OPTICAL CURSIVE HANDWRITTEN RECOGNITION USING VPP & TDP NATIVE SEGMENTATION ALGORITHMS AND NEURAL NETWORKS (PYTORCH)

[1]G. Tirumalesh, [2]K. L. Srinivas, [3]K. Pratima, [4]N. Arun, [5]Y. Hemanth
[1]student, [2]student, [3]student, [4]student, [5]student
Department of Computer Science and Engineering,
Anil Neerukonda Institute of Technology and Sciences, Visakhapatnam, India.

Abstract- In the domain of Artificial Intelligence, scientist brought ultra-modern changes in many fields, one of them is image processing. This paper aims to represent the process of Converting handwritten text to computer typed document which is an optical cursive handwritten recognition (OCR) by using segmentation based algorithms like VPP (vertical projection profile), TDP (top down profile) and the other histogram (vertical and horizontal) projection algorithms to achieve the solution. Several other approaches were also available for the segmentation of text into individual characters. For feature extraction and character recognition pytorch which is an open source machine learning tool library in python used for computer vision and natural language processing.

Keywords: Vertical Projection Profile (VPP), Top Down Profile (TDP), Pytorch.

1. INTRODUCTION:

OCR means Optical Character Recognition, it is also known as optical character reader. OCR translates the text in the given images to machine readable format. Character recognition is classified into two types based on the text. They are machine printed text and handwritten text. It is difficult to work with handwritten text mainly in the case of cursive handwritten because it varies from person to person and there is no perfect line spacing and size of the character and margins etc in handwritten text. A single character is written in many styles so it is difficult to identify and translate the script into machine readable format or ASCII format. In this scenario, there is a step by step process to convert given script into each individual character by starting with line segmentation followed by word segmentation and finally the character segmentation. Finally predicting each individual character under a trained model using pytorch to recognize the character and combined together again to generate the original machine-readable script to the end user.

1.1 Literature Survey

Previous handwritten recognition uses various segmentation algorithms like heuristic, skew recognition techniques and written pressure detection. Almost every segmentation algorithm is based on horizontal and vertical projections to segment the script into individual characters. Even when there are text lines or characters which are overlapped on each other can be separated by adjusting the threshold value. The existed method is tested on more than 1000 text images of IAM datasets and by using these existing method 91.55% lines and 90.5% words are correctly segmented from the IAM dataset and also normalize 92% lines and words perfectly with invisible error rate.[9]

2. PROPOSED ALGORITHM

The process for the optical cursive handwritten recognition and the required algorithms for various level of segmentation and character recognition using pytorch are as follow

It comprises of six steps.

1. Image scanning

2. Pre-processing
3. Segmentation
4. Feature extraction
5. Classification
6. Post-processing

2.1 Image scanning:

The input image can be obtained either by scanning the already existing handwritten image file (png, jpg) or by capturing the image instantly to provide input data to the model.

