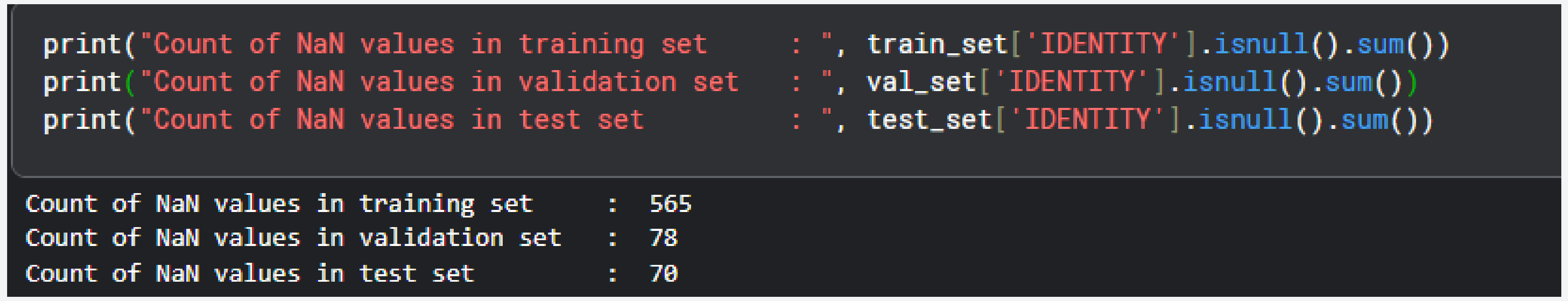
TEAM WHEYPROTEIN

PROJECT 9

PROBLEM STATEMENT

Develop a model that can accurately recognize and output handwritten names from a large dataset

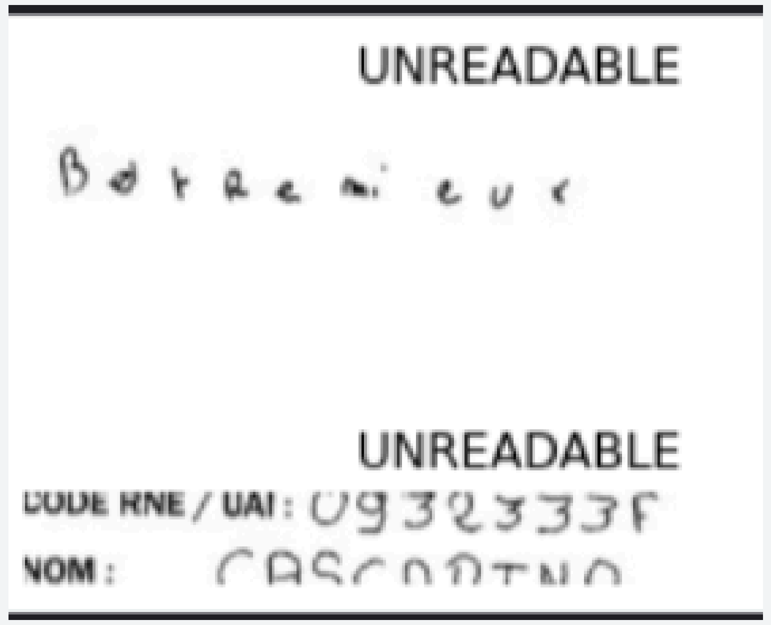
Exploratory Data Analysis

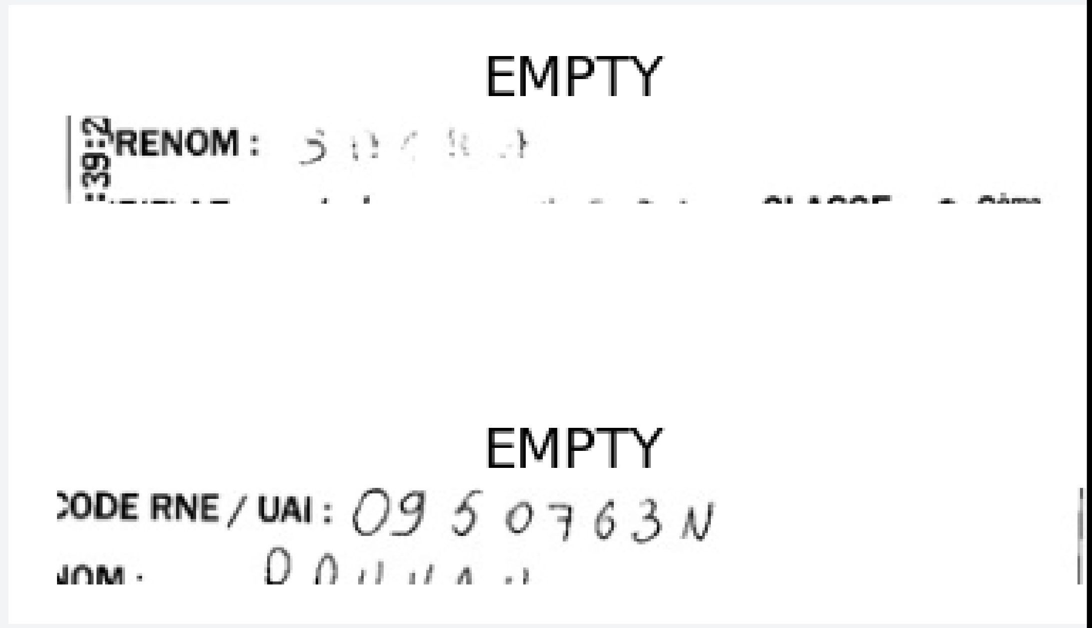


```
print("Count of NaN values in training set     : ", train_set['IDENTITY'].isnull().sum())
print("Count of NaN values in validation set   : ", val_set['IDENTITY'].isnull().sum())
print("Count of NaN values in test set         : ", test_set['IDENTITY'].isnull().sum())
```

```
Count of NaN values in training set     :  565
Count of NaN values in validation set   :  78
Count of NaN values in test set         :  70
```

