ECE 569A Artificial Intelligence

Course Project: Handwritten Character Recognition

Sannath Reddy (V00949217) & Nishra Shah (V00936027)

Table of Contents

1. Introduction

Handwriting recognition (HWR), or Handwritten text recognition (HTR), allows the computer to receive and interpret handwritten input from scanned paper documents or images.

In this Artificial Intelligence course project, we have implemented a search functionality for Handwritten English notes using EMNIST dataset. We are considering 62 classes : 10 digits + 26 (lower case) + 26 (upper case) to train our neural network model. Each class is trained with 5000 images. To implement and achieve our results, we have used concepts of Convolution Neural Network, Kernel Filters, Bounding boxes, Image segmentation - line segmentation , word segmentation , letter/character segmentation.

Background and Motivation

In current times when online learning is popular, we often want to digitize the scanned documents and struggle with doing so as scanned documents don't have various functions like search functionality. In this project, we are trying to address this problem by adding more features to the scanned document. Various neural network and computer vision concepts and libraries are used to add this functionality to handwritten scanned documents.

2. Related Work

We found this paper **Handwritten English Character Segmentation by Baseline Pixel Burst Method (BPBM)** by R. K. Mandal and N R Manna. We have used the concepts discussed in this paper for image segmentation. In the first step, the image is segmented in paragraphs, next is into lines, then words and finally characters. Later on, each character is used as input to the trained neural network.

3. Problem Formulation

Case:1 - Considering single stroke handwritten notes can be challenging as characters are inconsistent. Sometimes, they are irregular/skewed/slant/joint/ which makes handwritten text recognition a very complicated task. In order to segment each character out of the image, we first find the paragraph, then segment the lines, then words and lastly characters.

Line segmentation Algorithm

Segmentation of lines from a paragraph algorithm basically converts the scanned paragraph in bmp format, which is then converted into a two dimensional binary matrix. This conversion helps in locating the lines from the paragraph.

Below are the steps we followed for the paper:

Step 1: Convert the image into a ndarray.

Step 2: Next step is to locate the topmost and bottommost rows of the image in order to make a bounding box around the text of the image.

This is done by finding the first black pixel from the top for topmost row and similarly first black pixel from bottom of the image.

Step 3: Locate the extreme left and extreme right columns of the image to separate the lines.

Word segmentation Algorithm

Segmentation of the words from the lines of the paragraph includes considering the lines from the above algorithm.

Below are the steps we followed from the paper:

Step 1: Reading the lines of the paragraph stored in different matrices.

Step 2: Locating the extreme left column of the line of the paragraph and drift the column

Step 3: Next is to test the successive columns for the presence of black pixels.

Character segmentation Algorithm

Below are the steps we followed for the paper:

Step 1: From the above step, read the words of the lines of the paragraph stored in different matrices.

Step 2: Locate the extreme left column of the word of the line of the paragraph and drift the column towards the right to find the first blank column having no black pixel.

The above segmentation forms a major part of the projection. After segmentation, each character is converted to an image, which is fed into the trained Neural Network.

Case: 2 - We could have used NN with the below architecture to obtain our results.

Conv2D --> Maxpool --> Conv2D --> Maxpool --> Fully Connected (FC) --> Softmax --> Classification.

Case: 3 - We could have used the baseline concept which straightens and removes the skew from every line. After which we just need to segregate the alphabets and recognise them.

We have implemented Case:1 and reported the results and procedure in the following sections.

4. Methodology

This project is divided into three parts- First part is neural network - training neural network with all 62 classes using cnn, relu activation. 5 layers, three epochs
Saved the model

In computer vision concepts, open cv, Take actua; input. Break into para, lines,word ,letters using bounding boxes, image segmentation. Single characters and compare each character to search keyword sequentially to find the keyword in the input document.

Part3- combining 1 2 and implementing search functioning. Which uses these parts and gives the location of the search keyword in the input document.

