International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 04, April-2020

Handwritten Text Detection using Open CV and CNN

Dr. S Jessica Saritha
Assistant Professor, Dept of CSE, JNTUACE,
Pulivendula, AP, India

G Hemanth Kumar
Student, Dept of CSE, JNTUACE,
Pulivendula, AP, India

K R G Deepak Teja
Student, Dept of CSE, JNTUACE,
Pulivendula, AP, India

S Jeelani Sharief
Student, Dept of CSE, JNTUACE,
Pulivendula, AP, India.

***Abstract*:- 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 scanner 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.**

Keywords – CNN, Handwritter Text Recognition (HTR), openCV, Transcription

I. INTRODUCTION

Now a day's people have been using ebooks which do not occupy any space and ample copies can be carried comfortably. So there is a need for making more ebooks available. We have more number of handwritten texts available all over the world which need protection be safegaurded. By transcribing them we can increase the availability of ebooks. And also instead of striving hard to protect old texts which are hand written, they can be digitilized and stored as soft copies with ease.

We will apply the machine learning techniques in order to find the digital form of handwritten text from their scanned images. We will take the help of the readymade datasets that contain pixel values of scanned images as the inputs and we will be able to find the text in it. We can also extend this project for different languages and writing styles.

Problem Statement: To accurately predict the text from a scanned handwritten text image using Machine Learning algorithms.

For this we need to assume that all the images that contain same letters have same features and we can conclude that an image having those features contains that alphabet. However this hypothesis is ideal and may not come true always in practical.

II. DATASETS

The data plays a very important role in machine learning. The past data is used to predict the future outcome. The relevant data can be downloaded from the internet.

The data that is related to our project that is HTR consists of pixel values. The format of the data files is csv (Comma Separated Values). Each row represents an image and contains a lable in the first column and followed by 784 pixel values for 28 X 28 images.

The data related to our project will have thousands of instances. The data that we use for our project is obtained from kaggle.

Two types of data will be taken

- ✓ One for English alphabets
  A_Z Handwritten Data [1] and
- ✓ The other for digits - MNIST dataset [2]

DATA VISUALIZATION

For dataset_1 the shape will be (372450, 785) and for dataset_2 the shape will be (42000, 785).

A_Z Handwritten_Data:

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

0_9 Handwritten_Data:

The dataset will comprise of multivariate data 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 total of 42000 instances, and the total number of attributes is 784 and a label.

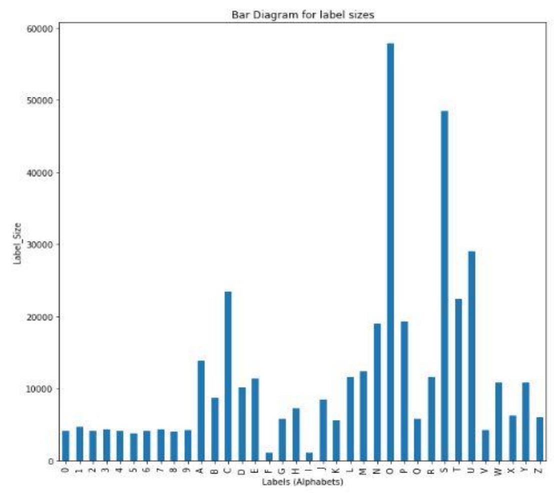
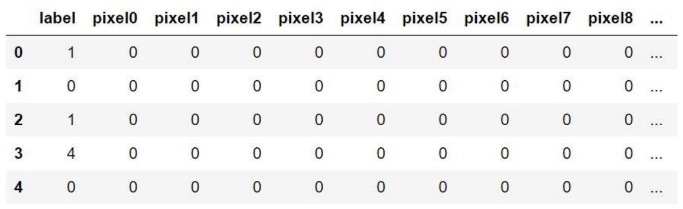


Fig 1. Bar diagram showing lable sizes of different labels



	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...
0	1	0	0	0	0	0	0	0	0	0	...
1	0	0	0	0	0	0	0	0	0	0	...
2	1	0	0	0	0	0	0	0	0	0	...
3	4	0	0	0	0	0	0	0	0	0	...
4	0	0	0	0	0	0	0	0	0	0	...

