Vehicle Data Analysis (Using CNN and Image Processing)

Aravind.S.Pallavoor

(USN : 01FB16EEE022)

PES University (RR Campus, Bangalore)

23-08-2019

Objectives

- The aim of this project consists of 3 parts :
  - Vehicle Detection : Automated way of detecting the presence of a car in a digital image.
  - License Plate Detection : Having identified the vehicle in the above process, the next step is to identify the existence of a license plate through Image Processing.
  - License Plate Text Extraction : From the above identified license plate, extract the text to determine the license number through Optical Character Recognition (OCR).

Implementation

- This project consists of 3 modules and has been implemented in Python 3.7 as follows :
  1. **Vehicle Detection** : To create a Convolution Neural Network model using Keras (Tensorflow backend) on a training and testing image data-set and implement principles of image classification such as Image Segmentation , Max-Pooling , Flattening, and Fully Connected layers.
  2. **License Plate Detection** : Design a program using Open-CV methods , to implement image processing techniques to detect and draw a shape contour bounding box around the license plate of any car image.
  3. **License Plate Extraction** : Design a program using Open-CV and PyTesseract to implement Optical Character Recognition to extract string characters from the obtained license plate image using cropping techniques and text extraction.

Vehicle Detection

- Chronology and implementation of the program :
  - A data-set of 200 random car images (training data) and 50 new car images (testing data) were obtained from Kaggle data-sets.
  - A Python program was created on Google Colab to implement the CNN model since it supports a free version of GPU. Keras libraries were loaded. The model specifications (model type, no. of layers, no. of features, etc.) were coded as inputs.
  - Intermediate CNN sequential model steps such as Flatten , Dense , Dropout and types of activations were mentioned.
  - The training and testing paths were initialised and linked to the model using ImageDataGenerator and Model.fit_generator functions.
  - The model was run over 10 epochs and the detection accuracy was displayed for each run.

License Plate Detection

- Chronology of the program
  - Libraries such as OpenCV , Imutils and Numpy were loaded on Python 3.7.
  - Images were sent as input values to a variable upon which Image Processing features were implemented.
  - The image was resized using Imutils and filters for conversion to grayscale , noise removal were implemented. A canny edge detection function was implemented as well.
  - Functions such as cv2.findContours and imutils.grab_Contours were implemented to obtain contours using a for loop where the contour specifications were defined to be of a rectangular form. The contour area was returned in the form of a list and displayed with a bounding box along the perimeter of the license plate area.

License Plate Text Extraction (Using OCR)

- Chronology of the program
  - The license plate that was detected from the previous program was cropped out as a new image and saved as a new image file.
  - Modules such as cv2 and PyTesseract were used in the program.
  - A region of interest called (ROI) was located and formed at regions potential textual content existed. This region was further cropped out and saved separately. This process was repeated in a For loop and the region with maximum text length was saved and returned as a variable. This variable was later sent as an input to the OCR function which is implemented using PyTessearct.

Working process of OCR

- OCR software often "pre-processes" images to improve the chances of successful recognition. The subsequent techniques include:

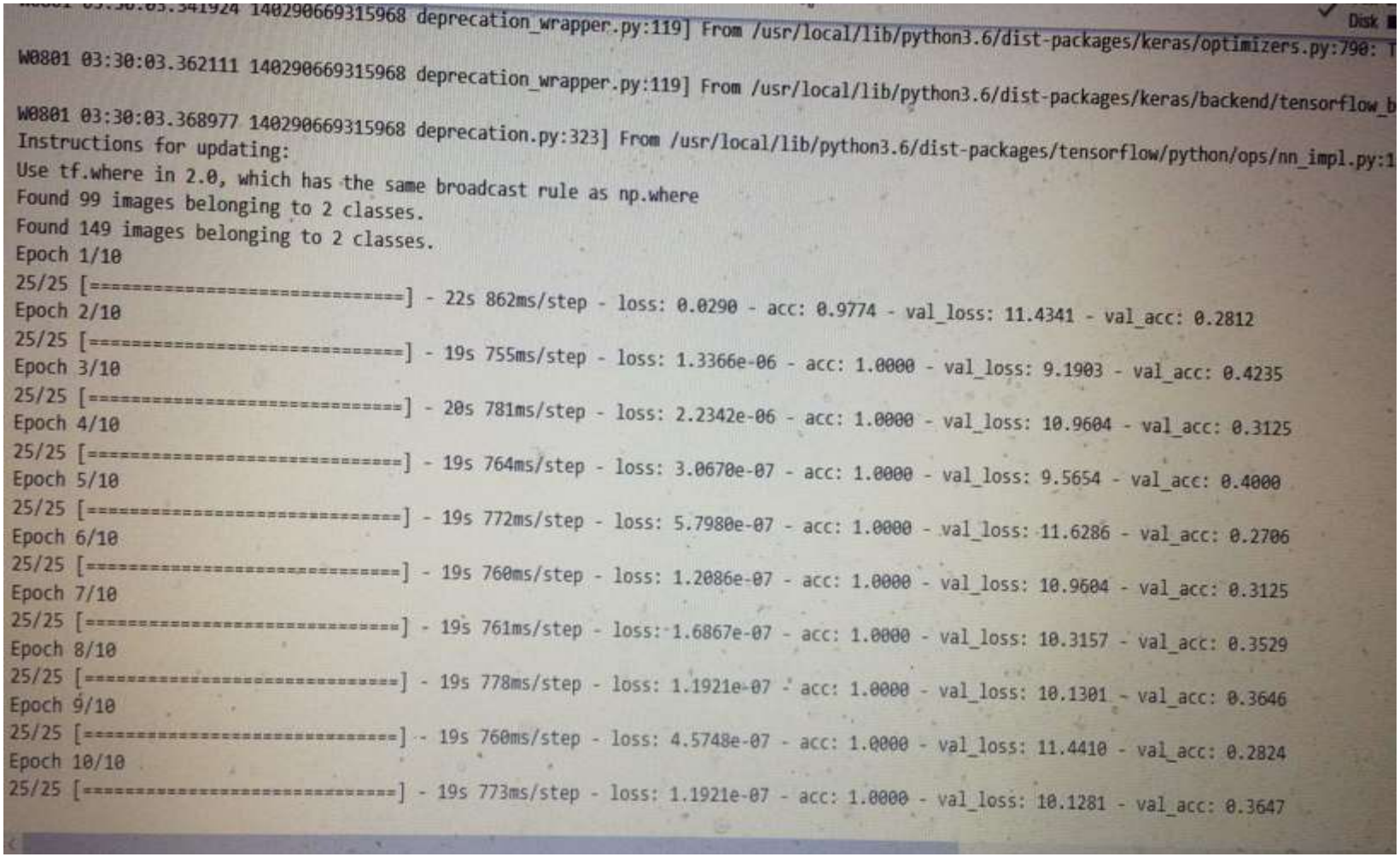
  - De-skew
  - Despeckle
  - Binarisation – Convert an image from color or grayscale to black-and-white
  - Layout analysis or "zoning“
  - Line and word detection – Establishes baseline for word and character shapes, separates words if necessary.
  - Script recognition
  - Character isolation or "segmentation“
  - Normalize aspect ratio and scale
  - Finally : Feature extraction and Matrix matching are done by comparing an image to a stored glyph on a pixel-by-pixel basis

Results/Outcomes

- Car identification : The testing data-set was fed to the model as input and having run over 10 epochs , the model returned an accuracy of 97.44 %.
- License plate detection and extraction details of execution for a sample data-set are given as follows :