Model Improvement

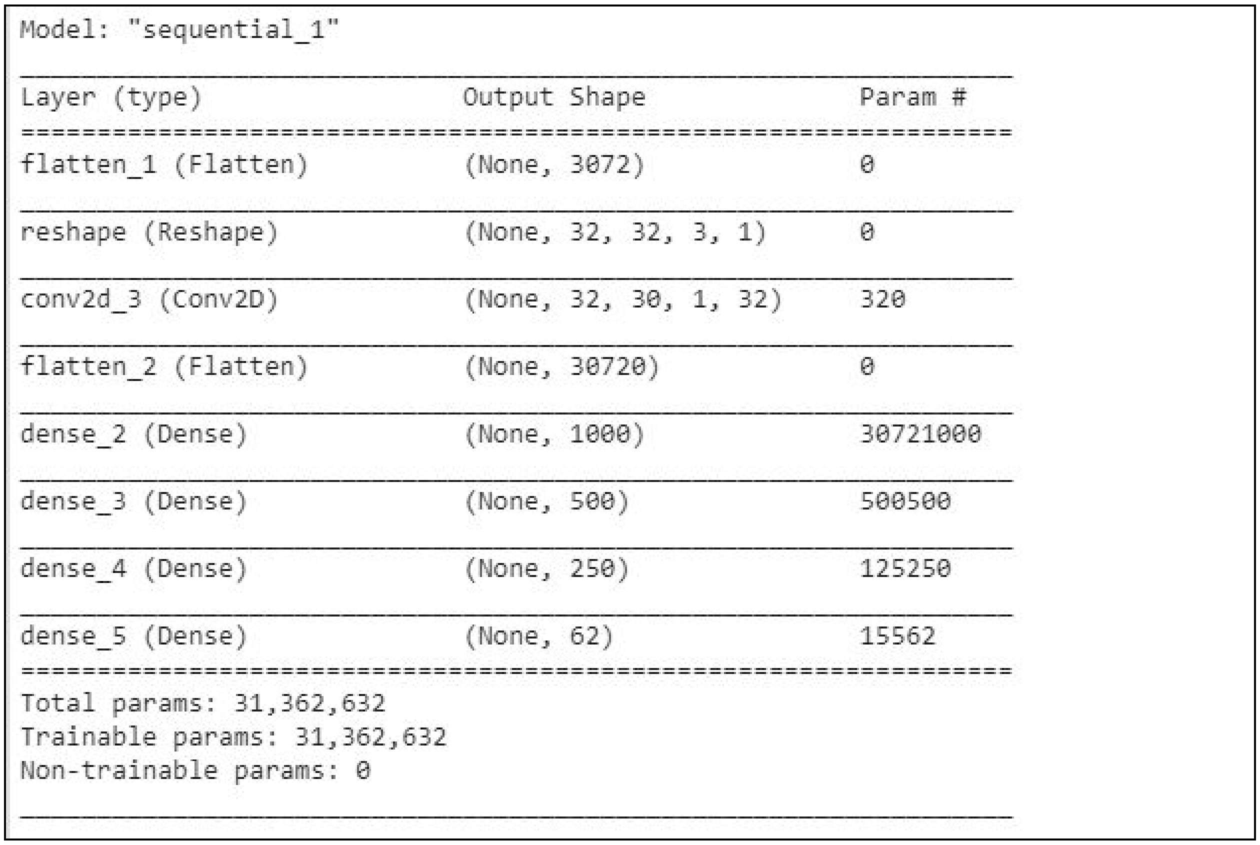
Our First model had 4 dense layers along with a 2D convolution layer featured with a (3,3) filter to detect the curved and straight lines of handwritten characters. Below is the brief description about the architecture of the model:

Input --> Flatten --> Reshape --> 2D Convolution layer --> Flatten --> Dense-1 --> Dense-2 --> Dense-3 --> Dense-4

The input is first fed into the Flatten layer which converts the input of (32,32,3) into a single column of size 3072. The column vector is reshaped to ndarray of (32,32,3,1) in the 2 layer of the architecture which is fed into 2D Convolution network with 32 filter size and (3,3) of kernel filter which is passed to flatten to 30720 column vector. Then we have 1st of dense layers comprises of 1000 neurons with activation function as Rectifier

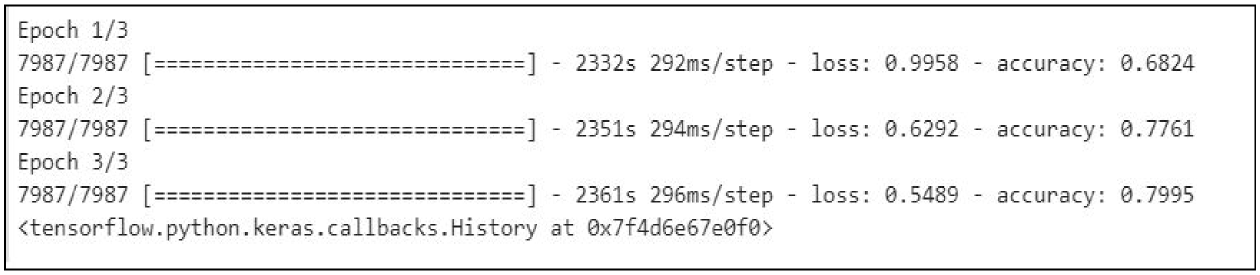
Linear Unit(relu), 2nd of dense layers comprises of 500 neurons with activation function as Rectifier Linear Unit(relu), 3rd of dense layers comprises of 250 neurons with activation function as Rectifier Linear Unit(relu) and 4th of dense layers comprises 64 neurons where each neuron represents each class (10 numbers + uppercase + lowercase).

Below screenshot represents the above described sequential neural network:



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 3072)              0
_________________________________________________________________
reshape (Reshape)            (None, 32, 32, 3, 1)      0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 32, 30, 1, 32)     320
_________________________________________________________________
flatten_2 (Flatten)          (None, 30720)             0
_________________________________________________________________
dense_2 (Dense)              (None, 1000)              30721000
_________________________________________________________________
dense_3 (Dense)              (None, 500)               500500
_________________________________________________________________
dense_4 (Dense)              (None, 250)               125250
_________________________________________________________________
dense_5 (Dense)              (None, 62)                15562
=================================================================
Total params: 31,362,632
Trainable params: 31,362,632
Non-trainable params: 0
_________________________________________________________________
```

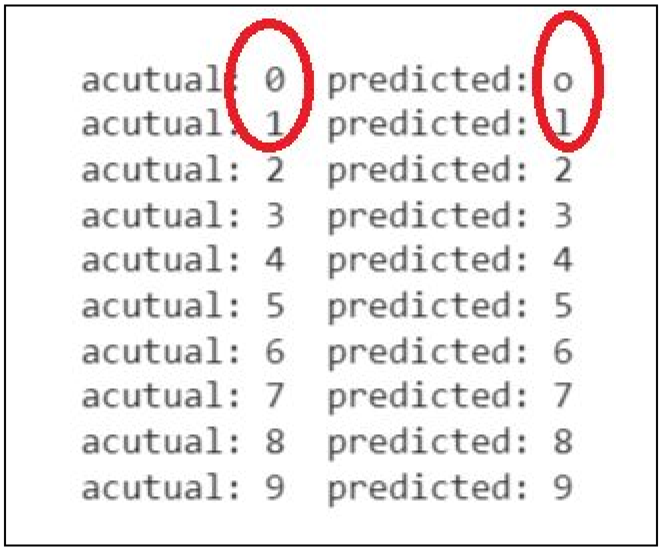
However, this model is not advisable to work on for the reason being multiple dense layers with more than required neurons in each layer. The model was trained with 5000 images for each class in 3 epochs and below are results of training the model.