Table 1. First five instances of the dataset

III. ARCHITECTURE

The problem convert a handwritten text which is in the form of pixels into its digital for is a data driven approach. The data which is already collected can be used for extracting the features of each letter. The availability of more powerful machine learning algorithms introduces an efficient and better approach to solve this problem.
The project is divided into two modules.

A **Segmentation module** in which an image is taken as input, letters are detected, bounded, cropped, resized and then segmented and a **Training module** where prediction occurs.
The output of the segmenation module is the input of the training module.

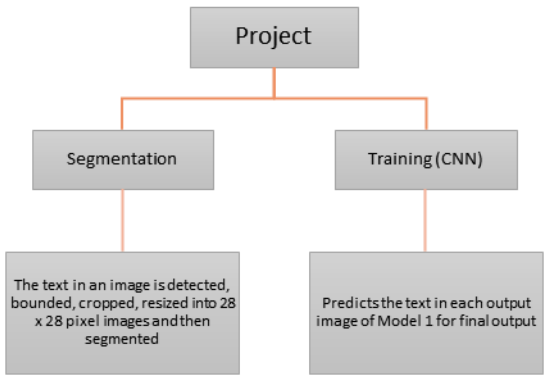

Project
Segmentation
Training (CNN)
The text in an image is detected, bounded, cropped, resized into 28 x 28 pixel images and then segmented
Predicts the text in each output image of Model 1 for final output

Fig 2. Architecture

IV. METHODOLOGY

The research methodology in this project include,

- Visualizing and understanding the data
- Choosing a suitable model
- Agreeing on a common evaluation metric
- Training and Testing the models
- Implementing the final model
- Analyzing the result

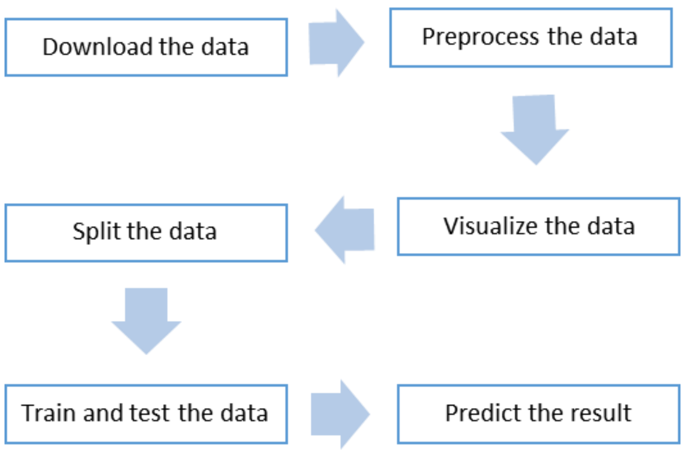


Fig 3. Flow of training module

V. EXPERIMENTAL RESULTS

A. Segmetation Module

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

1) Read the image

We have many ML libraries like Pillow, openCV etc. for performing operations on images. Here openCV is used to read and manipulate images. An image is read and then stored in multiple copies for performing different operations.
After reading the image is plotted in its shape to make sure it is read perfectly. That image contains letters that need to be images each cropped into 28 X 28 images by the end of segmentation model.

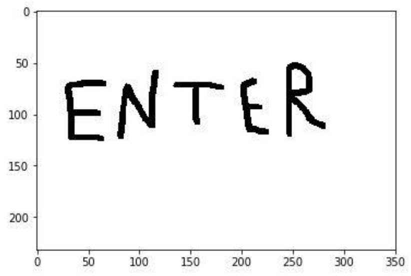


Fig 4. An image of dimensions 351 X 232 pixels

2) Detecting the letters

Object detection is a Computer vision technique that detectects certain components from in an image or a video. It makes use of Machine Learning and Deep Learning Algorithms to yield good results. Detecting the letters is same as detecting objects. We need to apply some standard filters to the input image for achieving this task.
Step 1: Convert a BGR image to Greyscale image.
An image with 3 channels is a BGR image but a Grayscale image consists of a single channel. A channel is a thid dimension of an image.