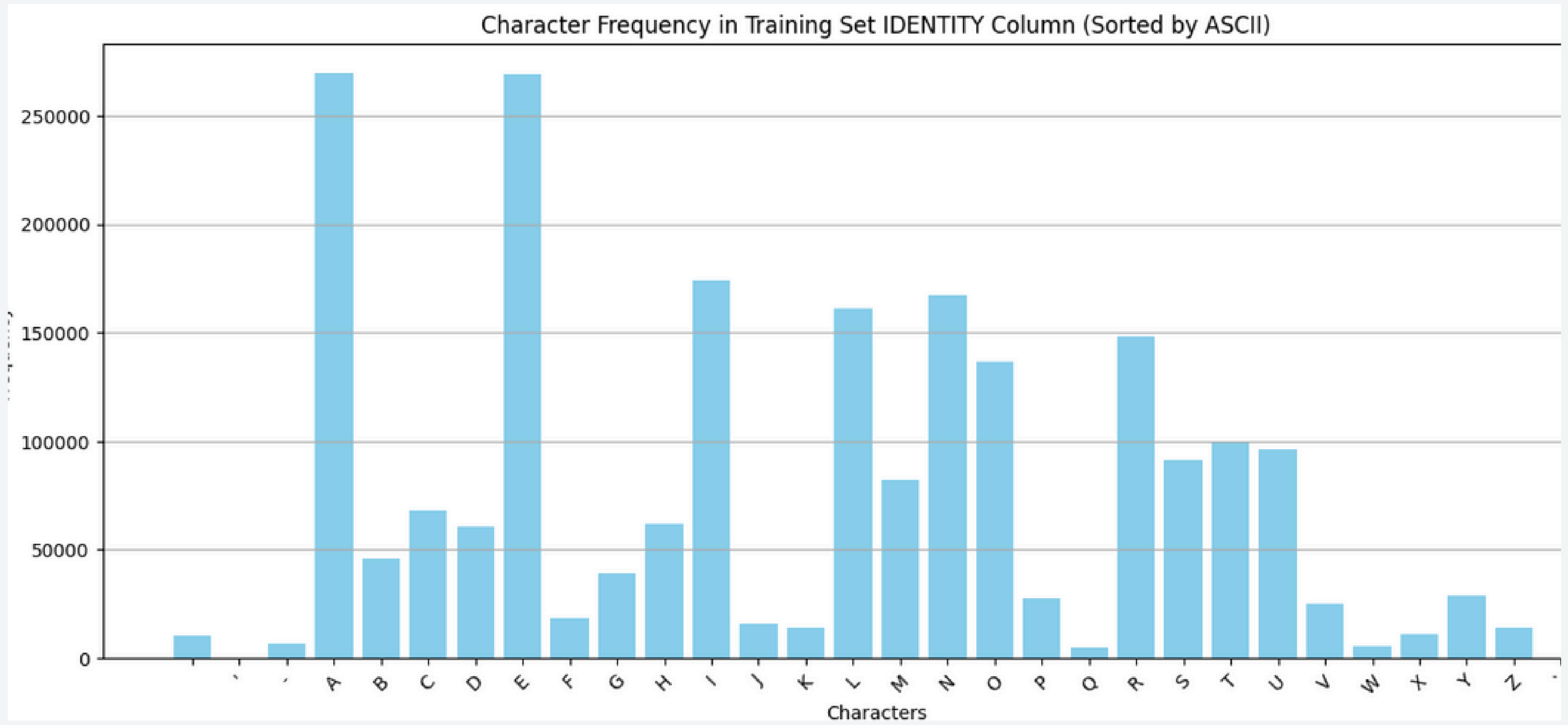
Existence of NULL values



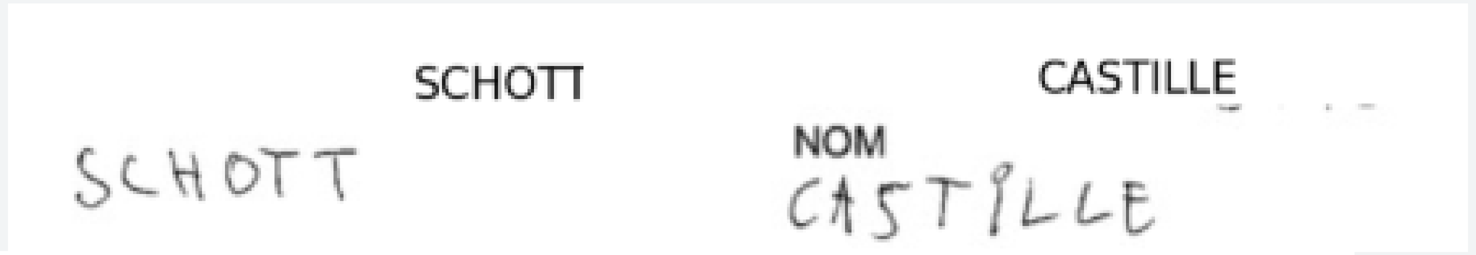


Images marked as unreadable and empty

Reccurring text units not relevant to the problem



Frequency distribution of Characters



Most names are in block letters, barely any cursive

Approach 1

Algorithm-based character extraction + CNN classifier

A common and simple image classification project is classification on the (E)MNIST dataset

Since we are not allowed to use external datasets, if we are able to create a good enough character segmenter, we can create our own version of MNIST.