```
Epoch 1/3
7987/7987 [==============================] - 2332s 292ms/step - loss: 0.9958 - accuracy: 0.6824
Epoch 2/3
7987/7987 [==============================] - 2351s 294ms/step - loss: 0.6292 - accuracy: 0.7761
Epoch 3/3
7987/7987 [==============================] - 2361s 296ms/step - loss: 0.5489 - accuracy: 0.7995
<tensorflow.python.keras.callbacks.History at 0x7f4d6e67e0f0>
```

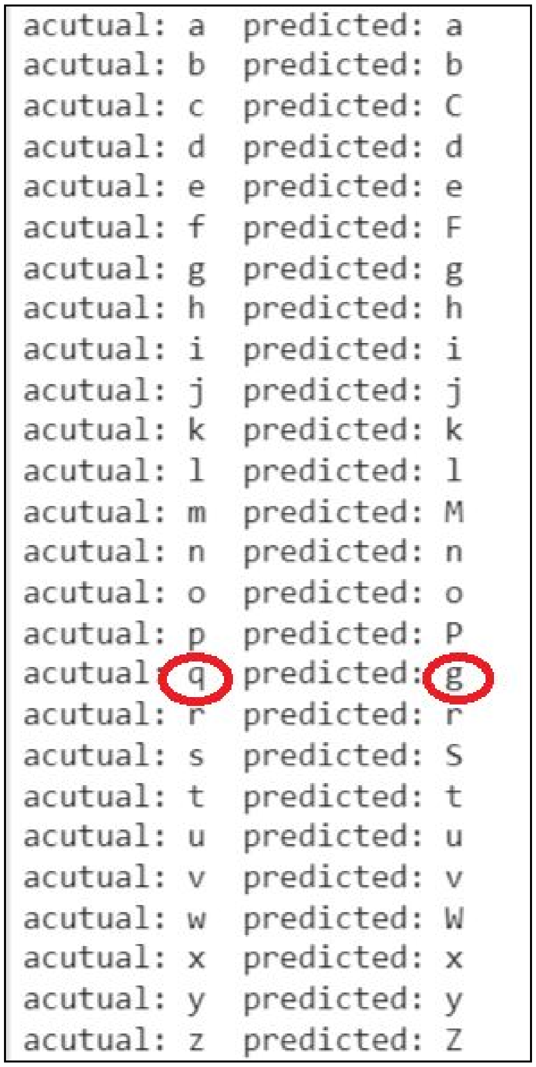
The model was trained very slowly because of the existence of dense layers with close to 2000 neurons. This creates complex computations when an image is passed through the network as an input eventually taking time for the entire training process. As we can see in the above screenshot, each epoch took a minimum of 40 minutes to get completed. The final accuracy after the 3rd epoch settled at 80% approx. with a loss of 0.5. Though the accuracy is good enough, this network clearly suffers an overfitting issue. Below is an example proving that the model is overfitted for the training set,

Neural Networks performance on samples from training set:

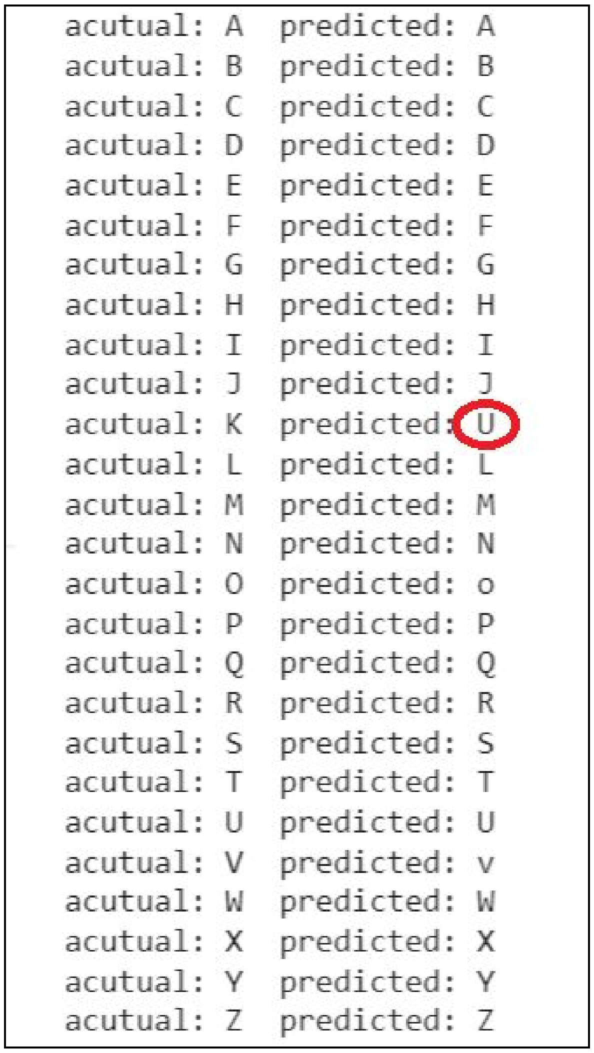


```
acutual: 0  predicted: o
acutual: 1  predicted: l
acutual: 2  predicted: 2
acutual: 3  predicted: 3
acutual: 4  predicted: 4
acutual: 5  predicted: 5
acutual: 6  predicted: 6
acutual: 7  predicted: 7
acutual: 8  predicted: 8
acutual: 9  predicted: 9
```

The above screenshot represents the results of testing the trained neural network on images from the training set where the circled characters are wrongly classified.