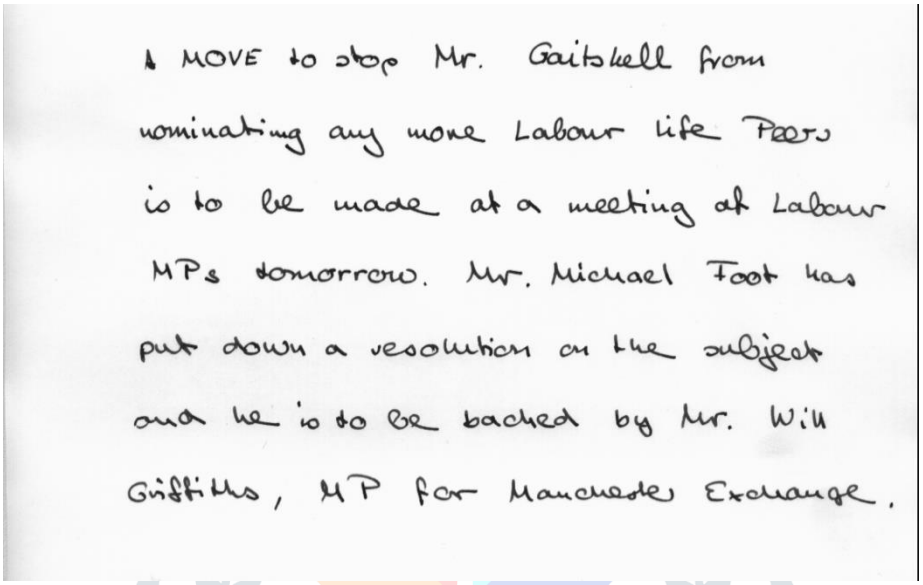

Fig 2.1 Scanned input image

2.2 Image pre-processing:

The main goal here is to make the input image free from noise. As a first step to move on convert the RGB image to a gray scale image and gently sharpen the given input image to avoid loss of edges. Calculate the mean gray intensity value to reduce the brightness of the obtained grey scale image on a threshold value of less than 0.65 and the contrast is increased to distinguish the character boundaries. The text which is present in the obtained result may turn dim and blur because of improper scanning of the text image. To overcome this, binarization plays a key role by converting the gray scale image where the values ranges between 0 and 255 to a binary image by making up a threshold value simply to decide like an on or off (0 or 1).

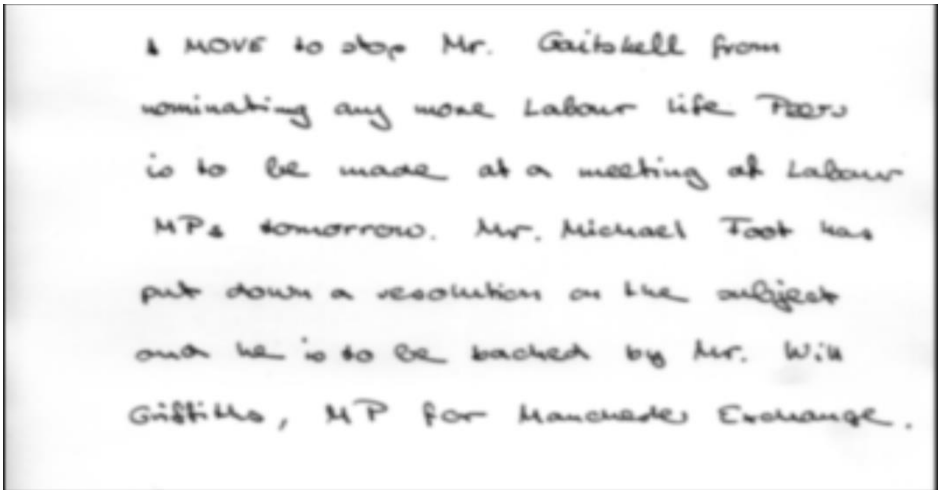

Fig 2.2 Blur image (for noise removal)

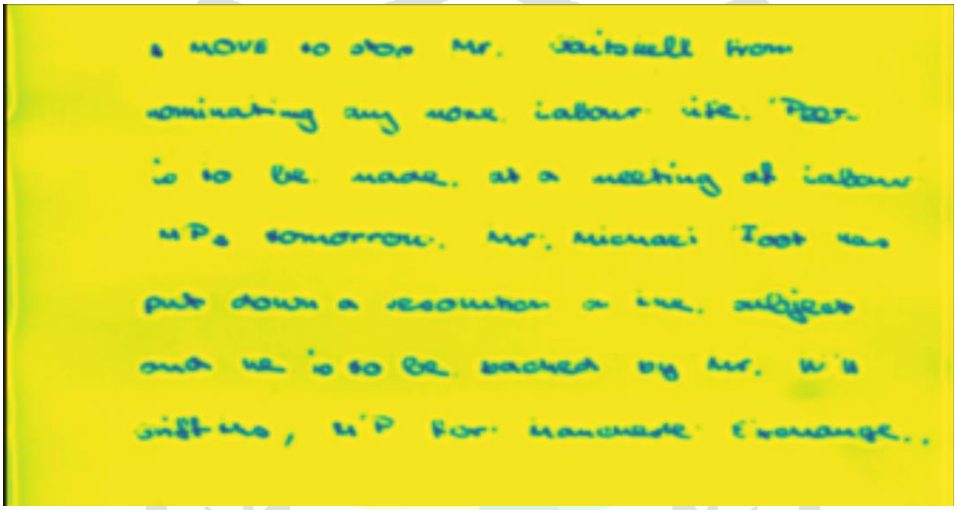

Fig 2.3 Binary image in color contrast

2.3 Segmentation

Basically, there are three levels of segmentation. Line segmentation, Word segmentation and Character segmentation.

