EC3066D

ARTIFICIAL INTELLIGENCE THEORY AND PRACTICE

MINI PROJECT

HANDWRITTEN TEXT DETECTION USING OPENCV AND PYTHON

Group-G2

R V V MANIKANTHA SAI - B180559EC
ROSHAN MADIPALLI - B180570EC
BANDARU BHAVYANATH - B180399EC

1 Aim:

The main aim of the project is Handwritten Text Recognition (HTR). HTR is the task of transcribing images of handwritten text into the digital text. In HTR, the text is written, captured by a mobile phone camera and then the resulting images are processed as input to return its text format. We need openCV and CNN for achieving this task. Our goal is to design a model that transcribes the images to text with great accuracy.

2 Datasets:

1. A_Z Hand_Written dataset
2. MNIST dataset

The format of the data files is csv(Comma Separated Values). Each row represents an image and contains a label in the first column and followed by 784 pixel values for 28 X 28 images.

2.1 A_Z Hand_Written dataset

The dataset will comprise of uppercase English alphabets. Here the dataset will have a label and pixel values which lie from 0 to 255. There will be a total of 372450 instances, and the total number of attributes is 784 and a label.

2.2 MNIST dataset

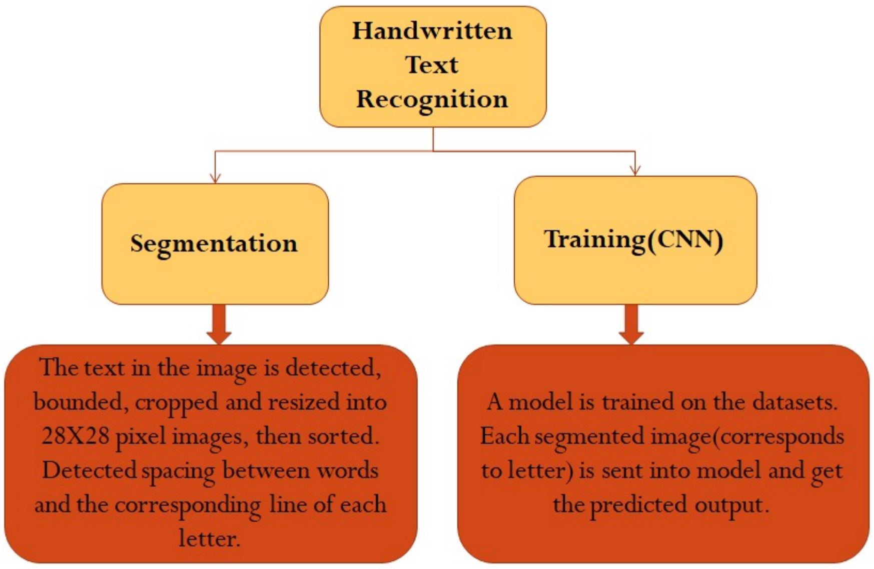
The dataset will comprise of digits from 0 to 9. Here the dataset will have a label and pixel values which lie from 0 to 255. There will be a total of 42000 instances, and the total number of attributes is 784 and a label.

3 Architecture:

Converting a handwritten text into its digital form, this problem can be divided into two modules

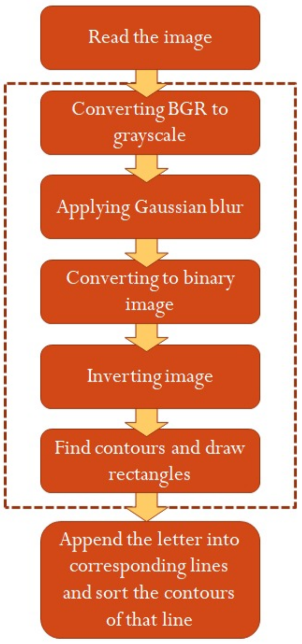
1. **Segmentation module:** In this module image is taken as input, letters are detected, bounded, cropped, resized and then segmented.