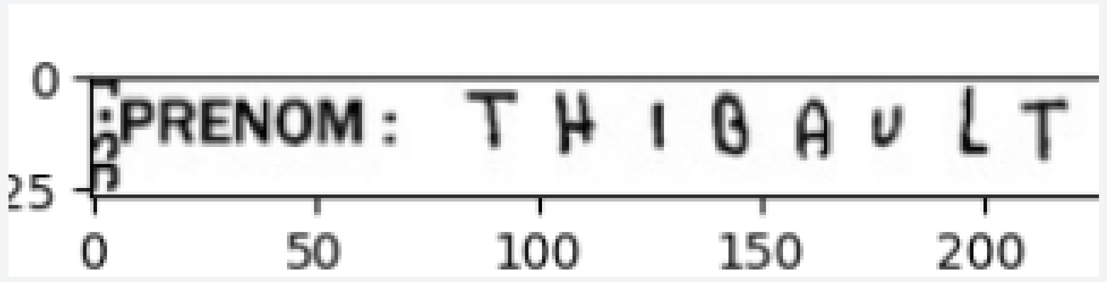
Then we just have to train a 26-class image classifier (CNN) and sequentially predict each character in the given image.

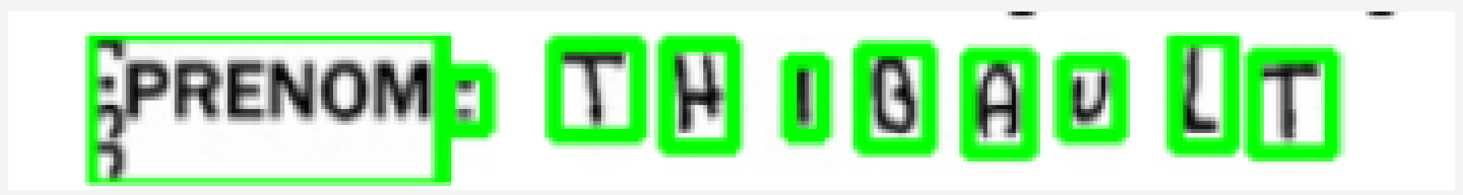
Character Segmenter

We used contour detection available through the imutils library

After processing all the contours, they are sorted left-to-right.

We stored the letters after resizing to 28x28 pixels to train our classifier





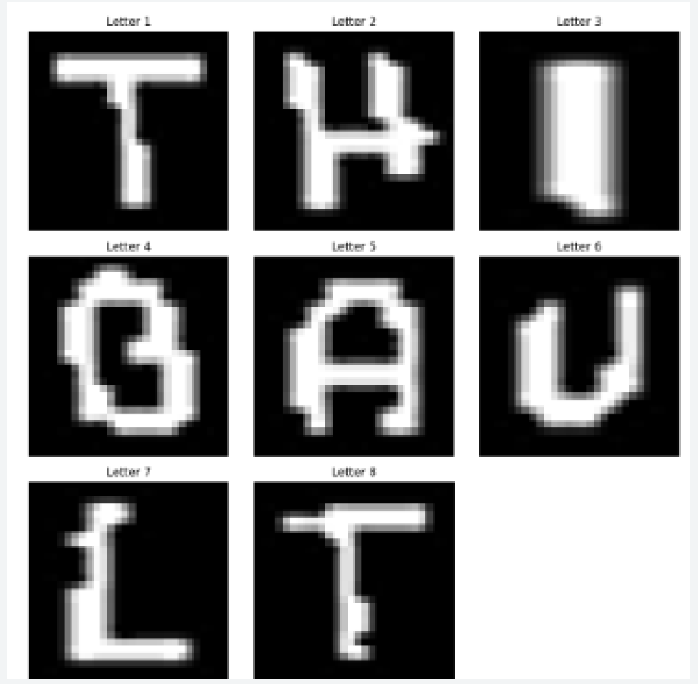
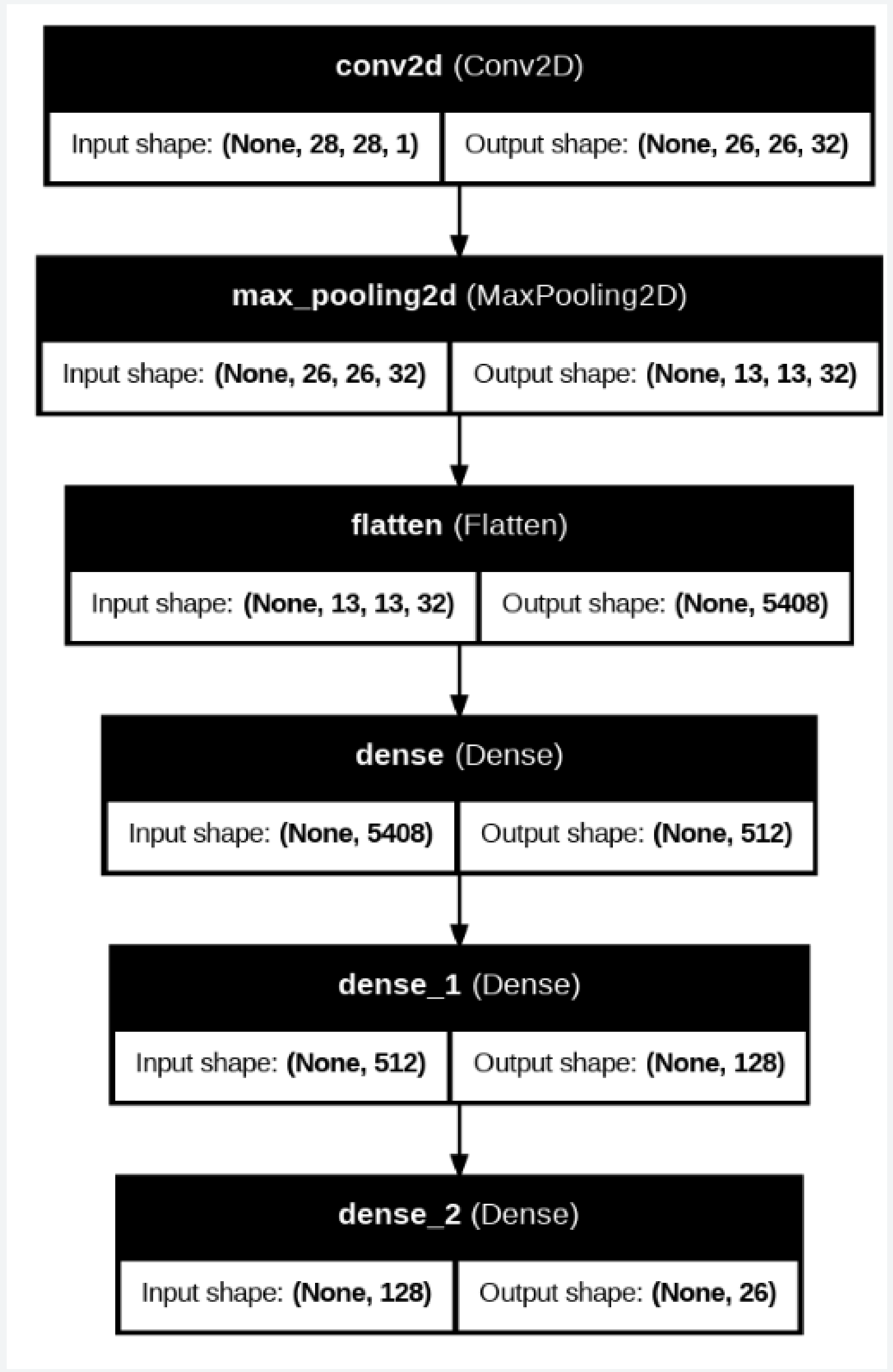


