ELL409 Course Project

Handwritten Name OCR using CNNs and RNNs

Pranav Kumar (2022EE31191)[1] and Mohit Dua (2022CS51745)[2]

[1]Department of Electrical Engineering, IIT Delhi
[2]Department of Computer Science & Engineering, IIT Delhi

November 24, 2024

Abstract

This assignment aims to develop and evaluate a machine learning model for recognizing handwritten text using a Kaggle dataset. The project involves exploratory data analysis (EDA) to visualize character distributions, data preprocessing for normalization and augmentation, and selecting an appropriate model architecture, potentially combining Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). The model will be trained and optimized, focusing on performance metrics such as character-wise F1 scores. Evaluation will include confusion matrix analysis and discussions on misclassifications. Deliverables will consist of a Jupyter notebook or Python script and a report summarizing the methodology and findings.

1 Introduction

Optical Character Recognition (OCR) converts images of text into machine-readable formats, enabling document digitization. **Convolutional Neural Networks (CNNs)** extract image features like edges and textures via convolution, mathematically expressed as:

$$(I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) \cdot K(m,n)$$

Here, I is the input image, K the kernel, and $(I * K)(i,j)$ the output at position (i,j). Extracted features are fed into **Recurrent Neural Networks (RNNs)**, which handle sequential data and maintain memory of previous inputs. The RNN operation is:

$$h_t = f(W_h h_{t-1} + W_x x_t)$$

where h_t is the hidden state at time t, W_h and W_x are weight matrices, and f is a nonlinear activation function. For sequence prediction, **Connectionist Temporal Classification (CTC)** loss enables alignment-free training, defined as:

$$L = -\log(\sum_{a \in A} P(a|x))$$

where A is the set of all possible output sequences, and $P(a|x)$ is the probability of sequence a given input x. This approach combines CNNs for feature extraction and RNNs for sequence modeling, making it effective for recognizing handwritten names.

2 Dataset

The dataset provided for this project consists of a large collection of handwritten names, comprising about 330,000 training samples, 40,000 validation samples, and 40,000 test samples. This extensive dataset is designed to facilitate the development and evaluation of machine learning models for handwritten text recognition. Each entry includes images of handwritten names along with their corresponding labels, allowing models to learn from a diverse range of handwriting styles. The variety in writing presents both challenges and opportunities for improving OCR technologies. Below is an image showcasing some examples from the dataset:

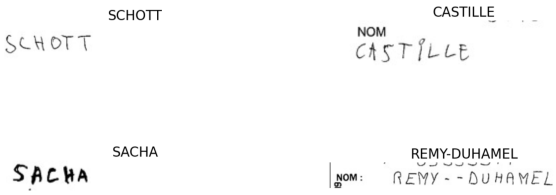


Figure 1: Training examples from the dataset

3 Exploratory Data Analysis & Data Preprocessing

During the exploratory data analysis phase, images that were not labeled with an identity were removed from the dataset. Additionally, images labeled as 'UNREADABLE' or 'EMPTY' were also excluded to ensure the quality of the training data. It was observed that certain pieces of text, which are irrelevant to the final prediction, appeared frequently throughout the dataset. This insight was instrumental in determining the hyperparameters for the width and height of selected contours in the rule-based feature extractor model. We also see that there are exactly 29 distinct characters - the 26 letters of the alphabet, and 3 special characters.

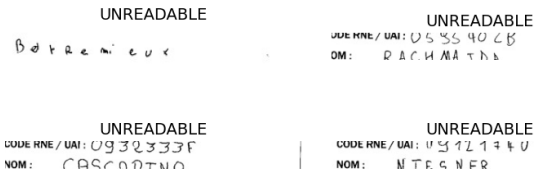


Figure 2: Images labelled 'UNREADABLE'

Figure 3: Irrelevant section

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

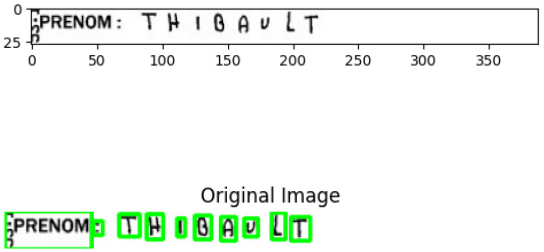
Figure 4: Number of NaNs dropped

4 Rule Based Character Extraction + CNN

The first model implemented utilizes contour detection to extract individual characters from the image. These extracted characters are then used to create a dataset of 28x28 pixel images, which serves as the input for training a character classification Convolutional Neural Network (CNN). Once trained, the model predicts names by sequentially identifying and classifying each character in the input image.

The letter extraction algorithm processes an image of handwritten text through the following steps:

1. **Preprocessing**: The image is cropped and converted to grayscale.
2. **Thresholding**: A binary inverse threshold is applied to highlight the text.
3. **Dilation**: The binary image is dilated to connect nearby characters.
4. **Contour Detection**: Contours are identified and sorted from left to right.
5. **Bounding Box Calculation**: Rectangles are drawn around contours that meet size criteria.
6. **Region of Interest (ROI)**: Each letter's ROI is extracted and thresholded again.
7. **Letter Extraction**: Detected letters are resized to 28x28 and normalized.



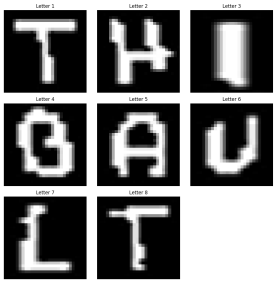


Figure 5: Feature extraction using contour detection

These images are utilized to train a Convolutional Neural Network (CNN) that consists of one convolutional layer followed by a max-pooling layer, a three-layer-deep MLP followed by a softmax layer for the classification of the 26 letters of the English alphabet. Duplicative

upsampling was used to balance the dataset. All activation functions are ReLU.