2.3.1 Line segmentation

Horizontal histogram projections are used in segmenting the entire script present in the input image into individual lines as shown in the figure below

The primary task here is to extract each individual line from the given input image. This can be obtained by applying horizontal histogram projection to the pre-processed image and then generate the threshold line value by adjusting the average value for those horizontal projections. Graphical representation of horizontal histogram projection is shown in the figure 2.4 below.[6]

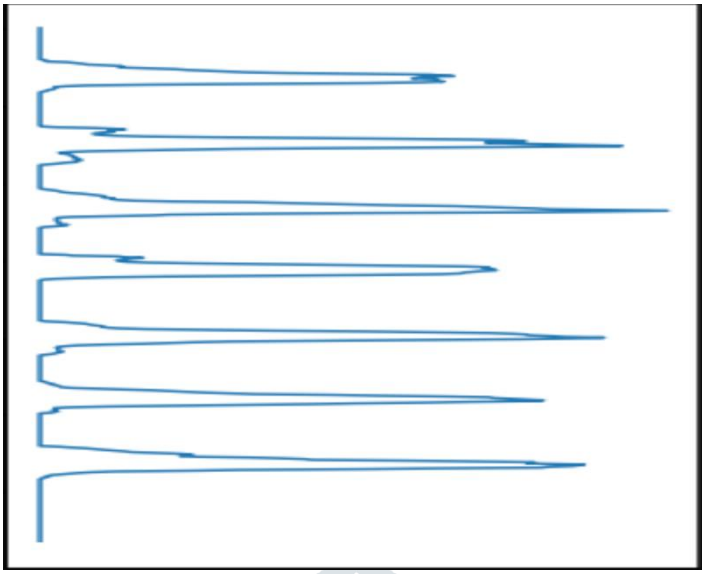

Fig 2.4 Horizontal projection graph

Finally, Lines can be segmented from the given input script by obtaining the break points by making use of the average threshold line value obtained from the above graph and having comparisons with each and every horizontal projections.

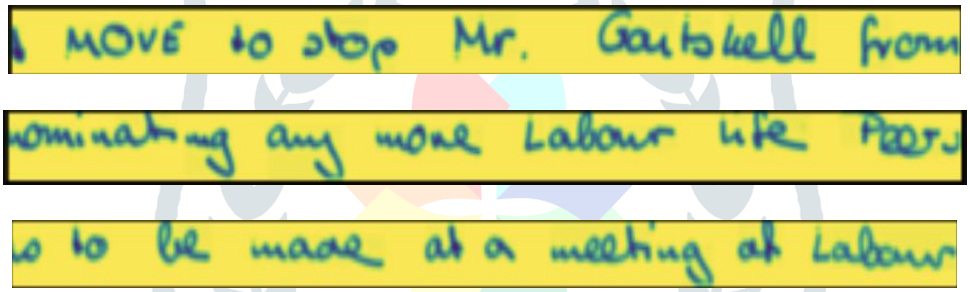

Fig 2.5 Segmented lines from the image

2.3.2 Word segmentation

Each word is treated as an object (Contour – in terms of image processing). Contour can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. Here, Contours are useful for object detection where each object is a word.

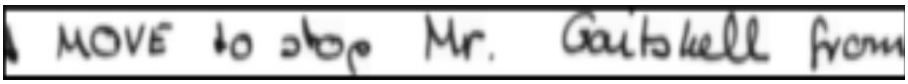

Fig 2.6 segmented line

The main reason for making use of contours here is each word can be treated as a curve joining all the continuous points along the boundary since it is cursively written. But sometimes there may be gaps in between the letters of a single word which causes the word to be split into two or more words as they are not continuous points joining as a curve.