Image Classifier

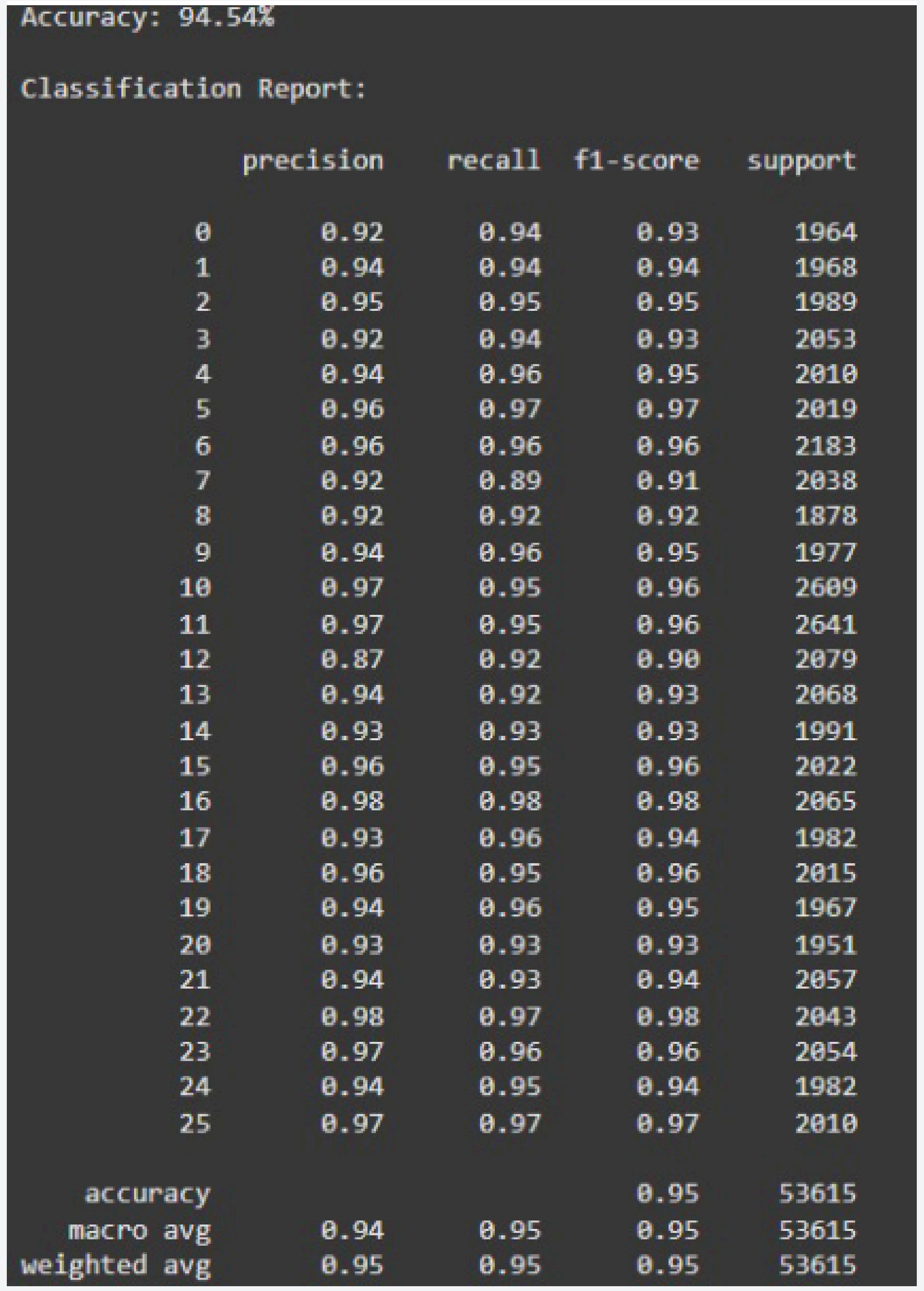
A shallow CNN with one convolutional + max-pooling layer connected to a feed-forward network with a 26-class softmax output.

The imbalanced data due to different character frequencies was balanced using duplication.