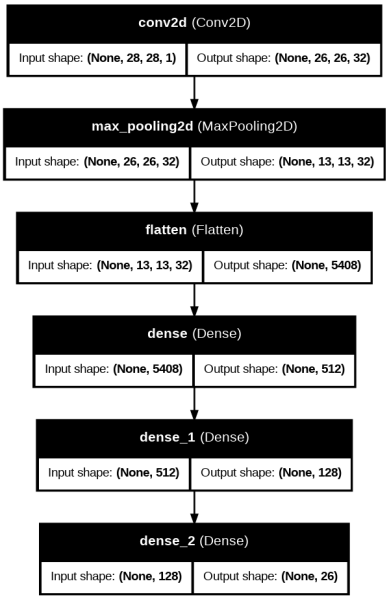


Figure 6: Architecture of the CNN used

Results & Discussion

Total correct character predictions - **74.66%**

We see that the model performed well at classifying the characters, with an average F1 score of 0.86. The F1 score is lower for characters that occur at lower frequencies in the dataset (Q, J etc.) which indicates the CNN may have overfit to the higher frequency characters.

Total correct word predictions - **54.33%**

The model does not perform as well at recognizing the entire word. This is due to it being heavily reliant on the character extraction algorithm, which serves as a bottleneck and significantly reduces overall accuracy.

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

Figure 7: Character-wise training results for the CNN

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

Figure 8: Results on test data

5 CRNN with CTC Loss

The second model overcomes the bottleneck of the first model's feature extractor by using a CNN to automatically identify essential features in the input image. The CNN output is flattened and fed into a network with two bi-LSTM layers. Each hidden state is processed through a feed-forward network, producing a 30-class softmax output layer that assigns characters at each time step. Notably, there are 29 distinct characters in our vocabulary, with the 30th class serving as a special 'blank' character used by Connectionist Temporal Classification (CTC) loss to align timesteps and generate meaningful outputs. CTC loss is tailored for sequence prediction tasks where input-output alignment is unknown. It enables models to produce variable-length outputs from fixed-length inputs by incorporating a 'blank' label that allows skipping timesteps. During training, CTC calculates the loss by summing the probabilities of all valid alignments to the correct output sequence. This flexibility makes CTC particularly effective for applications like handwriting and speech recognition, where character lengths and input-output relationships can vary.

Results & Discussion

Total correct character predictions - **87.91%** This is a significant improvement over the previous model, indicating the robustness of the model.

Total correct word predictions - **73.60%** The CNN feature extractor coupled with the LSTM sequence predictor predicts 20% more of the test set completely.

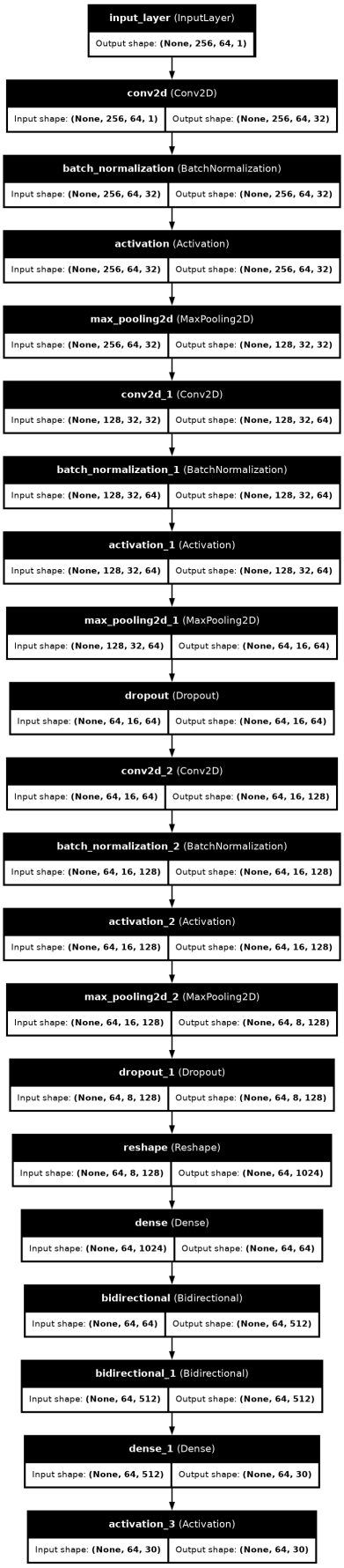


Figure 9: CRNN architecture

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

Figure 10: CRNN character-wise results

6 Conclusion

This project successfully explored two distinct deep learning architectures for Optical Character Recognition (OCR) on handwritten names, demonstrating the potential and limitations of each approach.

The first method, combining rule-based contour detection and CNNs, emphasized the importance of robust preprocessing. The CNN excelled in character-wise classification with an F1 score of 0.86 on average but faced challenges in word-level accuracy due to the limitations of the contour-based feature extraction algorithm. The reliance on preprocessing introduced a bottleneck, highlighting the critical role of data quality and preprocessing algorithms in OCR tasks.

The second method, employing a Convolutional Recurrent Neural Network (CRNN) with CTC loss, eliminated the need for explicit character segmentation. By combining the feature extraction capabilities of CNNs and the sequence modeling strength of bi-LSTMs, this model demonstrated a significant improvement in both character-level (87.91%) and word-level (73.60%) accuracy. The CTC loss enabled the model to handle variable-length sequences effectively, proving its suitability for real-world applications where character alignments are not predefined.