2. **Training module:** In this module prediction of letters will happen and the output of the segmentation module will be the input of the Training module.

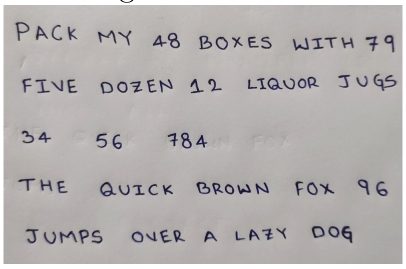

Handwritten Text Recognition
Segmentation
Training(CNN)
The text in the image is detected, bounded, cropped and resized into 28X28 pixel images, then sorted. Detected spacing between words and the corresponding line of each letter.
A model is trained on the datasets. Each segmented image(corresponds to letter) is sent into model and get the predicted output.

3.1 Segmentation module:

Segmentation model is very important for this project as its output will be the input of the other module.


Read the image
Converting BGR to grayscale
Applying Gaussian blur
Converting to binary image
Inverting image
Find contours and draw rectangles
Append the letter into corresponding lines and sort the contours of that line

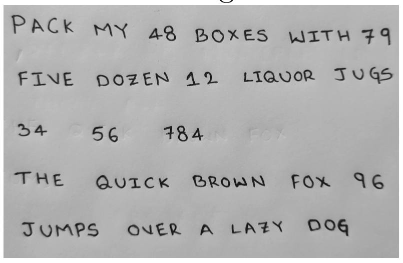
1. **Read the image:** We used OpenCV for performing operations and manipulating images.An image is read and is plotted to make sure it is read perfectly. That image contains letters that need to be cropped and resized into 28 X 28 images by the end of the segmentation model.



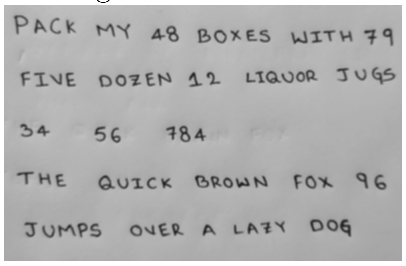
2. **Detecting the letters:**

   (a) **Step 1:** In OPENCV images are stored in BGR format. Convert a BGR image to Grayscale image. A

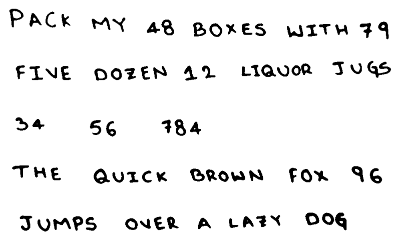
BGR image contains 3 channels,but a Grayscale image consists of a single channel.



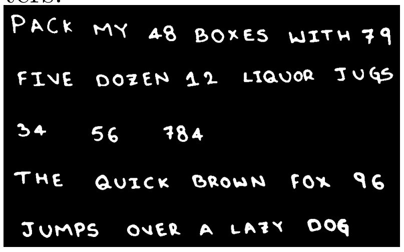
(b) **Step 2:** Applying Gaussian blur to the Grayscale image. This step is done to remove any noise and disturbances in the image. If the image is blurred the colour intensities can be recognised easily from the image.



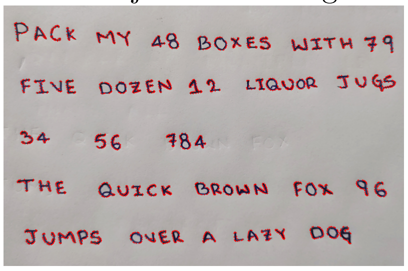
(c) **Step 3:** Thresholding: If pixel value is greater than a threshold value, it is assigned one value (white), else it is assigned another value (black). This is done to detect letters.



