VEHICLE NUMBER PLATE DETECTION

Identify the license place in the image and do an OCR to extract the characters from the detected license plate

DT Number: Dt20195528054
Contestant Name: MEHUL GUPTA
College Name: IIITDM Jabalpur

Background

Vehicle Number Plate Detection aims at the detection of the License Plate present on a vehicle and then extracting the contents of that License Plate.

A vehicle's license plate is commonly known as 'a number plate'. It is a metal plate which is attached to a vehicle and has the **official registration number** of a vehicle embossed on it. Number plates are placed at the **front and back of the vehicle** and help anyone to identify a vehicle.

Why registration required?

Motor vehicle registration is the registration of a motor vehicle with a government authority, either compulsory or otherwise. The purpose of motor vehicle registration is to establish a link between a vehicle and an owner or user of the vehicle. This link might be used for taxation or crime detection purposes

Contents of a Number Plates

The registration identifier is a numeric or alphanumeric code that uniquely identifies the vehicle within the issuing authority's database.

These number plates can be of different color & have different font and font size depending upon the country and other rules.

Some places where this solution can be used:

· Analysis of city traffic during peak periods

· Automation of weigh-in-motion systems

· Enhanced vehicle theft prevention

· Effective law enforcement

· Effective enforcement of traffic rules

· Flexible and automatic vehicle entry to and exit from the car park

· Management information about car park usage

· Improved security for both car park operators and car park users

· Improved traffic flow during peak periods

· Vehicle recognition through date and time stamping as well as exact location

· Inventory management · Comprehensive database of traffic movement

At last but not least state border control is one of the most important applications of automatic license plate recognition.

· Automation and simplicity of airport and harbour logistics

· Security monitoring of roads, checkpoints, etc.

· Vehicle surveillance · Prevention of non-payment at gas stations, drive-in restaurant

Your understanding

The stated problem can be divided into two sub-problems:

1)Number Plate Detection

This problem can be tackled using Object Detection approach where we need to train our model using the car/other vehicle images with number plates.

2)Extracting text from the detected Number Plate

This problem can be solved using OCR(Optical Character Recognition) which can be helpful in extracting alphanumeric characters from cropped Number Plate images.

It can be assumed that this solution has to be embedded with a web application as well where once the camera captures the image of the vehicle, the backend would call the solution and outputting the contents to the user.

Scope

The elements that has been considered in this solution:

1)Number Plate Detection:

- Both frontal and back plates are considered in the solution
- Due to lack of unavailability of annotated Indian car images with number plates, a mix of Tunisian and Indian Cars is used for training purpose
- Rotated images are considered
- Cases where cars are in some angle are considered
- Image of any dimension has been considered
- Considering only one detected object for feeding OCR(like if many plates detected,taking only one) per image.

2)Text Extraction from Detected Number Plate:

- Output of Number Plate Detection solution only(no rotated images though)
- Only cropped number plate fed to OCR and not the entire image
- Text with size 12pts are the best extracted

Out of Scope:

- No random images(cat, dog,etc.) for training purpose of Detection model
- No other vehicle considered other than cars in training (unavailability of dataset)
- Rotated images(in case of OCR)
- As Tunisian data has been used all through the problem, OCR is able to understand digits but not characters(Arabic language)
- Bad quality images(blur,incomplete,etc.)
- No video data

Assumptions

- GPU availability for training
- Though trained & tested over only Car images, It should be able to detect NumberPlates in any sort of vehicle(Bike,Truck,etc)
- Image quality is upto mark
- No limitation over Memory and Latency
- Internet Availability(For OCR purpose)
- Python 3+
- Microsoft Vision API being used is accessible(I myself using free trial,which might become unavailable during the evaluation(7 day window))
- Videos not considered

Solution Approach:

HIGH LEVEL APPROACH

As mentioned above, The problem has been divided into 2 parts:

- NumberPlate Detection
- OCR over extracted Number Plates extracted

The solutions has been approached in the following ways:

- Dataset preparation for training (Training/Validation) of Object Detection model
- Setting up Darknet for YOLOv3 on Google Colaboratory(due to the availability of free GPU)
- Once satisfactory results are obtained after changing certain configurations according the problem, Validated the model
- Now, the test dataset was fed and the objects were detected and co ordinates stored in a json file(output of YOLOv3)
- Using the json file, extracted the coordinates and cropped the objects accordingly
- Fed the cropped Number Plates to Microsoft Vision API which in turn performs OCR over the image

ALGORITHMS/MODELS Tried:

A)Detection model:

YOLOv1,YOLOv2,YOLOv3.For implementing any of the Yolo version, we need a neural network framework i.e either Darknet or Darkflow

B)Text Extraction from Number Plate:

A number of Python pre existing libraries were tried out:

Pytesseract,Pyocr,textract. Google Vision API and Microsoft Vision API were also tried up.

Finalized Models/Algorithms:

- The solution was finally implemented using YOLOv3 with the neural network framework Darknet.
- For text extraction, Microsoft Vision API is considered.
- For certain image processing steps, a combination of PIL and OpenCV has been used

Implementation Framework

The implementation has been divided into the following major parts:

- Dataset Preparation

The dataset used has been a mix of two dataset:

- https://www.kaggle.com/achrafkhazri/anpr-dataset-tunisian-plates-and-digits
- https://www.kaggle.com/dataturks/vehicle-number-plate-detection

The dataset format required for YOLO training can be found out here on my blog Here

1. The first dataset that has xml files containing the plate object coordinates which were brought to the correct format using xml.etree.ElementTree python library
2. The second dataset is a json file from which labels are converted to required format from json
3. Hence, the data for training was with 600+ images and for testing with 100+ images.

- Detection using YoloV3 and Darknet

Git clone : https://github.com/AlexeyAB/darknet.git

Here, we need to get some files changed:

Download

- Initial weights https://drive.google.com/file/d/1zy5LN4GdXXc_azIpecSysi4qQM9esDEg/view?usp=sharing
- Config file & Makefile

  https://drive.google.com/file/d/1LUn13-uzHkPDaCjnNvJwiDnUWYROZeBB/view?usp=sharing

  https://drive.google.com/file/d/1DS5MJXmRDfQRJ-Tw8GprgoxKHuYI764q/view?usp=sharing

- Train.txt,Valid.txt, obj.data & obj.name

  https://drive.google.com/file/d/1GR4MAlj4l5RrqqXvCrCMwRKmc0Qz13NK/view?usp=sharing

  https://drive.google.com/file/d/1do-En6hXvgPv2JG4-g1ro5Dz8hlYCx9R/view?usp=sharing

  https://drive.google.com/file/d/1dr9hrp6GlBih7b7Nl8orf0nY3yN8OdFV/view?usp=sharing

  https://drive.google.com/file/d/1XMeEWbGjHvBo7m-wSIeNM03n0UNioTuQ/view?usp=sharing

Now do the following

1. Place yolo-obj.cfg & weights in /darknet
2. Place train.txt, test.txt, obj.data & obj.names in /darknet/data
3. Place the dataset in /darknet/data
4. Once done, run-make command in CLI in /darknet & chmod +x *.sh for making darknet.sh executable
5. Once done the cloned repository can be used for training,testing and calculating mAP for object detection.Confidence and threshold are set to 0.2 & 0.3 respectively for detection purpose

Dataset for OCR over Number Plate:

- After training, pass the test images for YOLO.Result obtained is json file with coordinates of Number Plate per image
- Using these results, cropped the Number Plate portion and stored in a separate folder
- Passed each cropped image to Microsoft Vision API which in turn return the text written

SOLUTION SUBMISSION

https://github.com/mehulgupta2016154/TCS_HUMAIN.git

Dataset used and Weights(initial and final) are uploaded on Google Drive with their links shared and mentioned in the README.md

APPENDIX

- **Results from training YOLOv3**

-The columns in the below table represents

Confidence,Precision,Recall,F1-score,AVG-IOU,Threshold,mAP when validate over same data(confidence & threshold are the parameters to be analyzed)