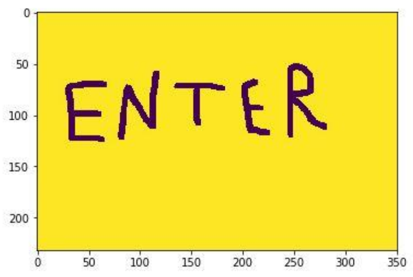


Fig 5. A Grayscale Image

Step 2: Applying Gaussian blur to the Greyscale image. This step is done to remove any noise and disturbances in the image. If the image is blurred the colour intensites can be recognised easily from the image. The blurring is technically called Gaussian Blur in the Computer Vision.

Step 3: Otsu Thresholding → A standard stepfor object detection

It is calcluation of the measure of spread for the pixel levels each side of the threshold.

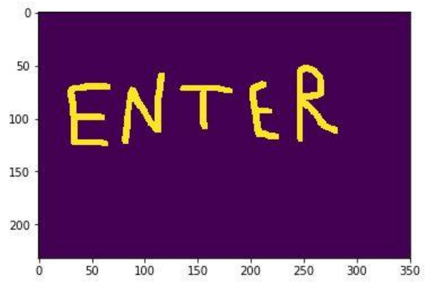


Fig 6. Image after applying Otsu thresholding

Step 4: 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.

The drawcontours need image, detected contours, colour and dimensions of the border as the parameters.

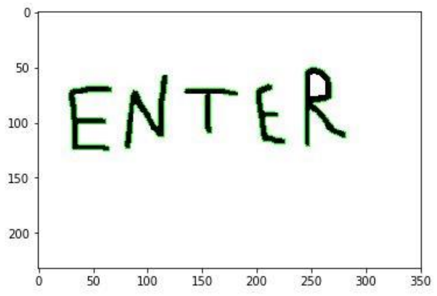


Fig 7. Image with contours

Step 5: Storing co-ordinates for rectangular bounding

Boundingrect() gets the list of x, y co-ordinates of top left point of the image, width and height allowing us to drawimages in the order of detected objects. We need to sort in the order of x co-ordinate of the top left corner to order them. All these lists are stored in a list and sorted with a base of list[i][0]th elements.

3) Bounding and Cropping

This part of code is added before cropping in order to add spaces between letters. In the list that contains bounding details, we add a space string " "if the distance between the corresponding x co-ordinates is > 50 pixel values representing they must be separated by a space.

And then Bounding and Cropping for non-spaces and storing them in a list 'img_lst'.

The elements in the list of detected objects may be the boundingrect values of a detected object from the image or a space string. The detected object is considered a letter only if its height is greater than 20 pixels. (Our assumption) Then for those which are considered letters we use boundingrect values to crop the letter from another copy of the original image stored in another variable and append each of the cropped images which are in numpy array form into a python list.

'rectangle()' draws a rectangular border around detected letters which has the image, bounding values, colour and width of the border.

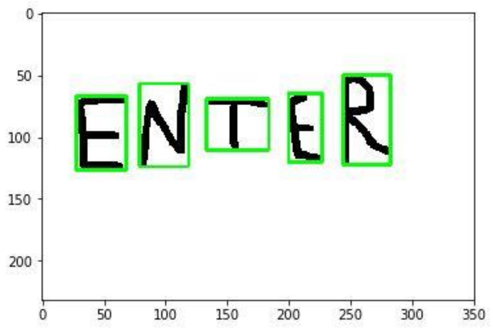


Fig 8. Image with rectangular bounds around

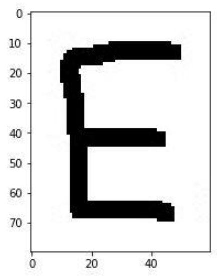


Fig 9. A cropped image of letter E is one of the results of cropping

4) Resizing

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

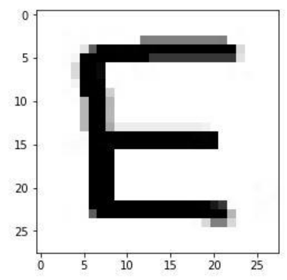
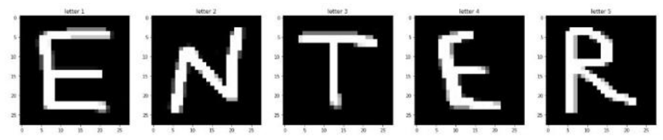


Fig 10. A resized image

Each resized images must be converted to Grayscale to match requirements of training model. And then sent as input to the training model.

