1. Introduction

1.1 General

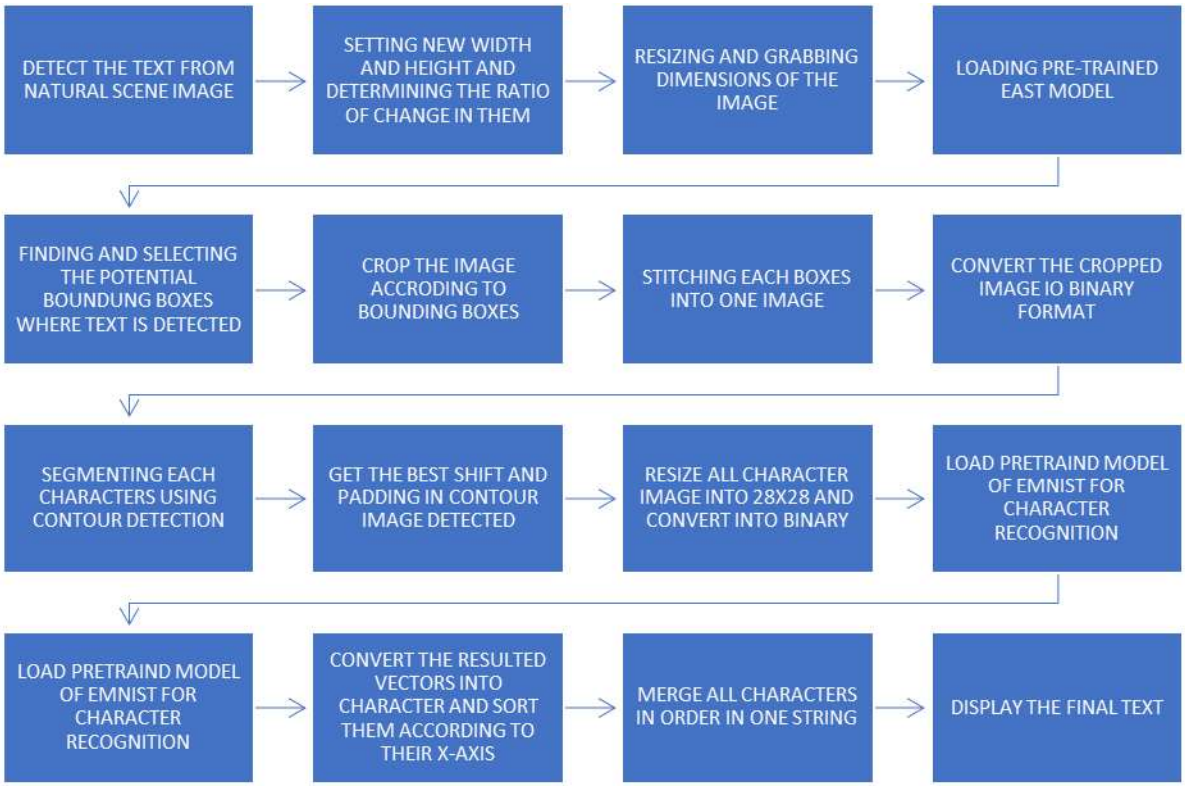
Natural Scene Text Detection and Recognition is a part of computer vision applications. This comes under the branch of Machine Learning and Deep Learning where neural networks are used for detection and recognition applications. Neural networks are inspired by the working of human brains. There are many kinds of neural networks of which CNNs are mainly used in imaging applications. Image processing is a field of study where the image is converted into digital format for performing various applications.

1.2 Problems addressed

This project can be applied to license plate recognition for safety purposes and for identifying the information about the driver as well as vehicle. Visually challenged peoples can also benefit from it as it can be applied to describe information by detecting the surrounding and sending the messages to the required person through speech data.

1.3 Expected Outcomes

The text is detected in images and bounding boxes are drawn on the region of interest (ROI). Then the image is cropped and converted to binary format on which the segmenting of characters is done which is further fed to the model where the recognition of the character is being done. Then the output is printed in text format.



2. Overall Descriptions

2.1 Tools Used

2.1.1 Python

It is the most widely used programming language in machine learning and deep learning applications.

2.1.2 OpenCV

It is known as open computer vision library used for the image processing applications.

2.1.3 NumPy

It is python library used for numerical analysis of data as well as used in vector and matrix operations.

2.1.4 Imutils

It is a library used for non-max suppression for detecting the most reliable bounding boxes of text in image.