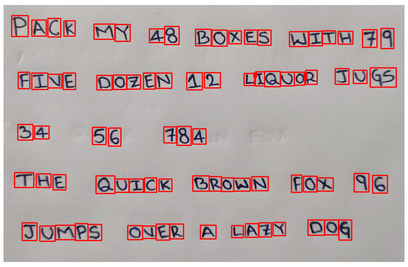
(d) **Step 4:** Inverting image i.e make the letters white and the background black to get the counters around the letters.



(e) **Step 5: Finding and Drawing contours**
The 'findcontours()' and 'drawcontours()' are the methods used for finding and drawing contours which are generally borders of a detected object in an image.



3. **Bounding and cropping:** Each letter is bounded by a rectangle using Boundingrect() method and the coordinates of the corners of the rectangle are stored. By using the values of the co-ordinates we cropped the image and stored it in a list. After cropping each image is padded with zeros to ensure that letter will be in the centre of the image.



4. **Resizing:** For each cropped image in the list, we resize them to 28 X 28 pixels because the output images from this module are going to be the input of the other module. For that module input must be images of size 28 X 28 pixels as the training data of that module is of that format.

5. **Detecting whether the letter is present after space or in next line:**

   (a) **Algorithm for detection of space:**
   For detecting a space between two words we have taken an arbitrary value. If the difference between the x coordinate values of the centroid of the current letter and previous letter is greater than this arbitrary value then we conclude that there exists a space between both the letters.

   (b) **Algorithm for detection of line:**

      i. Run a loop along the contours and find the minimum y coordinate of the contours and the maximum value of height.

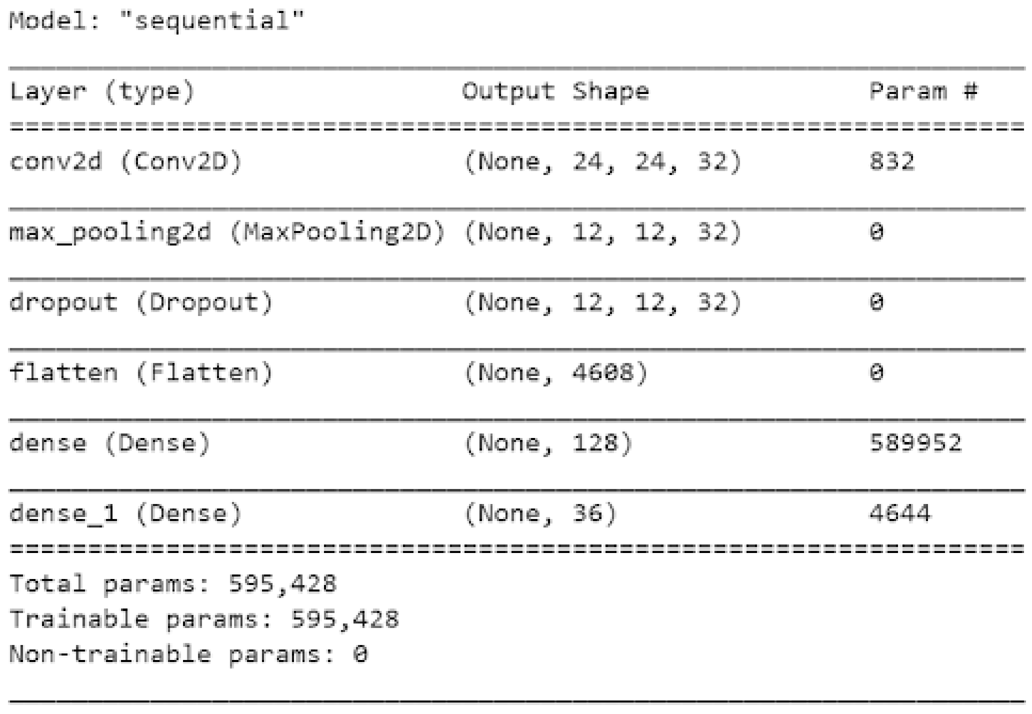
      ii. From this minimum y, we get to know the approximate y coordinate of the first line and the height obtained is the maximum height of the letters in this list.

      iii. Create a list of lists where each list stores the letters of the corresponding line.

      iv. Take absolute value of difference between the y coordinate of each letter and the minimum y and divide this difference with (height_constant*maximum height). Convert this value into an integer which gives you the line number of letter. This line number starts from zero. Append each letter into the corresponding list.