```
acutual: a  predicted: a
acutual: b  predicted: b
acutual: c  predicted: C
acutual: d  predicted: d
acutual: e  predicted: e
acutual: f  predicted: F
acutual: g  predicted: g
acutual: h  predicted: h
acutual: i  predicted: i
acutual: j  predicted: j
acutual: k  predicted: k
acutual: l  predicted: l
acutual: m  predicted: M
acutual: n  predicted: n
acutual: o  predicted: o
acutual: p  predicted: P
acutual: q  predicted: g
acutual: r  predicted: r
acutual: s  predicted: S
acutual: t  predicted: t
acutual: u  predicted: u
acutual: v  predicted: v
acutual: w  predicted: W
acutual: x  predicted: x
acutual: y  predicted: y
acutual: z  predicted: Z
```



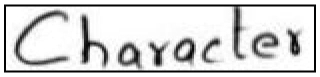
```
acutual: A  predicted: A
acutual: B  predicted: B
acutual: C  predicted: C
acutual: D  predicted: D
acutual: E  predicted: E
acutual: F  predicted: F
acutual: G  predicted: G
acutual: H  predicted: H
acutual: I  predicted: I
acutual: J  predicted: J
acutual: K  predicted: U
acutual: L  predicted: L
acutual: M  predicted: M
acutual: N  predicted: N
acutual: O  predicted: o
acutual: P  predicted: P
acutual: Q  predicted: Q
acutual: R  predicted: R
acutual: S  predicted: S
acutual: T  predicted: T
acutual: U  predicted: U
acutual: V  predicted: v
acutual: W  predicted: W
acutual: X  predicted: X
acutual: Y  predicted: Y
acutual: Z  predicted: Z
```

Though there are many other misclassifications like lower case ‘m’ being misclassified as Uppercase ‘M’, these are not considered as misclassifications or create an issue to implement the search functionality on handwritten notes. The ones which are circled in red are the major misclassifications which completely represent a different character.

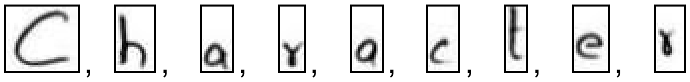
Neural network performance on unknown data:

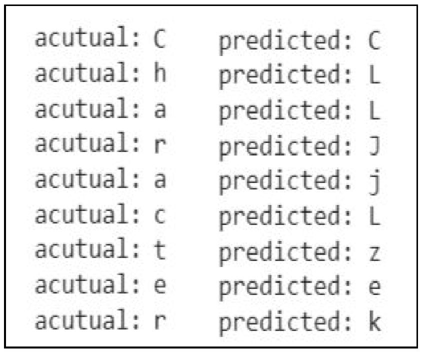
The characters (‘C’,’h’,’a’,’r’,’a’,’c’,’t’,’e’,’r’) are segmented from an input scanned document and passed through the neural network to test the accuracy on unknown samples.

Word from scanned document :



Characters segmented from the above word :