2.1.5 Argparse

It is used for argument parsing the path of images as well as the path of pre-trained model file.

2.1.6 Matplotlib

It is used for plotting (displaying) the characters which are segmented from the text bounded image.

2.1.7 Keras

It is high level api used for creating neural networks models and training as well as testing them on specified metrics.

2.2 EAST Model

2.2.1 Brief Introduction

It refers to **E**fficient and **A**ccurate **S**cene **T**ext Detector. The pipeline is capable to detect text in images having 720p resolutions and also for video that run on 13 FPS. Several factors are taken in account in designing the network that are varying word sizes, regions and many more.
It was combinedly used with many models like PVANET, VGG-16,etc.
Model gave the best results with PVANET2x.

Algorithm	Recall	Precision	F-score
Ours + PVANET2x RBOX MS*	**0.7833**	0.8327	**0.8072**
Ours + PVANET2x RBOX	0.7347	**0.8357**	**0.7820**
Ours + PVANET2x QUAD	0.7419	0.8018	0.7707
Ours + VGG16 RBOX	0.7275	0.8046	0.7641
Ours + PVANET RBOX	0.7135	0.8063	0.7571
Ours + PVANET QUAD	0.6856	0.8119	0.7434
Ours + VGG16 QUAD	0.6895	0.7987	0.7401

The structure of FCN used in EAST Model is shown as below.

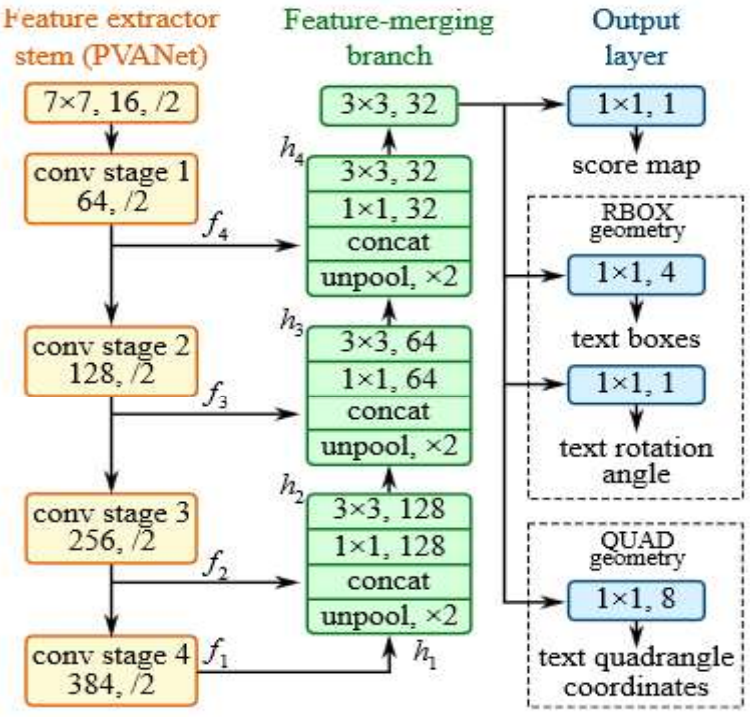


Figure 3. Structure of our text detection FCN.

2.2.2 Limitations

Longer text regions are not handled properly.
Also, sometimes the areas looking similar like text but not actually text area also detected due to the structure.
It gives improper predictions for vertical text instances as they take only small regions of text.

2.2.3 Usage

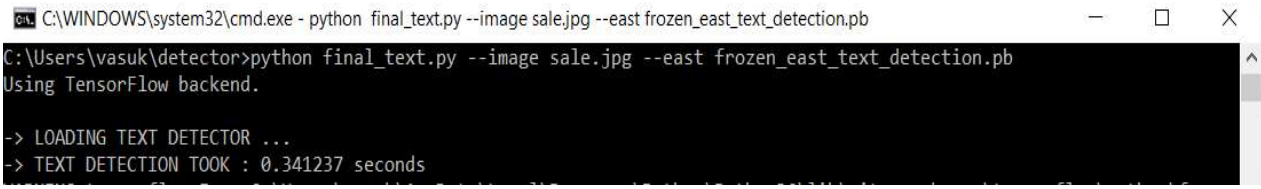
We only used the last two layers of this model which gives the scores or probability of the bounding boxes and other gives the geometry or dimensions of the bounding boxes.

3. Implementation

3.1 Taking the images and pre-trained model

Here argparse is used where the input image and pre-trained models file is taken in the python command on the command prompt.

