Text Detection for Ayata

Dr. Srinath Madasu

There are quite a few ways for text detection:

- Traditional way of using OpenCV
- Contemporary way of using Deep Learning models
- Based on more complex deep learning models for end-end hand writing recognition such as apache-mxnet, which involves Image Processing, Paragraph segmentation, Line/Word segmentation, Word to line Processing, denoising etc. It involves lot of computational power and time for training.

First Two Ways for Text Detection are explored here due to time constraints (reference -

https://www.analyticsvidhya.com/blog/2020/05/build-your-own-ocr-google-tesseract-opencv/).

Text Detection using OpenCV

Text detection using **OpenCV** is the classic way of doing things. Various manipulations like image resizing, blurring, thresholding, morphological operations, etc. can be applied to clean the image. Here we have Grayscale, blurred and thresholded images, in that order.Once you have done that, you can use OpenCV contours detection to detect contours to extract chunks of data. Finally, you can apply text recognition on the contours that you got to predict the text.

Tesseract assumes that the input text image is fairly clean. Unfortunately, many input images will contain a plethora of objects and not just a clean preprocessed text. Therefore, it becomes imperative to have a good text detection system that can detect text which can then be easily extracted. It performs poorly when the image has a lot of noise or when the font of the language is one on which Tesseract OCR is not trained. Other conditions like brightness or skewness of text will also affect the performance of Tesseract. Nevertheless, it is a good starting point for text recognition with low efforts and high outputs. The results in the image above were achieved with minimum preprocessing and contour detection followed by text recognition using Pytesseract. Obviously, the contours did not detect the text every time.

But, still, doing text detection with OpenCV is a tedious task requiring a lot of playing around with the parameters. Also, it does not do well in terms of generalization. A better way of doing this is by using the EAST text detection model.

Contemporary Deep Learning Model – EAST

EAST, or Efficient and Accurate Scene Text Detector, is a deep learning model for detecting text from natural scene images. It is pretty fast and accurate as it is able to detect 720p images at 13.2fps with an F-score of 0.7820.

The model consists of a Fully Convolutional Network and a Non-maximum suppression stage to predict a word or text lines. The model, however, does not include some intermediary steps like candidate proposal, text region formation, and word partition that were involved in other previous models, which allows for an optimized model.

You can have a look at the image below provided by the authors in their paper comparing the EAST model with other previous models. EAST has a U-shape network. The first part of the network consists of convolutional layers trained on the ImageNet dataset. The next part is the feature merging branch which concatenates the current feature map with the unpooled feature map from the previous stage.

This is followed by convolutional layers to reduce computation and produce output feature maps. Finally, using a convolutional layer, the output is a score map showing the presence of text and a geometry map which is either a rotated box or a quadrangle that covers the text. This can be visually understood from the image of the architecture that was included in the research paper:

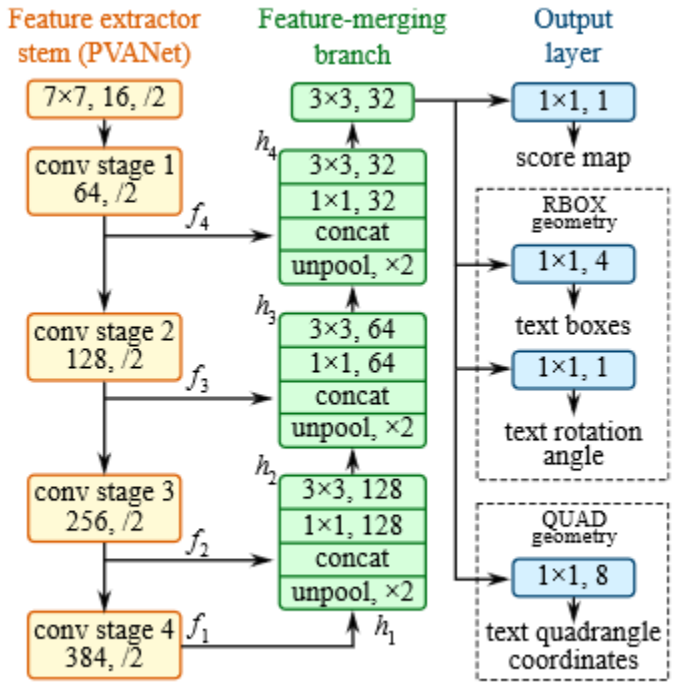

Figure 3. Structure of our text detection FCN.

I highly suggest you go through the paper yourself to get a good understanding of the EAST model.

OpenCV has included the EAST text detector model in version 3.4 onwards. This makes it super convenient to implement your own text detector. The resulting localized text boxes can be passed through Tesseract OCR to extract the text and you will have a complete end-to-end model for OCR.

Running the executable

Install Pytessearct, Tesseract, Pytextextractor, and tkinter etc. necessary libraries

Run the python code guitextextraction.py

GUI will open

Select the method –

- **Tesseract method uses OpenCV, Tesseract**
- **Textextractor method uses Pytextractor (based on EAST)**

Click on “Open image” to select your image

Wait for few minutes to get the results on the screen

Click on Exit to quit the application