```
acutual: C    predicted: C
acutual: h    predicted: L
acutual: a    predicted: L
acutual: r    predicted: J
acutual: a    predicted: j
acutual: c    predicted: L
acutual: t    predicted: z
acutual: e    predicted: e
acutual: r    predicted: k
```

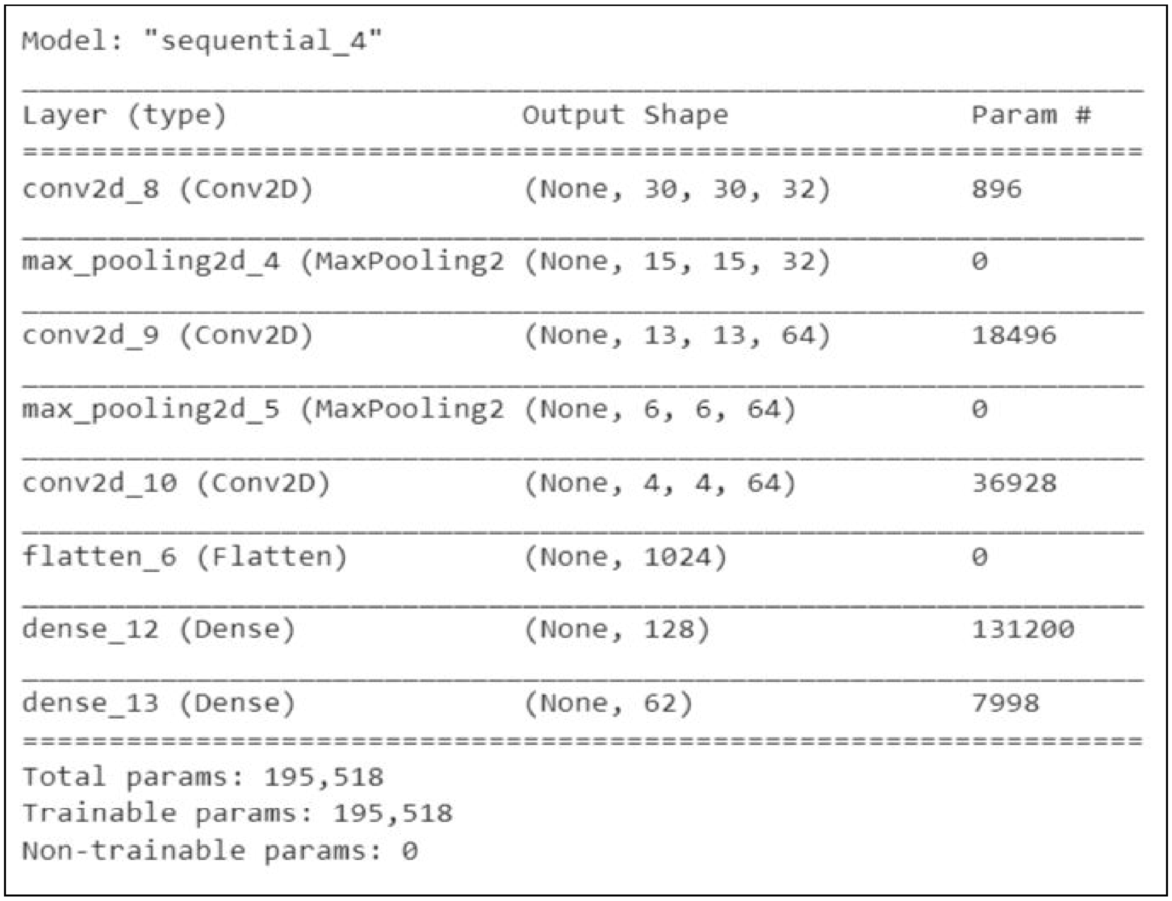
We can clearly see that the model created above is overfitted because of the use of multiple dense layers. To overcome this issue, we changed the architecture which has more 2D convolution layers and less dense layers.

The updated model is briefly described below:

The entire model is a sequential which has 8 layers. The first layer is a 2D convolution layer which takes the input image and works on it with a (3,3) filter of 32 size.The 2nd layer is a max pooling layer with a pool size of (2,2) to focus on the prominent features on the input. A combination of 2D convolution and max pooling layer is used 2 more times to focus on main features of the training set and also to train the neural network at much faster rate. The final 2D convolution layer is then flattened which is then fed into 2 dense layers.

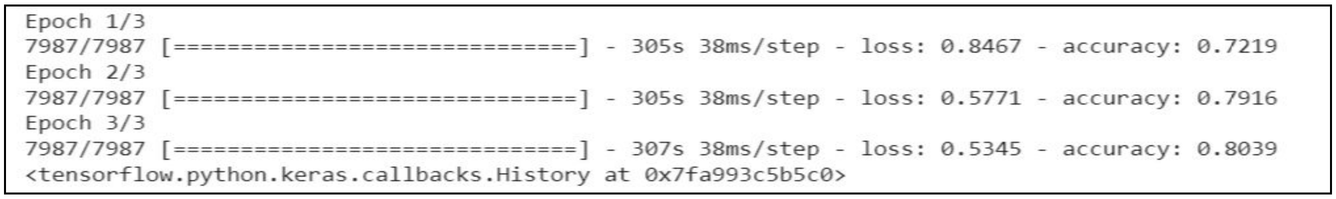
Below is the summary the updates neural network:

```
Input --> Conv2D --> MaxPooling2D --> Conv2D --> MaxPooling2D --> Conv2D
--> Flatten --> Dense-1 --> Dense-2
```



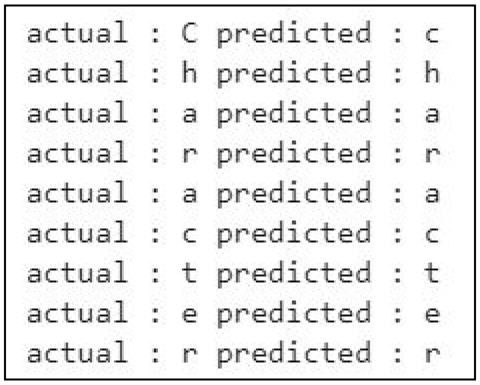
```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 30, 30, 32)        896
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 4, 4, 64)          36928
_________________________________________________________________
flatten_6 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_12 (Dense)             (None, 128)               131200
_________________________________________________________________
dense_13 (Dense)             (None, 62)                7998
=================================================================
Total params: 195,518
Trainable params: 195,518
Non-trainable params: 0
```

The updated model almost gave the same accuracy as that of the previous model but the major difference being the number neurons used to train on the handwritten characters are much less than that of used on the previous model. Each epoch took around 5minutesto execute which is way faster than what multiple dense layers took.