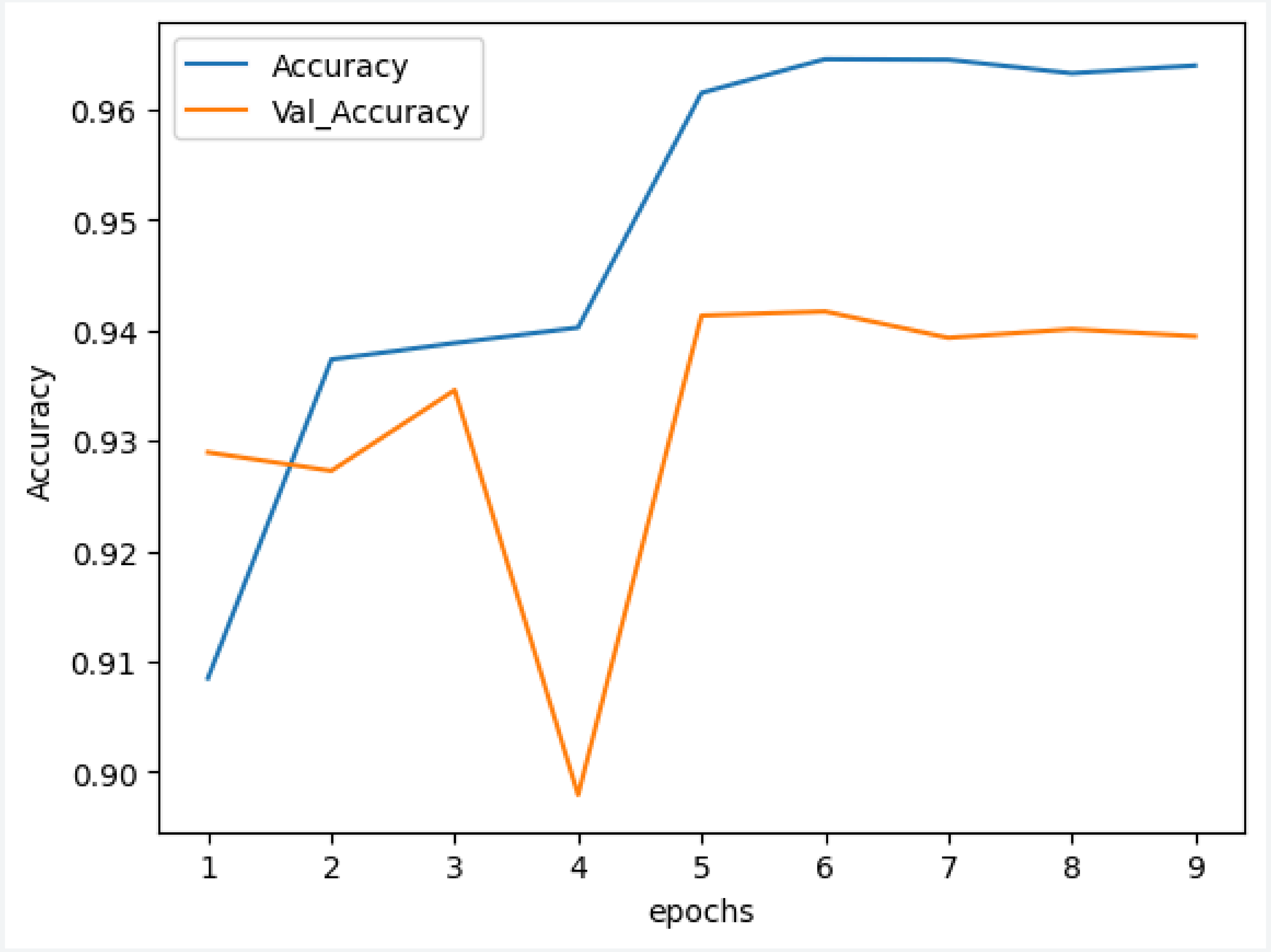
Trained on ~10k images for each letter.


conv2d (Conv2D)
Input shape: **(None, 28, 28, 1)**
Output shape: **(None, 26, 26, 32)**
max_pooling2d (MaxPooling2D)
Input shape: **(None, 26, 26, 32)**
Output shape: **(None, 13, 13, 32)**
flatten (Flatten)
Input shape: **(None, 13, 13, 32)**
Output shape: **(None, 5408)**
dense (Dense)
Input shape: **(None, 5408)**
Output shape: **(None, 512)**
dense_1 (Dense)
Input shape: **(None, 512)**
Output shape: **(None, 128)**
dense_2 (Dense)
Input shape: **(None, 128)**
Output shape: **(None, 26)**