-It can be observed that best results are observed by setting confidence:0.2 & any threshold.Tested images were taken on a confidence=0.2,threshold=0.3

```
Loading weights from backup/yolo-obj_last.weights.
144Total Detection Time: 17.000000 Seconds
conf,pre,rec,f1-score,avg_iou,iou_thr,map
0.20,0.97,0.99,0.98,78.98,(mAP@0.10),99.16
0.15,0.96,0.99,0.97,78.44,(mAP@0.10),99.16
0.10,0.96,0.99,0.98,78.34,(mAP@0.10),99.16
0.05,0.95,0.99,0.97,77.81,(mAP@0.10),99.16
0.20,0.97,0.99,0.98,78.98,(mAP@0.05),99.16
0.20,0.97,0.99,0.98,78.98,(mAP@0.10),99.16
0.20,0.97,0.99,0.98,78.98,(mAP@0.15),99.16
0.20,0.97,0.99,0.98,78.98,(mAP@0.20),99.16
0.20,0.97,0.99,0.98,78.98,(mAP@0.30),99.16
0.20,0.96,0.98,0.97,78.65,(mAP@0.50),98.46
```

- **<u>Comparative study of different YOLO versions(YOLOv3 vs rest)</u>**

1. YOLOv3 has more layers(106 layers) as compared to YOLOv2(30 layers) & YOLOv1(26 layers)
2. YOLOv3 predicts smaller objects better(3 scaled versions of the same image passed) as compared to other versions
3. 9 anchor boxes taken than YOLOv2(3 anchors) and YOLOv1(No anchors) helping in better detection
4. Improved loss function for YOLOv3
5. Multilabel detections as compared to multiclass(YOLOv2 & YOLOv1)

- **<u>Analysis of various OCRs available(</u>**kindly zoom in)