```
Epoch 1/3
7987/7987 [==============================] - 305s 38ms/step - loss: 0.8467 - accuracy: 0.7219
Epoch 2/3
7987/7987 [==============================] - 305s 38ms/step - loss: 0.5771 - accuracy: 0.7916
Epoch 3/3
7987/7987 [==============================] - 307s 38ms/step - loss: 0.5345 - accuracy: 0.8039
<tensorflow.python.keras.callbacks.History at 0x7fa993c5b5c0>
```

The updated model gave the following results for characters(‘C’,’h’,’a’,’r’,’a’,’c’,’t’,’e’,’r’) from unknown scanned document and on the same word presented above:



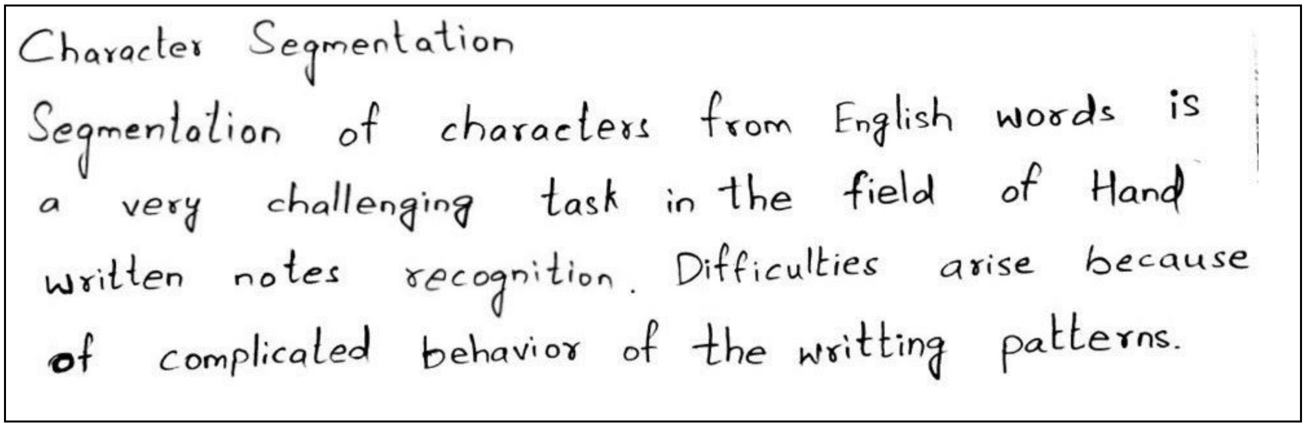
```
actual : C predicted : c
actual : h predicted : h
actual : a predicted : a
actual : r predicted : r
actual : a predicted : a
actual : c predicted : c
actual : t predicted : t
actual : e predicted : e
actual : r predicted : r
```

The only misclassification here is uppercase ‘C’ is recognized as lower case ‘c’ which does not affect the final result. Else, the whole word when fed to the update model as sequential letters, the neural network successfully detects all the characters of a word from an unknown input document precisely.

To Run The Code:
First, Input image is given to line_segmentation.ipynb in which the text is segmented into lines and store the results in the ‘data’ folder. The word_segemetation.ipynb file takes the output of line_segmentation.ipynb (lines segmented), segments the lines into words and stores the images in the ‘out’ folder. The words from the ‘out’ folder are fed into char_segmentation.ipynb which segments the words into characters. The characters segmented are fed into convolutional neural networks to recognize the handwritten text. The whole process undergoes on click of search button widget of search_func.ipynb