Results - CNN Classifier



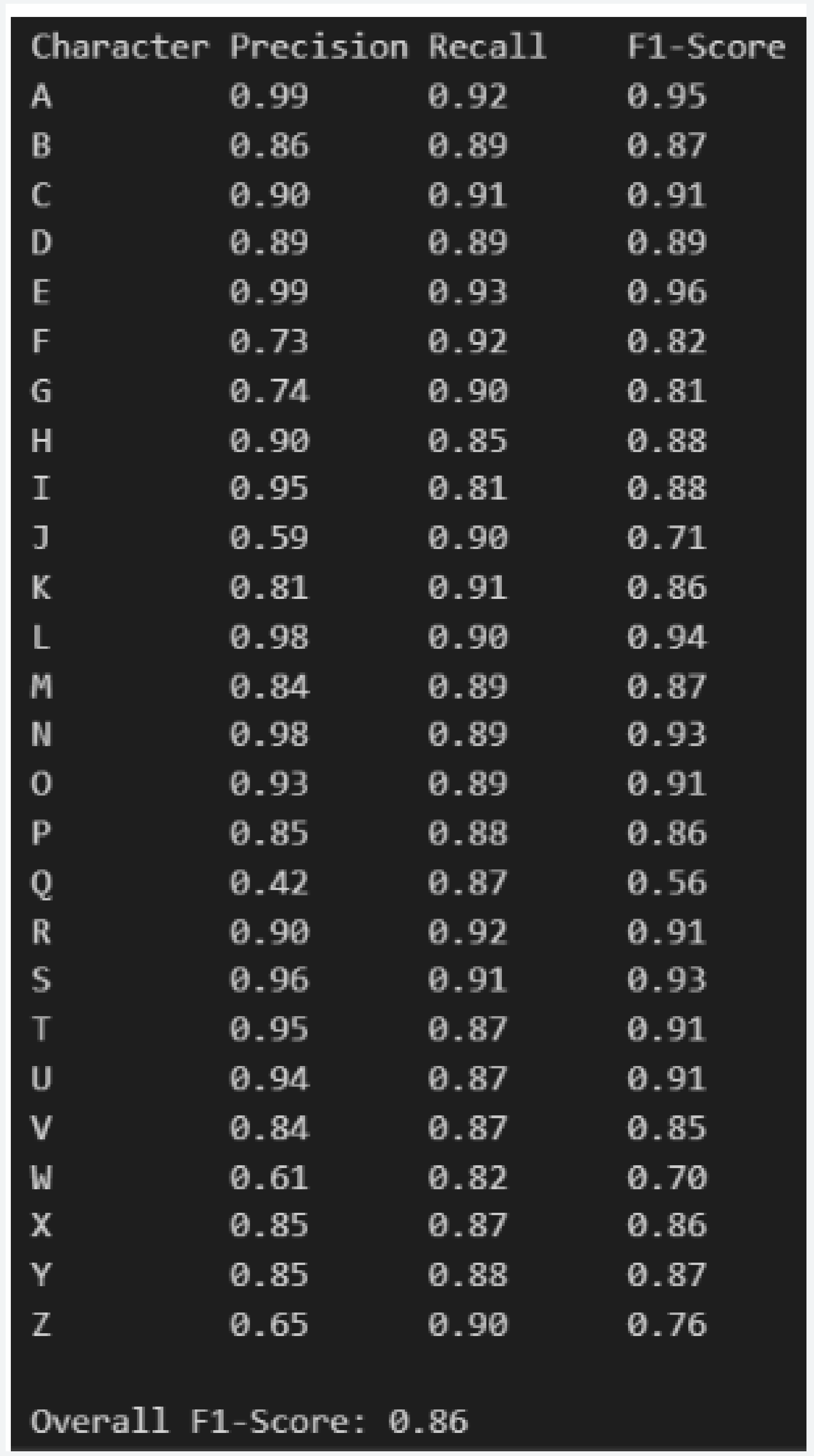
```
Accuracy: 94.54%

Classification Report:

              precision    recall  f1-score   support

           0       0.92      0.94      0.93      1964
           1       0.94      0.94      0.94      1968
           2       0.95      0.95      0.95      1989
           3       0.92      0.94      0.93      2053
           4       0.94      0.96      0.95      2010
           5       0.96      0.97      0.97      2019
           6       0.96      0.96      0.96      2183
           7       0.92      0.89      0.91      2038
           8       0.92      0.92      0.92      1878
           9       0.94      0.96      0.95      1977
          10       0.97      0.95      0.96      2609
          11       0.97      0.95      0.96      2641
          12       0.87      0.92      0.90      2079
          13       0.94      0.92      0.93      2068
          14       0.93      0.93      0.93      1991
          15       0.96      0.95      0.96      2022
          16       0.98      0.98      0.98      2065
          17       0.93      0.96      0.94      1982
          18       0.96      0.95      0.96      2015
          19       0.94      0.96      0.95      1967
          20       0.93      0.93      0.93      1951
          21       0.94      0.93      0.94      2057
          22       0.98      0.97      0.98      2043
          23       0.97      0.96      0.96      2054
          24       0.94      0.95      0.94      1982
          25       0.97      0.97      0.97      2010

    accuracy                           0.95     53615
   macro avg       0.94      0.95      0.95     53615
weighted avg       0.95      0.95      0.95     53615
```



Results - On the dataset

Percentage of words predicted correctly = <u>54.33%</u>

Percentage of characters predicted correctly =<u>74.66%</u>



```
Character Precision Recall    F1-Score
A         0.99      0.92      0.95
B         0.86      0.89      0.87
C         0.90      0.91      0.91
D         0.89      0.89      0.89
E         0.99      0.93      0.96
F         0.73      0.92      0.82
G         0.74      0.90      0.81
H         0.90      0.85      0.88
I         0.95      0.81      0.88
J         0.59      0.90      0.71
K         0.81      0.91      0.86
L         0.98      0.90      0.94
M         0.84      0.89      0.87
N         0.98      0.89      0.93
O         0.93      0.89      0.91
P         0.85      0.88      0.86
Q         0.42      0.87      0.56
R         0.90      0.92      0.91
S         0.96      0.91      0.93
T         0.95      0.87      0.91
U         0.94      0.87      0.91
V         0.84      0.87      0.85
W         0.61      0.82      0.70
X         0.85      0.87      0.86
Y         0.85      0.88      0.87
Z         0.65      0.90      0.76

Overall F1-Score: 0.86
```