4 Training module:

We have concatenated the A_Z Handwritten data and the MNIST data. In that We have taken 75The first hidden layer is 2Dimensional Convolution layer with kernel size 5 X 5 and 32 nodes. The activation function used is 'relu'. And then max pooled with 2 X 2 and over fitting is reduced using dropout of 0.3. Then two more layers are added with 128 and n nodes respectively, where n is the number of possible outputs (here n = 36).



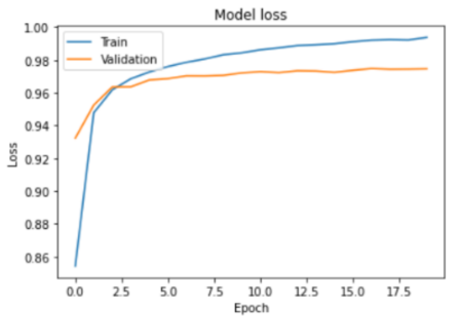
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 24, 24, 32)        832
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 12, 12, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 12, 12, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0
_________________________________________________________________
dense (Dense)                (None, 128)               589952
_________________________________________________________________
dense_1 (Dense)              (None, 36)                4644
=================================================================
Total params: 595,428
Trainable params: 595,428
Non-trainable params: 0
_________________________________________________________________
```

Dropout	Training Accuracy	Test Accuracy
0.1	99.45%	97.46%
0.2	99.18%	97.19%
0.3	99.03%	97.67%
0.4	98.87%	97.77%

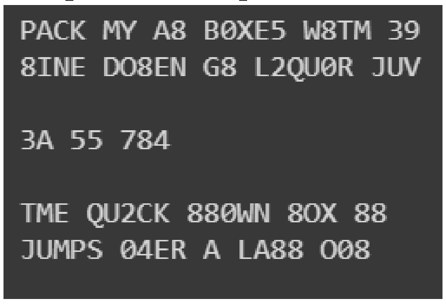
Table 1: Training Accuracy and Test Accuracy with respect to Dropout

4.1 Training and Test accuracy plots:

1. (a) Train Test accuracy with dropout 0.1

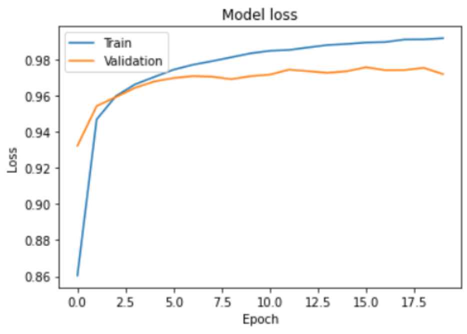


   (b) Output for dropout 0.1

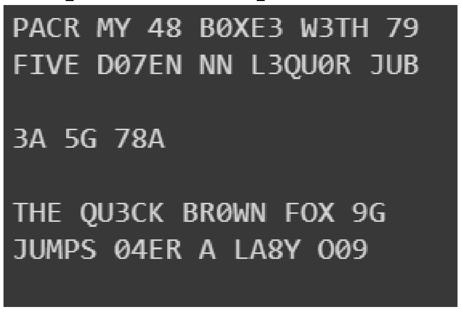