This type of words can be identified by making use of the minimum threshold value which is obtained by taking the average separation distance between the words and can be rejoined back as a single word (Contour) where the separation distance between the words less than the minimum threshold value.[7]

Minimum threshold value = sum of separation distances between words in the line/ no of words in the line

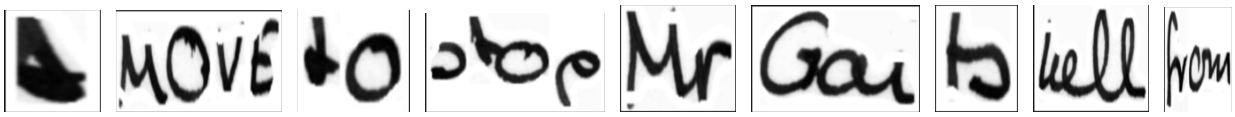

Fig 2.7 First level word segmentation

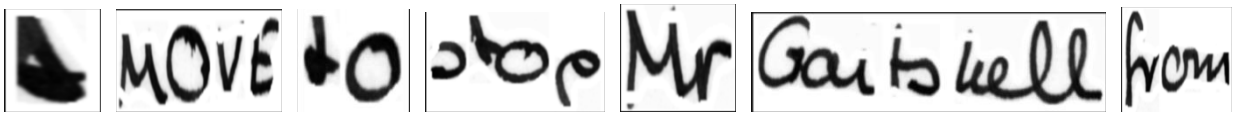

Fig 2.8 Second level word segmentation

2.3.3 Character segmentation

Segmenting each word into individual characters can be obtained by making use of two native algorithms:

1) VPP (Vertical Projection Profile)
2) TDP (Top Down Profile)

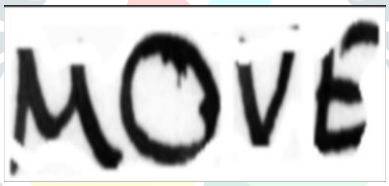

Fig 2.9 segmented word

VPP is a plot which maintains the total number of white pixels in vertical direction of the binary image. Characters can be segmented at the point where the VPP value is zero (0) for some certain no of times (Threshold). But in the case of touching characters, the VPP value can never be zero even though the characters should be segmented (connected components).[1]

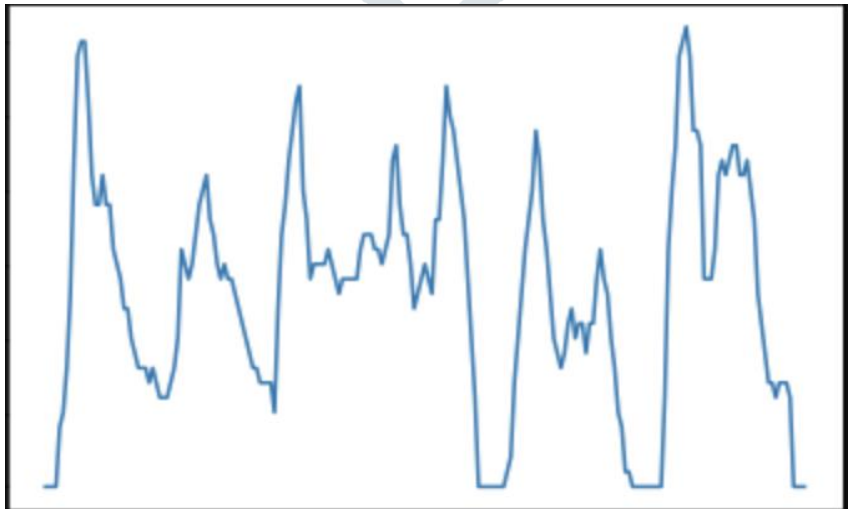

Fig 2.10 VPP intensity graph for word 'MOVE'

Connected components can be identified by making use of characters width and height. When the width of the character is greater than 0.8 times the height of the character then it is identified as a connected component otherwise it is a single character as per basic font size measurement.[1]

Width > 0.8 * height (Connected component)

Connected component single character single character

Fig 2.11 First level VPP character segmentation

TDP is a plot which maintains the first white pixel in vertical direction of the binary image. Touching characters can be segmented into individual characters by taking the combined value of both VPP AND TDP. Then obtain the minimum value in the graph where we can segment them into individual characters and continue this process recursively until no more touching character found in a single word. [2]

Fig 2.12 touching characters (connected component)