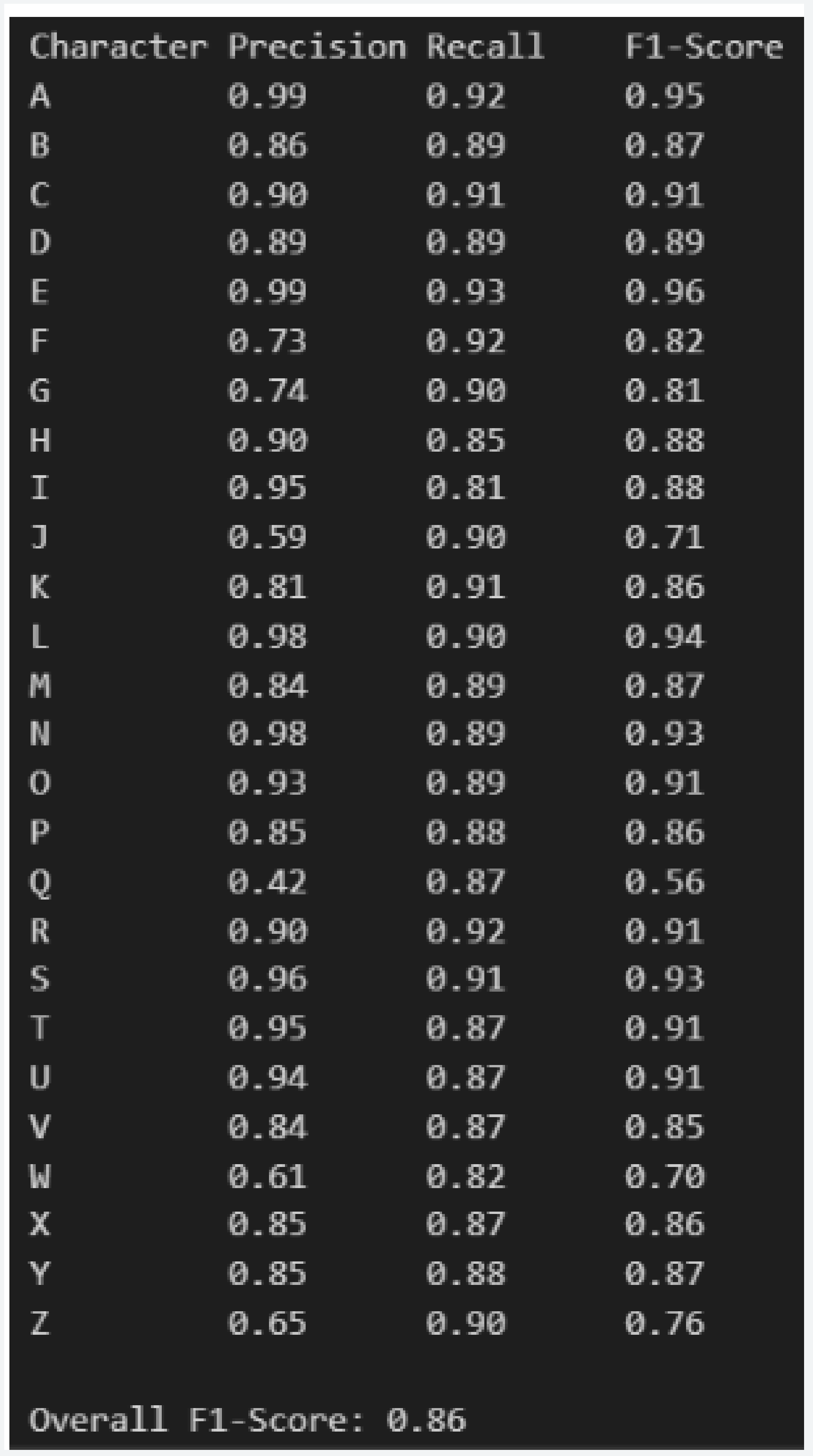
Discussion

The model predicts characters correctly most of the time.

The accuracy drop is because of a sub-optimal character extractor.

Low F1-scores on infrequent characters (Q, J, Z etc)

Overall F1-Score: 0.86



```
Character Precision Recall    F1-Score
A         0.99      0.92      0.95
B         0.86      0.89      0.87
C         0.90      0.91      0.91
D         0.89      0.89      0.89
E         0.99      0.93      0.96
F         0.73      0.92      0.82
G         0.74      0.90      0.81
H         0.90      0.85      0.88
I         0.95      0.81      0.88
J         0.59      0.90      0.71
K         0.81      0.91      0.86
L         0.98      0.90      0.94
M         0.84      0.89      0.87
N         0.98      0.89      0.93
O         0.93      0.89      0.91
P         0.85      0.88      0.86
Q         0.42      0.87      0.56
R         0.90      0.92      0.91
S         0.96      0.91      0.93
T         0.95      0.87      0.91
U         0.94      0.87      0.91
V         0.84      0.87      0.85
W         0.61      0.82      0.70
X         0.85      0.87      0.86
Y         0.85      0.88      0.87
Z         0.65      0.90      0.76

Overall F1-Score: 0.86
```

Approach 2 - CNN + RNN = CRNN

The rule-based feature extractor is causing a bottleneck, leading to a large drop in accuracy.

We will now improve upon the feature extractor, making it data-driven instead of algorithmic.

For this, we will use a CNN to produce a feature map, and an RNN for sequence prediction.

Connectionist Temporal Classification (CTC) wiil be used for predicting sequences of variable lengths.

Connectionist Temporal Classication (CTC)

- CTC is a loss function used for sequence-to-sequence tasks, enabling models to make predictions on sequences of varying lengths without needing alignment between input and output sequences.

- Working Principle: CTC decodes the output of a neural network by considering all possible alignments between the input sequence and the target sequence. The model outputs a probability distribution for each time step, and CTC training optimizes the alignment of these distributions with the target sequence.

- Applications: CTC is commonly used in speech recognition (e.g., transcribing audio to text), handwriting recognition, and other tasks where input-output sequence alignment is not predefined or needs to be learned from the data. It allows for end-to-end training without manual segmentation of input data.

Architecture

Feature Extractor

3 Conv2D+MaxPooling Layers with BatchNorm and Dropout; output is flattened at the end and fed into the RNN as a fixed-length sequence of **64** timesteps.

Sequence-Predictor

Two Bidirectional LSTM layers, a feed-forward network followed by softmax at the output of every hidden state to predict the character.