The subplot of the output of segmentation module looks like



B. *Training Module*

A model is trained by using past data and Machine learning Algorithms. It learns from the past data by feature extraction and patterns. In this project Convolutional Neural Networks are used. We split the data into training and testing in the ratio of 80:20.

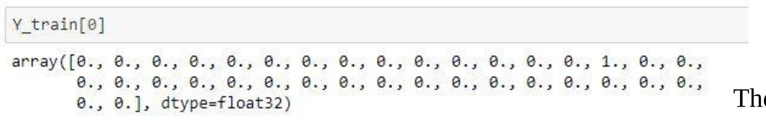
The dataframe with attributes, the dataframe with only label column, train_size or test_size and shuffeled are important parametere of trin_test_split. Shuffeled will be 'True' by default which shuffeles the data before spliting. This method need not always return same output.

Scikit-learn library is used to change the form of data. We need to convert the attribute values to the float datatype and their labels into the categorial form to train them.

The attribute values are converted to floating point numbers ranging from 0 to 1 which earlier were 0 to 255. A pixel value of 0 will be 0.0000, a pixel value of 1 will be 1.0000, and a pixel value of 128 will be 0.5000.

Categirial form results in a list of size equals number of all possible labels in which an instance with lable value i will have i = 1 and other elements 0. Even these values must be in float.

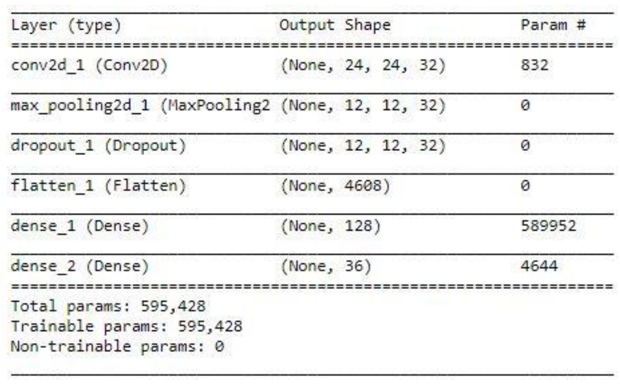
The categorial form will look like this.



```
Y_train[0]

array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0.], dtype=float32)
```

Now we can proceed to train our data by using standard Machine Learning Algorithms.
A sequence of hidden layers are created with some nodes in each of them. The first hidden layer is 2Dimensional Convolution layer with kernal 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).
Now we compile the model by 'categorial_crossentropy' loss function, 'adam' optimizer with a metric of 'accuracy'.



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 24, 24, 32)        832
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 32)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4608)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               589952
_________________________________________________________________
dense_2 (Dense)              (None, 36)                4644
=================================================================
Total params: 595,428
Trainable params: 595,428
Non-trainable params: 0
_________________________________________________________________
```

Fig 11. Summary of the training model

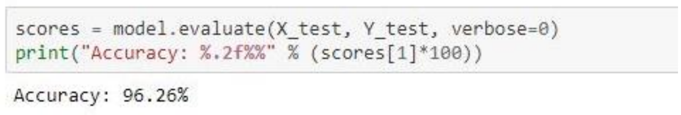
The fitting of a model is shown below. It is passed with 1 iteration. 200 batch size and the dataset contains 331560 instances.

```
model.fit ( X_train, Y_train, validation_data = (X_test, Y_test),
epochs=1, batch_size=200, verbose=2)
```

```
Instructions for updating:
Use tf.cast instead.
Train on 331560 samples, validate in 82890 samples
Epoch 1/1
-     350s – loss: 0.3029 – acc: 0.9184 – val_loss: 0.1381 – val_acc: 0.962
```

VI. RESULTS

The above model will give us the train accuracy of 0.9184 and the test accuracy of 0.9626.