5. Results and Discussions

Input Image:

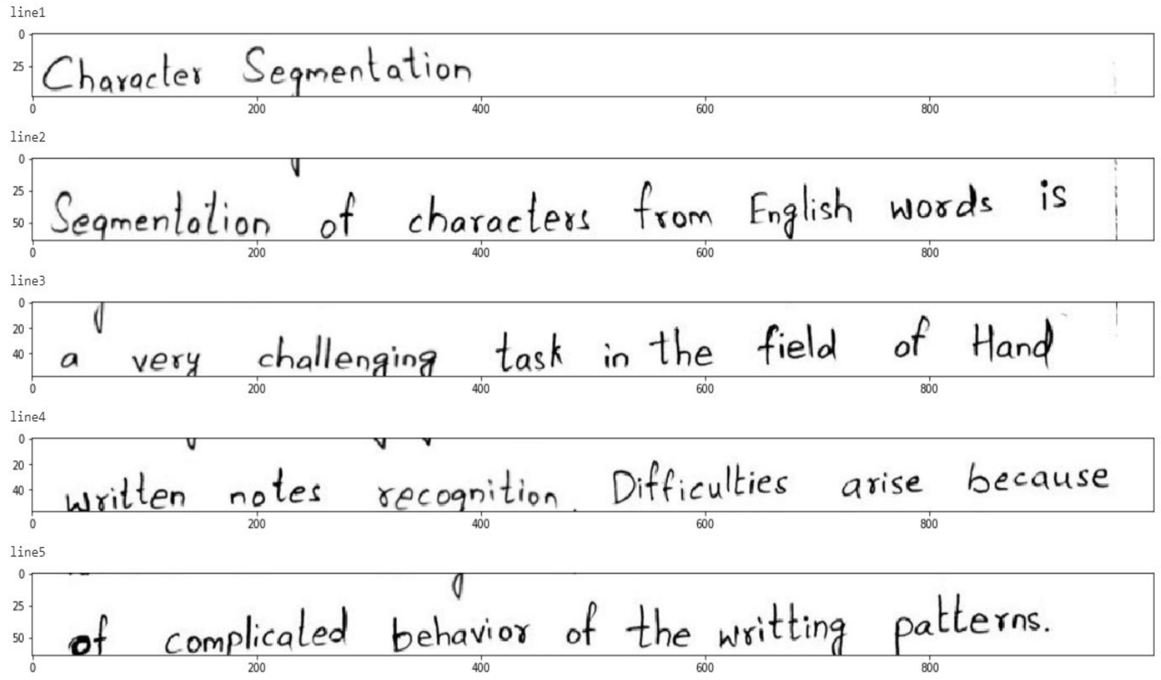


Character Segmentation

Segmentation of characters from English words is a very challenging task in the field of Hand written notes recognition. Difficulties arise because of complicated behavior of the writting patterns.

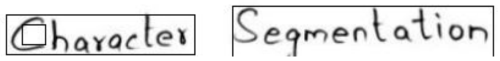
Image Segmentation

Results of line segmentation:

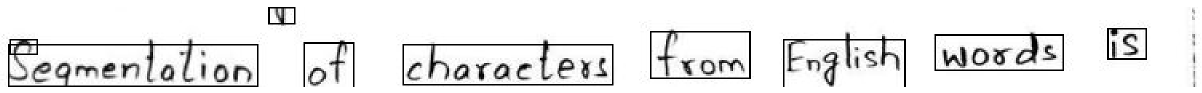


Results of Word Segmentation:

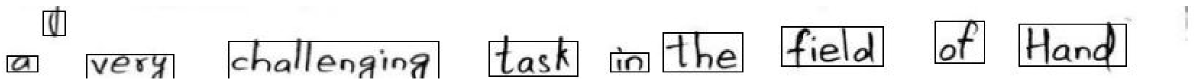
Line 1:



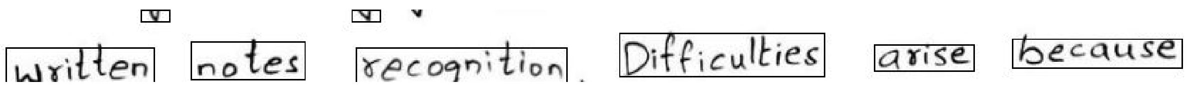
Line 2:



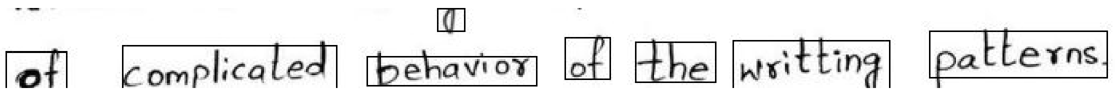
Line 3:



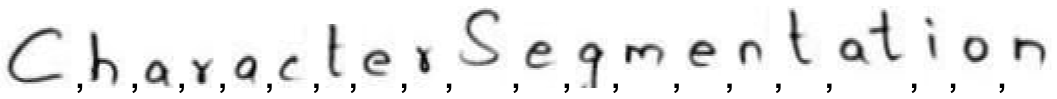
Line 4:

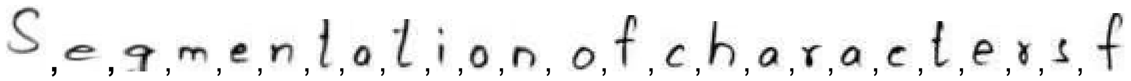


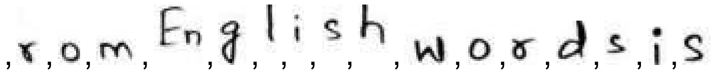
Line 5:

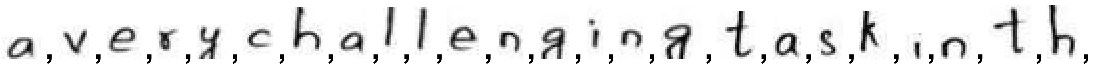


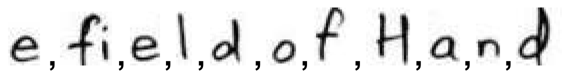
Results of character segmentation:

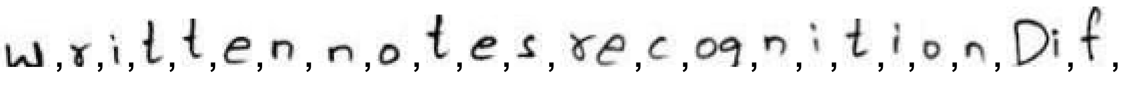
Line 1: 

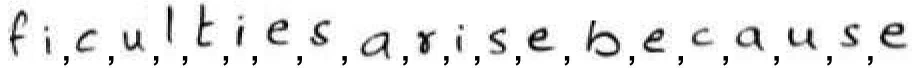
Line 2 : 

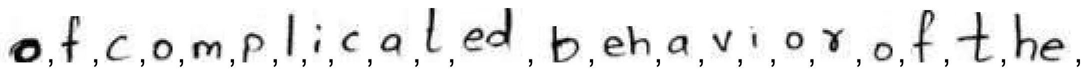


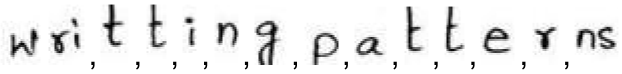
Line 3: 



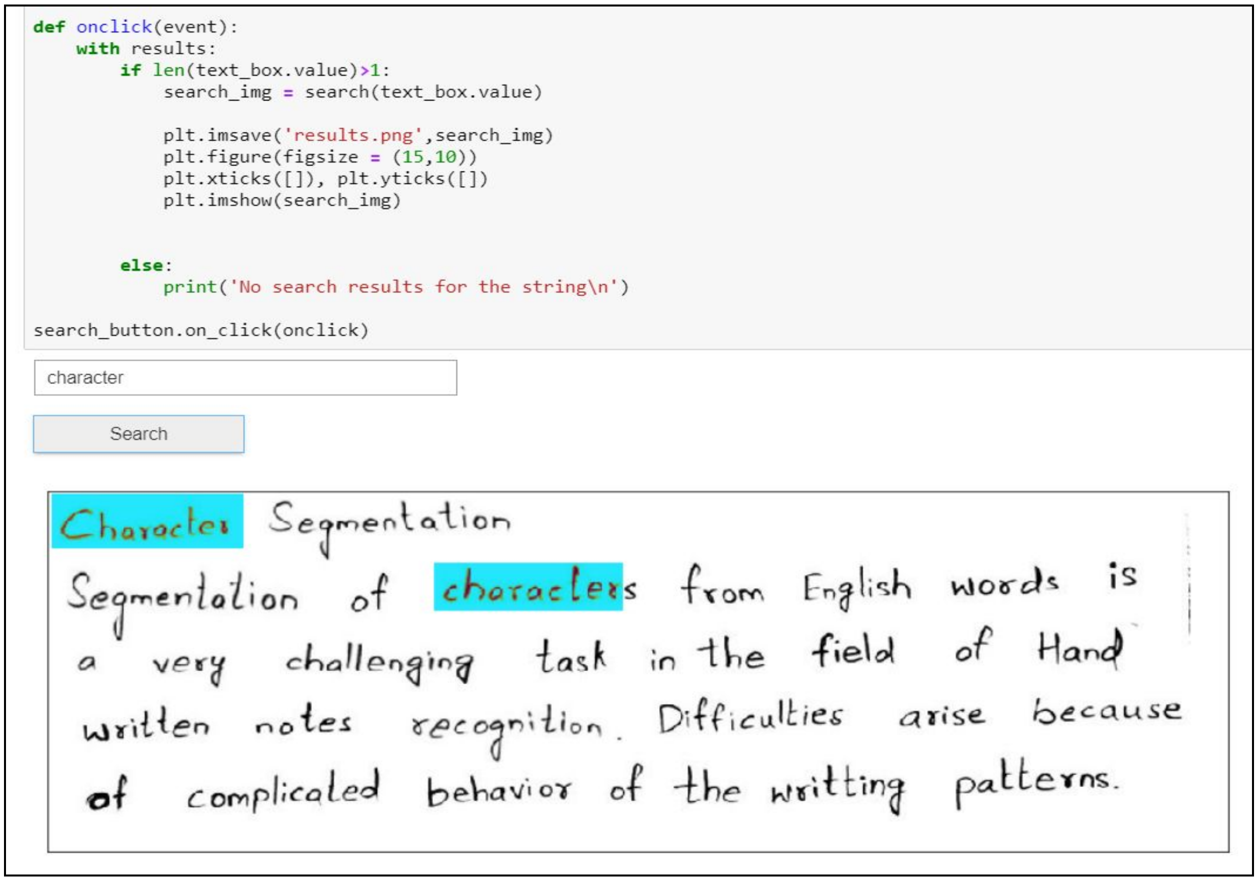
Line 4: 



Line 5: 



Results of search functionality (Integrating neural network with image segmented part):



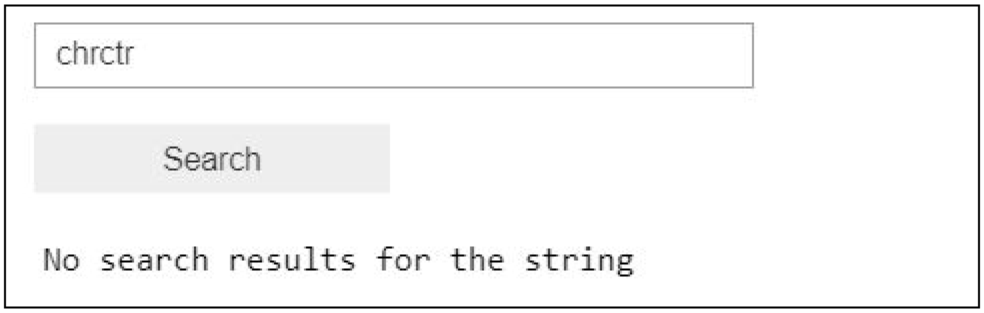
```
def onclick(event):
    with results:
        if len(text_box.value)>1:
            search_img = search(text_box.value)