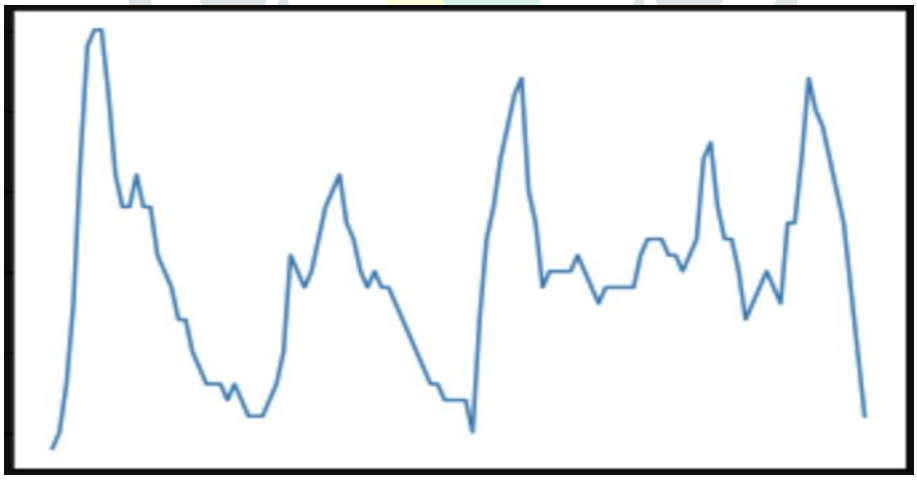

Fig 2.13 VPP intensity graph for word ‘MO’

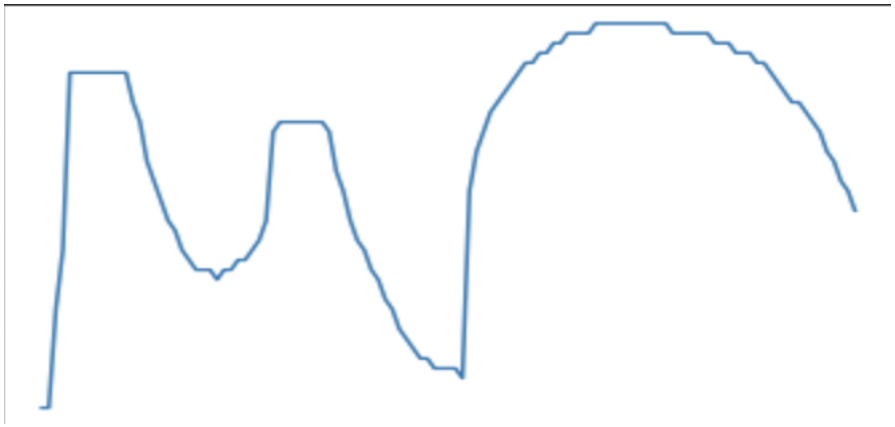

Fig 2.14 TDP intensity graph for word ‘MO’

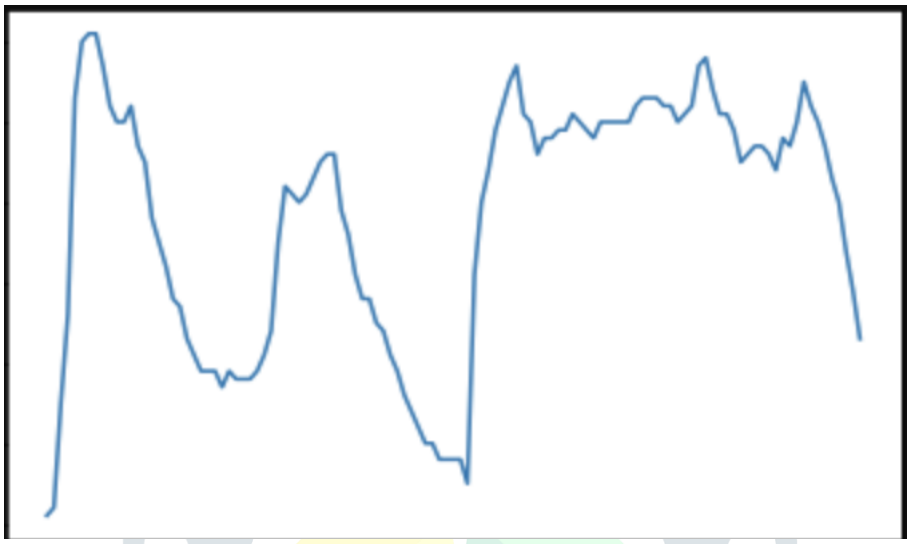

FIG 2.15 Combined intensity graph for word ‘MO’

Fig 2.16 Final level character segmentation

2.4 Feature Extraction:

The main goal here is to extract the features from the segmented characters which are required to train the data.