```
scores = model.evaluate(X_test, Y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))

Accuracy: 96.26%
```

Model	Loss	Accuracy
Neural Network	0.1381	0.9626

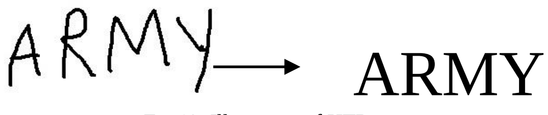


Fig 12. Illustration if HTR

VII. CHALLENGES INVOLVED

The challenges we have faced while modelling handwritten text recogniser are
(i) Letters like 'i'and 'j'which have break in them cannot be detected as a single letter.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 04, April-2020

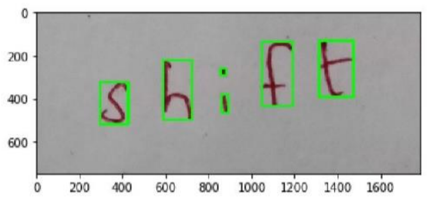


Figure An image with rectangular border around detected contours

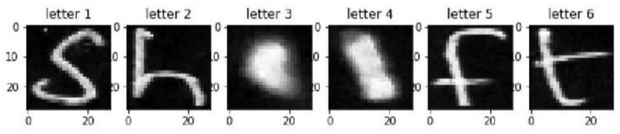


Figure A subplot for cropped images of above image

(ii) If two letters touch each other (like in cursive writing), they are recognized as a single letter.

VIII. CONCLUSION

Convolutional Neural Network learns from the real time data and simplifies model by reducing the number of parameters and hence gives considerable accuracy.
Future Enhancements
We can increase the accuracy:
→ By taking huge datasets
→ By adopting much suitable algorithms
→ We can compile the model at more number of epochs.
→ Hyper-parameter Tuning (There are a lot of parameters that we can play with).
→ Use of deeper architectures

This application can be taken to next level by
→ Extending its scope to different writing styles
→ Extending its scope to different writing styles

ACKNOWLEDGEMENT

We would like to acknowledge the help of ‘JNTUA College of Engineering, Pulivendula’ for the kind support provided and our faculty and friends for the helpful discussions. We also would like to thank ‘kaggle’ who provided datasets.

REFERENCES

[1] https://www.kaggle.com/sachinpatel21/az-handwritten-alphabets-in-csv-format
[2] https://www.kaggle.com/c/digit-recognizer/data
[3] *CHARACTER RECOGNITION IN NATURAL IMAGES* By Te´ofilo E. de Campos, Bodla Rakesh Babu, Manik Varma https://www.researchgate.net/publication/221416071_Character_Recognition_in_Natural_Images/link/5dd6e92892851c1feda56fc1/download
[4] *Text detection and recognition in raw image dataset and seven segment digital energy meter display* By Karthick Kanagarathinam, Kavaskar Sekar https://reader.elsevier.com/reader/sd/pii/S235248471930174X?token=FFC0111CC7487898FEFE8637DDA6CE1692B76C48DBB26C375D1CD755667BBC2109D8C5287A2205169F20461A43BDD304
[5] *Scene Text Detection and Recognition: The Deep Learning Era* By Shangbang Long, Xin He, Cong Yao https://arxiv.org/pdf/1811.04256.pdf
[6] *Automatic Text Detection and Classification in Natural Images* By C.P. Chaithanya, N. Manohar, Ajay Bazil Issac https://www.ijrte.org/wp-content/uploads/papers/v7i5s3/E11330275S19.pdf
[7] *An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition* By Baoguang Shi, Xiang Bai and Cong Yao https://arxiv.org/abs/1507.05717
[8] *EMNIST: an extension of MNIST to handwritten letters* By Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andr´e van Schaik https://arxiv.org/pdf/1702.05373.pdf
[9] *Handwritten Text Recognition for Historical Documents* By Veronica Romero, Nicholas Serrano, Alejandro H. Toselli, Joan Andreu Sanchez and Enrique Vidal https://www.aclweb.org/anthology/W11-4114.pdf
[10] *Arabic Cursive Text Recognition from Natural Scene Images* By Saad Bin Ahmed, Saeeda Naz, Muhammad Imran Razzaq and Rybiyah Yusof https://www.mdpi.com/2076-3417/9/2/236/pdf