Image name	Plate detection	Plate text extraction	Canny edge	Format	Fault observation
Car_image_2	yes	no	yes	.jpg	license plate blurred
Car_image_3	no	no	yes	.png	license plate blurred
Car_image_4	yes	yes	yes	.png	
Car_image_5	yes	yes	yes	.png	
Car_image_6	yes	yes	yes	.jpg	
Car_image_7	yes	no	yes	.jpg	license plate blurred
Car_image_8	yes	no	yes	.png	unable to crop proper region
Car_image_9	no	np	yes	.jpg	plate contour not well defined
Car_image_10	yes	yes	yes	.jpg	
Car_image_11	yes	no	yes	.png	unable to crop proper region
Car_image_12	yes	yes	yes	.png	
Car_image_13	no	no	yes	.jpg	plate contour not well defined
Car_image_14	yes	yes	yes	.jpg	

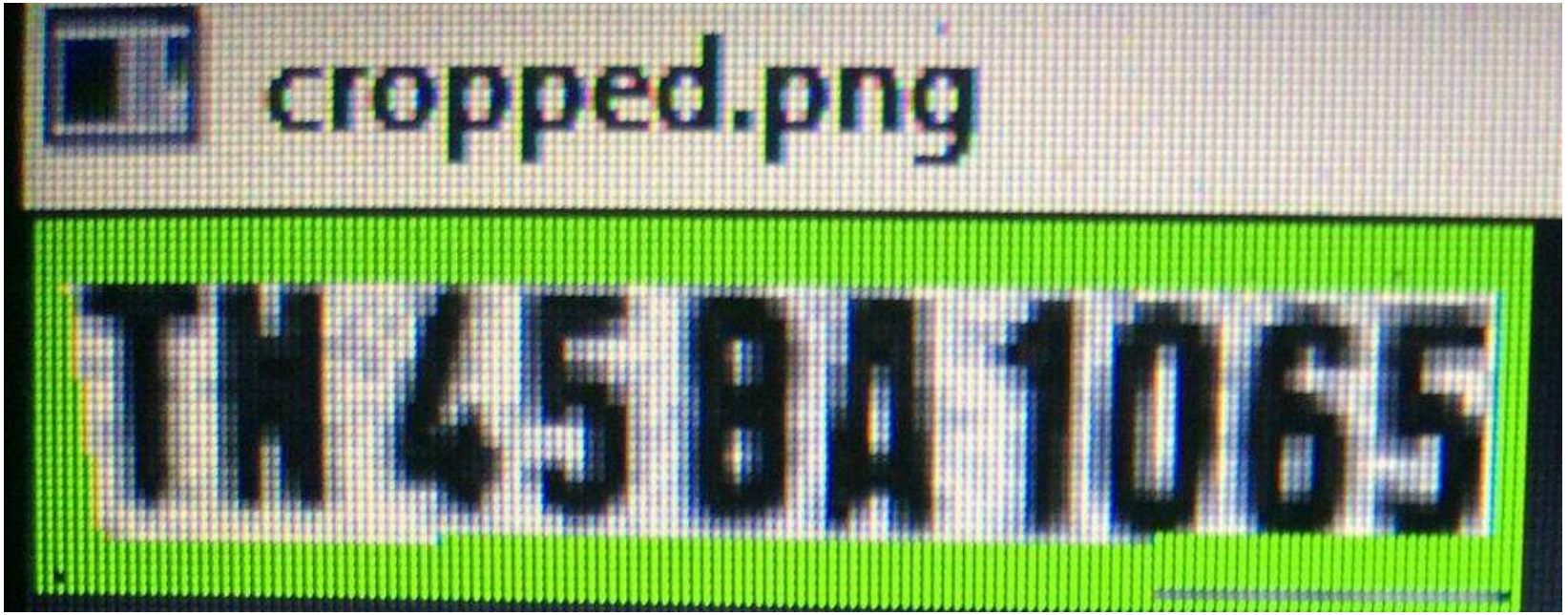