Results

Character Accuracy

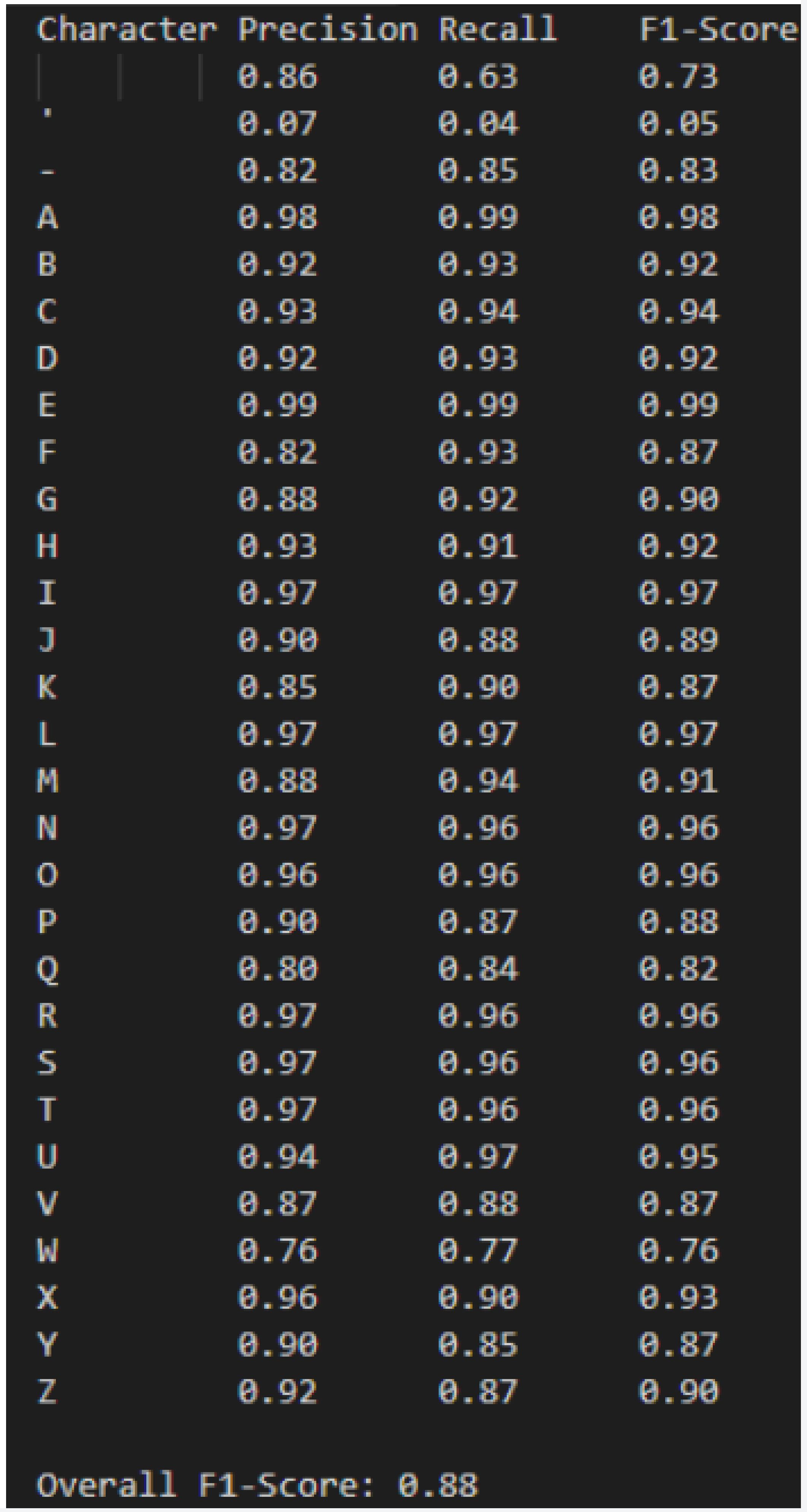
87.91%

Word Accuracy

73.60%

Overall Character-wise F1-score

0.88



```
Character Precision Recall    F1-Score
          0.86      0.63      0.73
'         0.07      0.04      0.05
-         0.82      0.85      0.83
A         0.98      0.99      0.98
B         0.92      0.93      0.92
C         0.93      0.94      0.94
D         0.92      0.93      0.92
E         0.99      0.99      0.99
F         0.82      0.93      0.87
G         0.88      0.92      0.90
H         0.93      0.91      0.92
I         0.97      0.97      0.97
J         0.90      0.88      0.89
K         0.85      0.90      0.87
L         0.97      0.97      0.97
M         0.88      0.94      0.91
N         0.97      0.96      0.96
O         0.96      0.96      0.96
P         0.90      0.87      0.88
Q         0.80      0.84      0.82
R         0.97      0.96      0.96
S         0.97      0.96      0.96
T         0.97      0.96      0.96
U         0.94      0.97      0.95
V         0.87      0.88      0.87
W         0.76      0.77      0.76
X         0.96      0.90      0.93
Y         0.90      0.85      0.87
Z         0.92      0.87      0.90

Overall F1-Score: 0.88
```

Discussion

The CRNN model outperforms the CNN model by a significant margin in word and character prediction accuracy.

This model was able to overcome the bottleneck caused by the feature extractor.

This illustrates the effectiveness of data-driven methods of feature engineering.

Challenges Faced

Memory and computation time issues, especially when generating the letter dataset and training the neural networks

Use of parallel processing, online GPUs (Kaggle, Colab)

Challenges Faced

Finetuning the character-segmenter

<u>It would often include irrelevant text boxes</u>

-> Use of area and width hyperparameters to filter contours.

<u>It would segment multiple characters as one-block or one character as multiple</u>

-> Hierarchical contouring, division at multiple levels; merging of boxes close to each other.

Outcomes

Successfully built multiple ML models for Optical Character Recognition

Familiarised with CNN and RNN models.

Explored a new sequence modelling technique.

Thank You!