This process comprises of zero padding, convolution layer, activation function, max pooling and flatten. As a first step add zeros to the image to overcome the loss of edges termed as zero padding. Then apply multiple layers of convolution and max pooling filters(kernels) to obtain an image with reduced in size where each move is a stride. Max pooling comes with selecting the max value from the filter and replace it with the remaining and the same way the average pooling works by taking the average pixel value. Now the activation function comes into picture where the ReLU activation function identifies all the negative pixel values and replace it with zero without any change in the positive pixel values. Finally flatten the image by reshaping the image obtained.

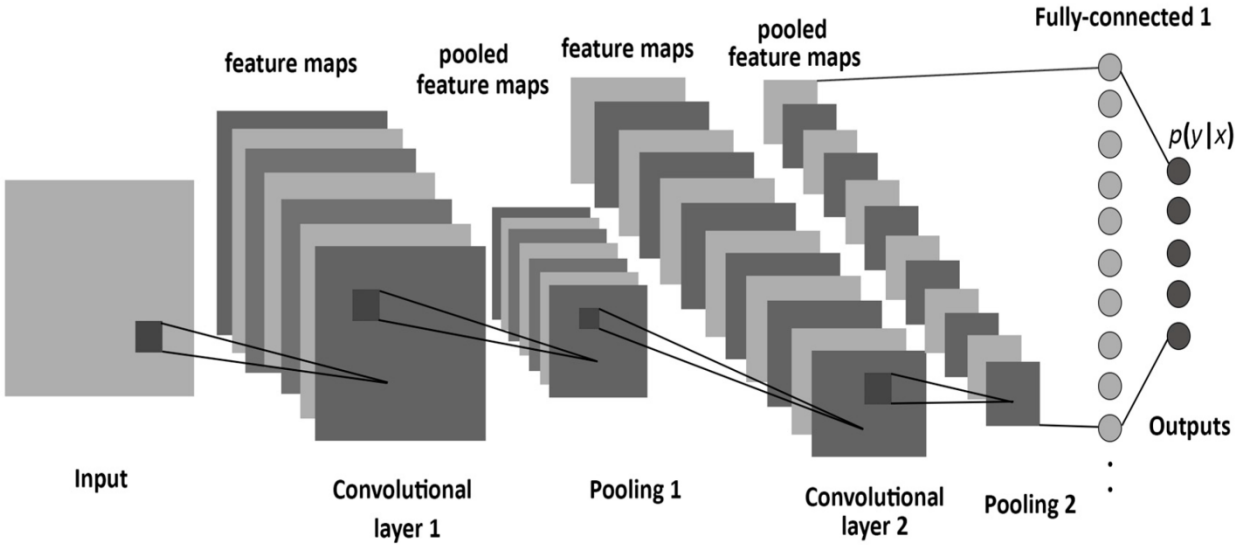

Fig 2.17 Feature extraction of characters

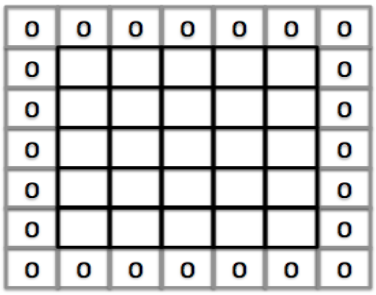

Fig 2.18 Zero padding

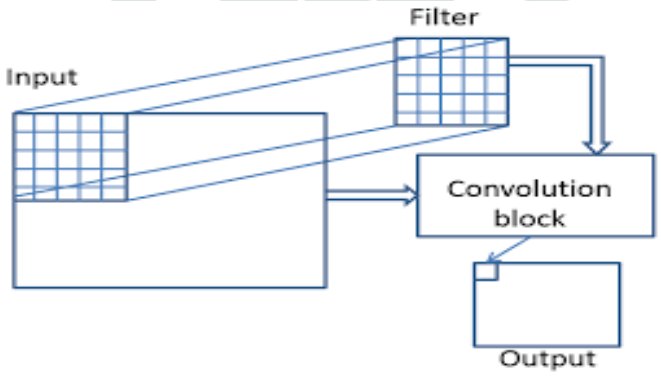

Fig 2.19 Convolution layer

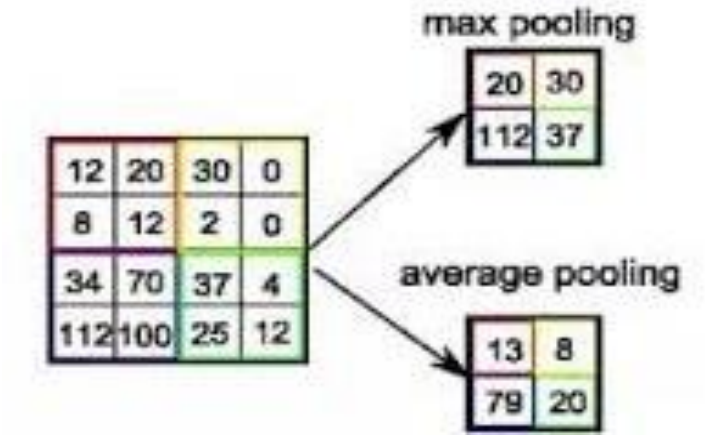

Fig 2.20 Max pooling and Average pooling

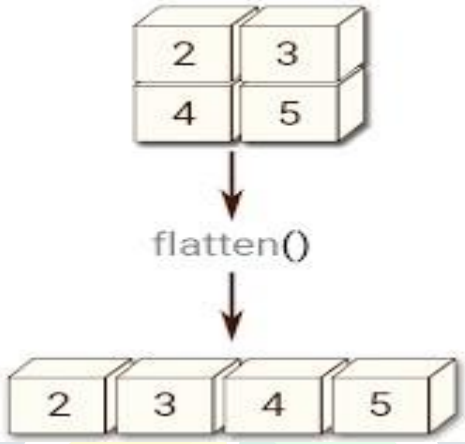