Model file: frozen_east_text_detection.pb
Image: sale.jpg



3.2 Reading Image and Taking the ratios

The image is read using OpenCV library. The general width and height to be used is taken 320*320. But the images can be of any size so the ratio is stored and further used for resizing the image into the defined format for inputting it in the model.

3.3 Defining Layer, Loading Model and Creating Blobs

The important 2 layers in the model which are of scores and geometries are defined. The model is loaded OpenCV. Blobbing of image is done to convert RGB to BRG format which is used by OpenCV.

3.4 Detecting Boxes and Retrieving Scores and Geometries

The image is fed to the model and scores and dimensions are retrieved and stored in the defined variables. Non-Max suppression is applied on the retrieved boxes to find the most probable bounding boxes of the same instances.

3.5 Drawing Boxes

The dimensions are again resized by the ratio because the model reduces the size of the image due to convolutions. Then the appropriate dimensions are used in drawing the bounding boxes. The image looks as follows.

3.6 Resizing, Stitching and Binary Conversion

The multiple images cropped in the bounding boxes regions are taken max width and height between them is found, max width is then doubled for all instances and stitched together into one single image. Further using OpenCV binary conversion is done by thresholding and OTSU method.

3.7 Contours Detection and Segmentation

This image is then applied on the OpenCV findContours function which gives the bounding boxes (contours) on each character and thus segmenting is performed by cropping each contour and saving their dimensions in a list for further processing.

3.7 Order the contours:

Contours detected in random manner is ordered according to their X – axis in the whole image. And stored in the dictionary containing their X-axis as key.

3.8 Character Recognition

Pretrained model on EMNIST is loaded and cropped images are fed into the model to recognize the character & then output vector will given by the model. Maximum value index of vector will give the class of character by using that character will be obtained. And character recognized will be concatenated in the final text string.

```
C:\Users\vasuk\detector>python final_text.py --image sale.jpg --east frozen_east_text_detection.pb
Using TensorFlow backend.

-> LOADING TEXT DETECTOR ...
-> TEXT DETECTION TOOK : 0.481085 seconds
WARNING:tensorflow:From C:\Users\vasuk\AppData\Local\Programs\Python\Python36\lib\site-packages\tensorflow\python\framew
ork\op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a f
uture version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From C:\Users\vasuk\AppData\Local\Programs\Python\Python36\lib\site-packages\keras\backend\tensorflow
_backend.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed i
n a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
2019-04-26 13:31:19.363172: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that thi
s TensorFlow binary was not compiled to use: AVX2
WARNING:tensorflow:From C:\Users\vasuk\AppData\Local\Programs\Python\Python36\lib\site-packages\tensorflow\python\ops\ma
th_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Model successfully loaded
[21, 93, 171, 322, 438, 580, 634, 711, 777]
Detected text: N0WONSARE
```

3.9 Model for character recognition using EMNIST

3.9.1 EMNIST balanced dataset:

It is training & testing dataset contains digit & character images for training and testing the model for character recognition. It basically contains 0-9 digits, A-Z alphabets, and small characters that differs from their respective capital letters like 'a' ,'b' , 'd', 'e', 'f', 'g', 'h', 'n', 'q', 'r' are included in different classes. Training dataset contains 112799, & testing dataset contains 18799 images

3.9.2 Pre-processing dataset:

In pre-processing we resize the image into appropriate shape 28x28x1 & convert it into the binary format.

3.9.3 Model:

In model architecture we have used Convolution layer & Max Pooling for extracting the features from the image & Dropout for smoothing the learning and at the end for output we have used softmax layers for output of 47 classes output with maximum value will be the detected character.

3.9.4 Hyper parameters used in architecture:

nb_filters = 32 # number of convolutional filters to use

pool_size = (2, 2) # size of pooling area for max pooling

kernel_size = (3, 3) # convolution kernel size

3.9.5 Training the model:

For training the model we have used “Adadelta” optimizer & loss function categorical cross entropy. In training following hyper parameter are used.
Batch size = 256 Number of epochs = 10.

4. Conclusions

4.1 Detection Model

The EAST Model is sometimes detecting unknown regions (text like areas but not text actually). Otherwise the text areas are detected perfectly. There is some distortion in boxes when the texts are tilted.

4.2 Recognizing Model

EMNIST is having some problems in differentiating between O's and 0's (zeroes) due to similar shape.

On MNIST (Only digit dataset)

Tested accuracy: 99.27%

On EMNIST balanced dataset:

TRAIN ACCURACY: 87.05%

TEST ACCURACY: 87.21%