            plt.imsave('results.png',search_img)
            plt.figure(figsize = (15,10))
            plt.xticks([]), plt.yticks([])
            plt.imshow(search_img)


        else:
            print('No search results for the string\n')

search_button.on_click(onclick)
```

character

Search

Character Segmentation

Segmentation of characters from English words is a very challenging task in the field of Hand written notes recognition. Difficulties arise because of complicated behavior of the writting patterns.



chrctr

Search

```
No search results for the string
```

The search functionality here implemented is case insensitive as we can see in the above screenshot that the search keyword is ‘Character’ which is capitalized / sentence case but in the search results we can see ‘character’ (lower case) as one of the matched words.

The handwritten notes being recognized here is multiple strokes or non-cursive and for cursive handwritten recognition, character segmentation is challenging and requires advanced algorithms because of the irregularity and inconsistent in size and shape. Single stroke handwritten recognition can be implemented with data augmentation by considering the algorithms for deskewing and deslanting the words in the scanned image.

6. Conclusion

We have successfully executed the search functionality for handwritten notes by implementing the concepts of Computer Vision (Bounding Box, Image Segmentation) and 2D Convolution Neural Network. The report also focuses on overfitting issues and also presents the sample architecture which leads to overfitting. We have demonstrated and presented the results of Image segmentation and Convolution NN which finally is integrated as the backend functionality of the search button for the scanned document.

7. Future Work

Currently this is limited to single stroke handwritten notes. Our future work includes using a similar approach and dealing with all types of handwriting notes. Also, applying neural networks for image segmentation is something we are planning to achieve our goal.

8. References

Link -
https://amsemodelling.com/publications/advances_in_modelling/Signal_Processing_and_Pattern_Recognition/57_vol_1/Handwritten_English_character_segmentation_by_baseline_pixel_burst_method_(BPBM).pdf