```
PACK MY A8 B0XE5 W8TM 39
8INE DO8EN G8 L2QU0R JUV

3A 55 784

TME QU2CK 880WN 8OX 88
JUMPS 04ER A LA88 008
```

2. (a) Train Test accuracy with dropout 0.2

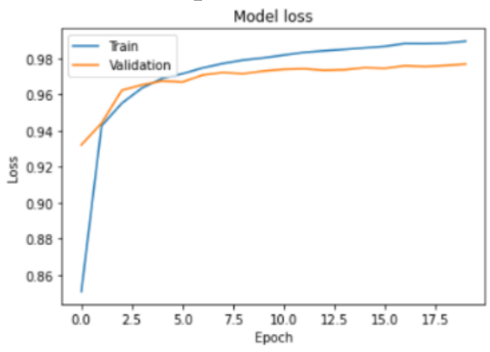


   (b) Output for dropout 0.2

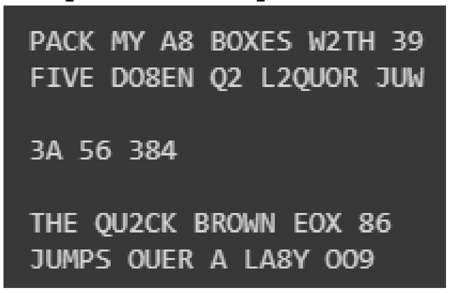


```
PACR MY 48 B0XE3 W3TH 79
FIVE D07EN NN L3QU0R JUB

3A 5G 78A

THE QU3CK BR0WN FOX 9G
JUMPS 04ER A LA8Y 009
```

3. (a) Train Test accuracy with dropout 0.3

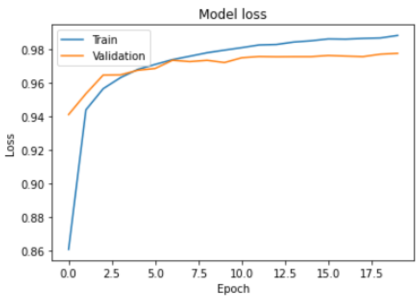


   (b) Output for dropout 0.3

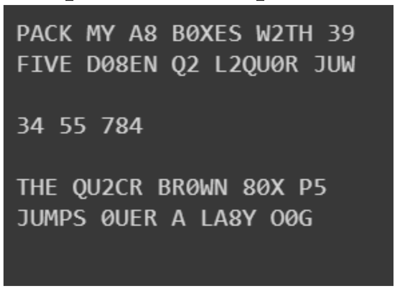


```
PACK MY A8 BOXES W2TH 39
FIVE DO8EN Q2 L2QUOR JUW

3A 56 384

THE QU2CK BROWN EOX 86
JUMPS OUER A LA8Y OO9
```

4. (a) **Train Test accuracy with dropout 0.4**



(b) **Output for dropout 0.4**



