**README**

**(Un)structured**

In simple words, the problem we're targeting is as follows - "You have an image as an input which contains text at certain locations in the image. You need to detect the text and generate the text present in the input as the output. This involves segmenting the given image to focus only on the part of the image which contains text. Then you need to implement text recognition to generate the output.

This project was done as part of **IIT Bombay Seasons Of Code (SoC).** Majority of the project takes inspiration from the paper – ‘text recognition in the wild’ and implementation is done entirely in python.

***Link to detection model-***

[***https://www.kaggle.com/aryankolapkar/text-detection***](https://www.kaggle.com/aryankolapkar/text-detection)

***Link to recognition model-*** <https://colab.research.google.com/drive/1uQfpg94M2mBf8yg05JUktulRKYP_ulsC?usp=sharing>

**LEARNING**

Concepts used in this project were learned mostly from Andrew NG’s Deep Learning Course on Coursera.

**DATASETS USED-**

* ***SVT*** -used for overall testing (**download link** - <http://vision.ucsd.edu/~kai/svt/>)
* ***IIIT 5K*** -this dataset was used to train a recognition model (**download link**- <https://cvit.iiit.ac.in/research/projects/cvit-projects/the-iiit-5k-word-dataset>)
* ***MJSYNTH 90K*** -recommended for training recognition model, but its size is too large and will take a lot of time to train. (**download link** - <https://www.robots.ox.ac.uk/~vgg/data/text/>)

**DATA PREPROCESSING**

1. So after downloading the SVT dataset, you will have folders containing train and test images respectively. The text annotations of these images will be present in a .xml file. These annotations contain information of the words present in each image and their respective ground truth bounding box coordinates (x & y coordinates of the top left corner of the rectangle and the height and width of the rectangle).
2. We will be storing data in the annotations.xml file as Pandas DataFrame as it makes operations easier. To do so we will have to convert the xml file to CSV format and then read it using Pandas. Open the annotations file using Microsoft Excel in the csv format then read this csv file in python using the pd.read\_csv(‘<file path>’) function available in the Pandas library to form a DataFrame. This DataFrame should contain information about the word bounding box and its corresponding image name. convert the image names in the DataFrame to their respective image paths by performing a simple string addition operation over a loop so that it can be more usable.
3. Now you need to extract the image pixel arrays by storing them into a list variable X, by reading each image through matplotlib’s imread(path) function. You can now resize these images to any required size by calling the cv2.resize() function from the cv2 library. I have resized the images to (416,416). NOTE- it is better if you resize the actual images associated with the image paths directly as it will be useful in the future. Right now we have only resized the image pixel arrays associated with them.

**TEXT DETECTION**

Before recognition we will first need to detect bounding box coordinates of the words present in the image. There are many object detectors that can be trained to detect custom objects. We will be using the Retina net based object detector. The steps to use this are well documented in the links – (<https://github.com/fizyr/keras-retinanet>)

We have used a relatively basic detector as our data preprocessing format matched the retina-net format. There are better text detectors like EAST and textboxes++ which you can also use.

**TEXT RECOGNITION**

The retina net based detector gives many bounding box proposals along with respective scores. To filter these boxes you can either perform simple thresholding or you can train a text classifier. In the paper ‘text recognition in the wild ’ they have performed series of bounding box regression and coordinate modifications so as to obtain a more accurate proposal, it is worthwhile to look at that method if your proposals are having a relatively low IOU with the ground truth.

For text recognition, we have used an end to end method by transforming this into a simple classification problem like MNIST.

We make a CNN model having an end layer SoftMax output consisting of dictionary words. You can train this on a larger Synth 90K dataset if you have the computational resources. We have used a smaller IIIT 5K dataset since our purpose was just to check if this method works.

To train a text recognition model we first converted the annotation file from the .mat format to a pandas DataFrame and then resized and converted all images to grayscale and appended them in a variable called X\_train.

We created an array Y\_train consisting of the words the respective images have. The order of both these arrays were same, therefore each image was associated with its word. We then converted the words which were in string format and assigned them unique integer values so as to quantify them. We further converted these integers into their One Hot Encodings (using keras function to\_catagorical, see keras documentation) so that we can use them in our SoftMax layer.

**OPTIMIZING THE RECOGNITION MODEL**

We found that you need a relatively deep CNN model so as to capture all the features accurately. While training we faced an issue where our loss and accuracy weren’t changing after some epochs. This may happen due to the zero gradient problem where your training gets stuck as it approaches a minima. To solve this it is advised to try out different optimizers such as Adam, SGD, RMSprop etc.. and increase or decrease the learning rate.

We achieved 100 % accuracy on the train set of IIIT 5K. But since this is a small dataset which mostly contains one or two images associated with a word, our model doesn’t generalize well.

**OVERALL RESULT**

I have not tested the detector and recognition model together on the SVT dataset yet, there’s a lot of work remaining to be done on improving the recognition model. I will post the results and keep updating the project as my work progresses.