Fig 2.21 Flatten the image

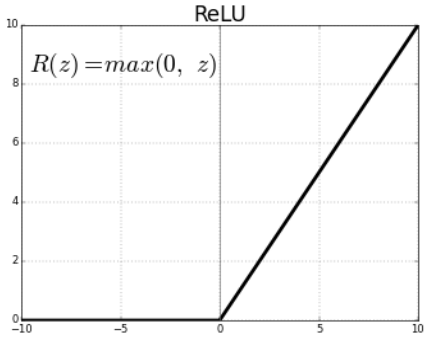

Fig 2.22 ReLU activation function

2.5 Classification:

Finally, Classification is done using a fully connected layer where we get the probabilities of each and every class for the given input character. Classify the given input character to their respective class by selecting the class with the maximum probability. In total there are classified into 62 classes (0 to 9, a to z, A TO Z)

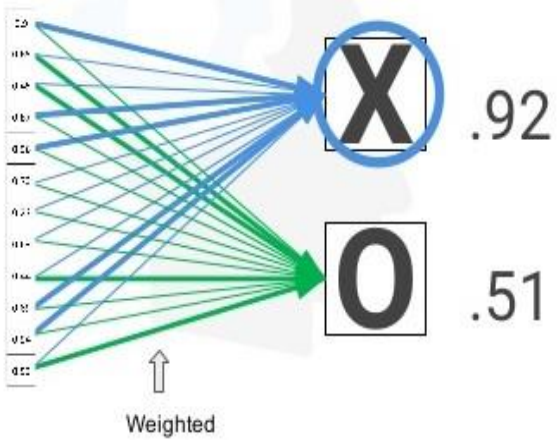

Fig 2.23 Fully connected layer with classes (x and o) along with probabilities

2.6 Post-processing:

As a final step obtain the accuracy for all the levels of segmentation and the character recognition by minimizing the error rate. Then combine all the recognized characters into word, words into lines and lines into the original script present in the image.

Final Result = **M O V E**

Fig 2.24 Final result

3. PYTORCH LIBRARY TOOL:

Pytorch is an open source machine learning library based on torch library used for applications such as computer vision and natural language processing.