```
PACK MY A8 B0XES W2TH 39
FIVE D08EN Q2 L2QU0R JUW

34 55 784

THE QU2CR BR0WN 80X P5
JUMPS 0UER A LA8Y O0G
```

4.2 Results:

Dropout is generally used to reduce overfitting. We have used 4 different values for dropout. The change in train, test accuracy has no significant difference. When we have kept the real time images, the models with 0.2,0.3 performed better than the other 0.1, and 0.4. So we took 0.3 as the final value for dropout.

4.3 Training and Test accuracy plots:

We have trained the model with different number of dense layers and different number of filters in the convolution layers. Initial model contains a Conv2D layer with 32 filters and one dense layer with 128 nodes.

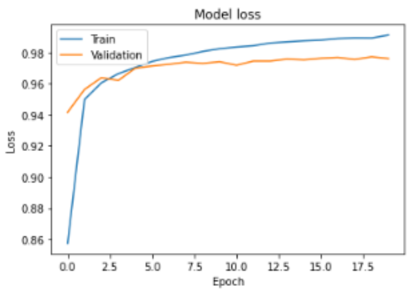
S.No	Experiment	Training Accuracy	Test Accuracy
1	Original Model	99.03%	97.67%
2	Model contains 2 dense layers one with 128 nodes and the other with 64 nodes	99.19%	97.60%
3	Model contains 1 dense layer with 64 nodes instead of 128 nodes	98.35%	97.23%
4	Model contains a conv2D layer with 64 filters.	99.31%	97.56%

Table 2: Training Accuracy and Test Accuracy under different experiments

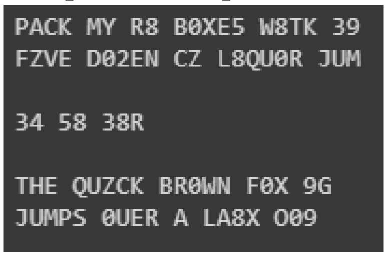
S.No	Experiment	Training Accuracy	Test Accuracy
1	Model with stochastic gradient descent	99.44%	97.55%
2	Model with Adam optimizer	99.03%	97.67%

Table 3: Training Accuracy and Test Accuracy under different optimizers

1. (a) **Train Test accuracy with experiment 1**

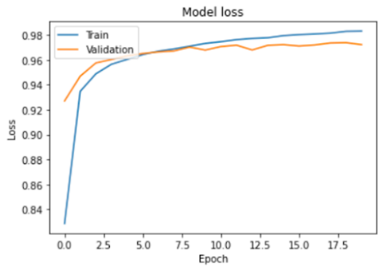


   (b) **Output for experiment1**

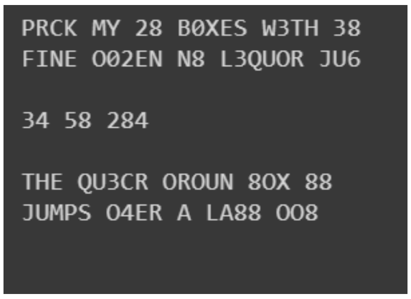


```
PACK MY R8 B0XE5 W8TK 39
FZVE D02EN CZ L8QU0R JUM

34 58 38R

THE QUZCK BR0WN F0X 9G
JUMPS 0UER A LA8X 0O9
```

2. (a) **Train Test accuracy with experiment 2**

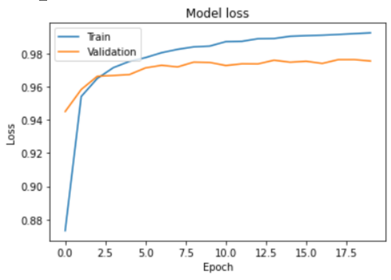


   (b) **Output for experiment 2**

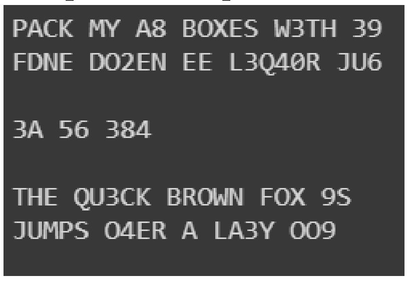


```
PRCK MY 28 B0XES W3TH 38
FINE O02EN N8 L3QUOR JU6

34 58 284

THE QU3CR OROUN 80X 88
JUMPS O4ER A LA88 OO8
```

3. (a) **Train Test accuracy with experiment 3**



   (b) **Output for experiment 3**



```
PACK MY A8 BOXES W3TH 39
FDNE DO2EN EE L3Q40R JU6

3A 56 384

THE QU3CK BROWN FOX 9S
JUMPS O4ER A LA3Y OO9
```

4.4 Results:

From the above experiments we have observed that the original model is performing better than the other three.

5 Challenges:

1. During segmentation of letters, first we have cropped based on the bounding rectangle, the observed prediction is poor. So we have zero padded the image, so the image gets positioned in such a way that features can be detected.This has improved the performance
2. Initially we have trained on only uppercase alphabets. The performance of the model is better. Then we have combined both alphabets dataset and numbers dataset. Initially we shuffled the dataset but the observed prediction is bad. The reason for that is uneven distribution of data in training and testing sets. So we have used startify on data based on labels inorder to have uniform distribution..
3. The performance when both uppercase letters and numbers are trained is poor because we have similar symbols like ‘A’,4 ;‘O’,0; ’Z’,’7’; ‘S’,’5’.