Sortable table

Name	Founded year	Latest stable version	Release year	License	Online	Windows	Mac OS X	Linux	BSD	Programming language	SDK?	Languages	Fonts	Output Formats	Notes
Google Drive OCR or Google Cloud Vision			2015	Free	Yes	Browser	Browser	Browser	Unknown	Unknown	Yes	200+	All fonts	text	Google blog post [1] [2]
Tesseract	1985	4.0.0	2018	Apache	No	Yes	Yes	Yes	Yes	C++, C	Yes	100+[3]	Any printed font	Text, hOCR,[4] PDF, others with different user interfaces[5] or the API	Created by Hewlett-Packard; under further development by Google[6]
ABBYY FineReader	1989	14	2017-01-25	Proprietary	Yes	Yes	Yes	Yes	Yes	C/C++	Yes	192[7]	?	DOC, DOCX, XLS, XLSX, PPTX, RTF, PDF, HTML, CSV, TXT, ODT, DjVu, EPUB, FB2[8]	ABBYY also supplies SDKs for embedded and mobile devices. Professional, Corporate and Site License Editions for Windows, Express Edition for Mac.[9]
E-aksharayan	2010					Yes	No	Yes	No			14		RTF, TXT, BRL	
Asprise OCR SDK	1998	15	2015	Proprietary	Yes	Yes	Yes	Yes	Yes	Java, C#,VB.NET, C/C++/Delphi	Yes	20+[10]	?	Plain text, searchable PDF, XML[11]	Java, C#, VB.NET, C/C++/Delphi SDKs for OCR and Barcode recognition on Windows, Linux, Mac OS X and Unix.[12]
AnyDoc Software	1989	?	?	Proprietary	No	Yes	No	No	No	VBScript	?	?	?		Works with structured, semi-structured, and unstructured documents.
LEADTOOLS	1990[13]	19.0	2014	Proprietary	Yes	Yes	Yes	Yes	No	C/C++, .NET, Objective-C, Java, JavaScript	Yes	56[14]	Any printed font	PDF, PDF/A, DOC, DOCX, XLS, XPS, RTF, HTML, ANSI Text, Unicode Text, CSV[15]	Supports Latin, Asian, Arabic, and MICR character sets.[16] For full page, zonal, and form image processing. Includes OCR, barcode, OMR and forms recognition.[17] ICR (handwritten text recognition) is supported.[18]
CuneiForm	1996	1.1	2011-04-19	BSD variant	No	Yes	Yes	Yes	Yes	C/C++	Yes	28	Any printed font	HTML, hOCR, native, RTF, TeX, TXT[19]	Enterprise-class system, can save text formatting and recognizes complicated tables of any structure
Dynamsoft OCR SDK	2003	8.2	2012	Proprietary	Yes	Yes	No	No	No	C/C++	Yes	40+[20]	?	PDF, TXT	
OmniPage	1970s	19.2	2015	Proprietary	Yes	Yes	Yes	Yes	No	C/C++, C#[21]	Yes	125[22]	Machine and handprinted fonts	DOC/DOCX XLS/XLSX PPTX RTF PDF PDF/A Searchable PDF HTML Text XML ePUB MP3	Product of Nuance Communications
Microsoft Office OneNote 2007	2011	?	2007	Proprietary	No	Yes	No	No	No	?	?	?	?		
GOCR	2000	0.52[23]	2018-10-15	GPL	Yes[24]	Yes	Yes	Yes	Yes	C	?	20+	?		
Ocrad	?	0.26[25]	2017-03-31	GPL	Yes	No	Yes	Yes	Yes	C++	Yes	Latin alphabet	?		Command line
SmartScore	1991	10.5.8	2015-07	Proprietary	No	Yes	Yes	No	No	?	?	?	?		For musical scores
Microsoft Office Document Imaging	?	Office 2007	2007	Proprietary	No	Yes	No	No	No	?	?	?	?		Uses OmniPage[citation needed]
Puma.NET	?	?	2009-10-29	BSD	No	Yes	No	No	No	C#	Yes	28	Any printed font		.NET OCR SDK based on Cognitive Technologies' CuneiForm recognition engine. Wraps Puma COM server and provides simplified API for .NET applications
ReadSoft	?	?	?	Proprietary	No	Yes	No	No	No	?	?	?	?		Scan, capture and classify business documents such as invoices, forms and purchase orders integrated with business processes.
Scantron	?	?	?	Proprietary	No	Yes	No	No	No	?	?	?	?		For working with localized interfaces, corresponding language support is required.
OCRFeeder	2009-03	0.8.1	2014-12-22	GPL	No	No	No	Yes	No	Python	?	?	?		Features a full user interface and has a command-line tool for automatic operations. Has its own segmentation algorithm but uses system-wide OCR engines like Tesseract or Ocrad
OCRopus	2007	1.3.3	2017-12-16	Apache	No	No	Yes	Yes	Yes	Python	?	All languages using Latin script (other languages can be trained)	Normal Latin script and Fraktur (other scripts can be trained)	TXT, hOCR[26], PDF[27]	Pluggable framework under active development, used for Google Books
Name	Founded year	Latest stable version	Release year	License	Online	Windows	Mac OS X	Linux	BSD	Programming language	SDK?	Languages	Fonts	Output Formats	Notes

References

https://www.bankbazaar.com/driving-licence/vehicle-number-plate.html

https://en.wikipedia.org/wiki/Vehicle_registration_plate

https://www.acko.com/articles/general-info/vehicle-number-plate/

http://www.platerecognition.info/1106.htm

https://github.com/AlexeyAB/darknet

https://azure.microsoft.com/en-in/services/cognitive-services/computer-vision/

https://www.geeksforgeeks.org/xml-parsing-python/

https://pjreddie.com/darknet/yolo/

https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006

https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

https://pjreddie.com/media/files/papers/yolo.pdf

https://pjreddie.com/media/files/papers/YOLO9000.pdf

https://pjreddie.com/media/files/papers/YOLOv3.pdf

https://github.com/AvivSham/YOLO_V3_from_scratch_colab