It is a very popular framework for deep learning. The feature extraction and the classification stages are implemented under pytorch library tool. As a first step install all the required modules/packages to train the data using pip namely efficientnet-pytorch and torchsummary. Then include all the required modules by importing them into the python script (torch, torchvision, torch.nn, torch.util, torch.autograd, torch.optim, torchvision.transforms, Efficientnet).

Prepare the Dataset for training the model from the NIST dataset (700000 images) by dividing them into two around 600000 images as training data and remaining as test data.

Create a class named DATASET for listing all the training images and their respective labels for training. Download the pretrained model efficientnet-b0 and assign all the required parameters like batch size, learning rate,

error rate, no of classes and transformations if any. Finally train the model under certain epcohs until the loss gets minimized and accuracy gets increased.

Finally predict each input segmented character under the trained model to be classified into one of the class.

4. EXPERIMENTAL RESULTS AND ANALYSIS:

This system is trained under around 1600 text images (paragraphs) of IAM dataset with almost 5678 labelled sentences, 13353 isolated and labelled text lines, 115,320 isolated and labelled words with an accuracy of around 98% in terms of line segmentation, 93 % in terms of word segmentation and 88% in terms of character segmentation.[8]

In terms of character recognition, this system is trained under around 623,000 images of 62 different characters (0 to 9, a to z, A to Z) with an accuracy of around 96% in character recognition.

5. CONCLUSION:

This paper mainly carries out a study on segmenting the connected components (touching characters). There are many more challenges involved in the optical cursive handwritten recognition like the skewness, pressure detection etc can be treated as a future study. The proposed method for segmenting the connected components (touching character) is by using VPP (Vertical projection profile) and TDP (Top Down Profile) and various other histogram projections (horizontal and vertical) for the line and word segmentation respectively. pytorch is a python library tool used for the recognition of the segmented characters. [4]

6. ACKNOWLEDGMENT:

The project team members would like to express our thanks to our guide B. shiva jyoti Assistant professor of Computer Science and Engineering Department, Anits for her valuable suggestions and guidance in completing out project model.

7. REFERENCES:

[1]Nafiz arica, Student Member, IEEE, and Fatos T. Yarman-Vuraj, Senior Member IEEE, “Optical Character Recognition for Cursive Handwriting”, IEEE transaction on pattern analysis and machine intelligence, vol. 24 no. 6, June 2002

[2] Subhash Panwar and Neeta Naina, “A Novel Segmentation Methodology for Cursive Handwritten Documents”, IETE JOURNAL OF RESEARCH, VOL 60-NO 6 NOV-DEC 2014

[3] Nibaran Das Sandip Pramanik, Subhadio Basu, Punam Kumar Saha, “Recognition of handwritten Bangla basic characters and digits using convex hull based feature set”, 2009 International conference on Artificial intelligence and pattern recognition (AIPRL-09)

[4]Abhishek Bala and Rajib Saha, “An IMPROVED Method for Handwritten Document Analysis using Segmentation, Baseline Recognition and Writing Pressure Detection”, 6th International Conference on Advances In Computing Communication, ICACC 2016, 6-8 September 2016, Cochin, India, Elesevier-2016

[5] Kanchan Keisham and Sunanda Dixit, “Recognition of Handwritten English Text Using Energy Minimisation”, Information Systems Design and Intelligence Applications, Advances in Intelligent Systems and Computing, Bangalore, India, Spinger-2016

[6] Namrata Dave," Segmentation Methods for Hand written Character Recognition", International Journal of Signal Processing, Image Processing and Pattern Recognition Vol.8, No.4 (2015), pp.155-164

[7]G. Louloudisa, B.Gatosb, I.Pratikakisb, C.Halatsis, "Text line and word segmentation of handwritten documents" a Department of Informatics and Telecommunications, University of Athens, Greece Computational Intelligence Laboratory, Institute of Informatics and Telecommunications, National Center for Scientific Research Demokritos, 15310Athens, Greece

[8] IAM dataset http://www.fki.inf.unibe.ch/databases/iam-handwriting-database

[9] Offline handwritten character recognition using neural networks https://www.researchgate.net/publication/239765657_Offline_handwritten_character_recognition_using_neural_network