CNN model output



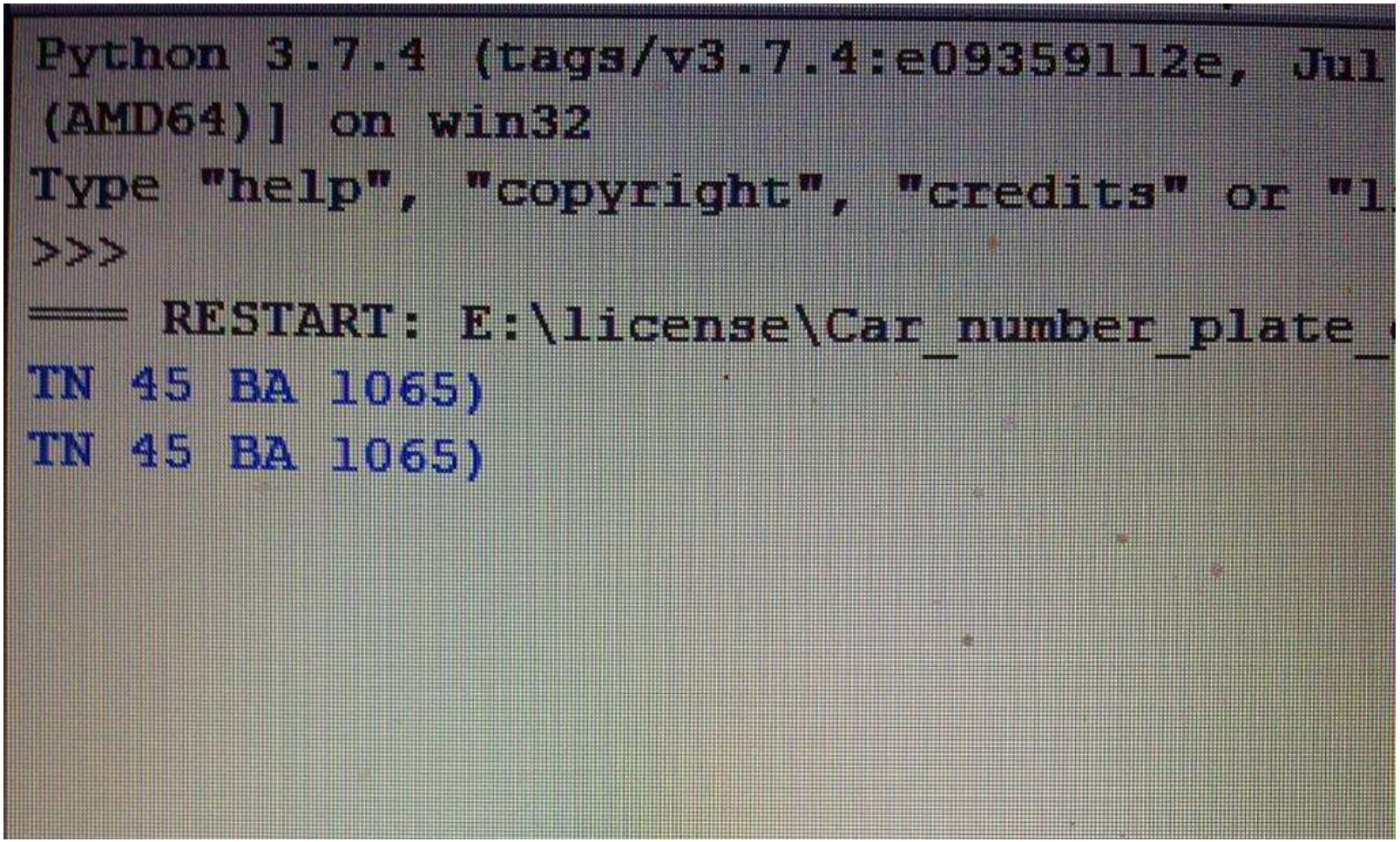
Canny edge detection

License plate detection

Cropped region



OCR Text Character Extraction



Next steps :

- Integration of the as is software for image recognition with Raspberry Pi using the help of a camera module.
- Integration of software for image recognition with NVIDIA JetSon hardware components and leveraging the GPU.
- Fine tuning of the current model for different cases of input license plate image orientations from different positions.
- To run all the 3 modules with a local GPU in Python 3.7 and integrate them under one Python file using a orderly procedure.

The End