**REAL TIME VEHICLE CLASSIFICATION AND LOCALIZATION USING EDGE COMPUTING**

[ Rahul Krishnan Swapnil Panwala Akhil John ]

                                                Department of Computer Science

Abstract

The project aims at detecting and localizing relevant objects in a video stream (Surveil-

lance Video) in real time. The project is a step forward in the era of Edge Computing, by delivering a fully functional and independent end-user device, such as the Raspberry Pi. We discuss in length the progress made leading upto the current development scene and possible future enhancements that can serve as motivation for further work. We study various working models and further divulge into the chosen model with a discussion on the steps involved for the same.



**LeadingIndia.ai: PROJECT REPORT**

Under the guidance of Dr. Vipul Mishra,

Asst.Professor, Bennett University

Contents

1. Introduction

1.1. Problem Statement

1.2. Steps Involved

1.3. Why and how the project was chosen

1. Preparing Custom Data-set
2. Setting-up system
3. Related Work
4. Development cycle

5.1. ResNet with Embedded Learning Library-Microsoft

5.2. Setting up Raspberry Pi3 (stretch)

5.3. You Only Look Once algorithm (Version 2)

1. Future Scope
2. Learnings and reflections from the project
3. References

1.    Introduction

Our project is an extended work on the blooming Computer Vision on vehicle localization. We implement a real-time system capable of accounting to vehicles entry-and-exit of subjected areas. Most notable and recent work in the industry has been credited to models trained on the western context, with subtle yet significant changes in contrast with the Indian context. Our model will finally be executed in a low spec device, the Raspberry Pi3 B and all further correspondence to our project is parallel with implementation of the same in Pi.

Most exhaustively used models include resNet and the YOLO model. Both of which has significant accuracy, but again with the western context. We lean towards the two as they have superior localization and classification in addition to easier modification of layers, via **transfer learning**. We further seek to implement a fully functional system trained on indian vehicles. Our expected levels of accuracy and threshold values are mentioned in later parts of the report.

**1.1  Problem Statement**

Due to the extensive rise of urbanization, there is an abrupt increase in activity and a bombardment of vehicles on the Indian roads, thus there rises a very potent need for a smart device that can not only detect the activity but also monitor it, read on it’s data and give significant inputs to maintain ease of control over months of assimilated input from our real-time feeds. The project was aimed at classifying multiple classes of vehicles based off of a real-time camera feed to help monitor traffic activity in the Indian context. (Present use case, Fixed cameras setup on a Pi board in security gate).

**1.2  Steps Involved**

* Identify vehicle presence
* Classification based on class
* Localization
* Tracking Image (Real-Time)

**1.3  Why and how the project was chosen**

The main inspiration behind this project was our observation of guards maintaining manual records of vehicles entering or leaving the University campus. We found a need to automate this process of maintaining records. A device could be made that could take accurate records not only at gates but also at different locations. The data collected from such devices, over some months or years could give insight into people’s activity and the growth of a city.

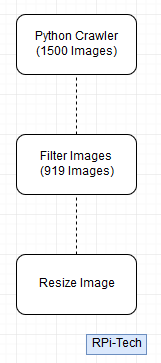
2    Preparing custom data-set

We cover the sources from which we obtained the data and the data-manipulation required to make it model-ready for training in this section. We divide the different classes of vehicles to different sub-topics and explain source and data-cleaning required.

2.1 Auto-Rickshaws:

We obtained around 60% of our data from cvit Auto-rickshaw detection challenge ([Open Data](http://cvit.iiit.ac.in/autorickshaw_detection/)). Further data was scraped from the google search engine results. [Google API is deprecated]. All images of the cvit data-set followed uniform scale and did not require data-cleaning.

 2.1.1 Web-scraped Images:



*visualizing the process we followed for the data-cleaning*

2.2 Rest Classes:

We obtained the images of rest of the classes namely motorbike, bicycle, car and bus from PASCAL VOC2012 dataset.

3  Setting up system

Admin system configuration:

|  |  |
| --- | --- |
| Operating System | Windows 10 |
| Central Processing Unit | Intel Core i5-8250u 1.6GHz with Turbo boost |
| Graphics Card | (Laptop Graphics) Nvidia GeForce MX150 |
| Memory (Hard-disk) | 1TB HDD |
| RAM | 8GB |

3.1 Software requirements:

3.1.1    TensorFlow for Windows (GPU support)

A detailed account on the set-up of the TF with GPU support has been logged [here](https://rahulkrishnanlive.wordpress.com/2018/06/15/tensor-flow-with-gpu-support-windows-updated-on-15-06-18/) in our blog. (Version numbers are consistent with one's used to test the models).

3.1.2    OpenCV for windows

3.1.3    Keras (TensorFlow backend)

3.1.4    Python 3.6

3.2) Raspberry Pi3 setup:

         3.2.1      CMake

3.2.2      openBLAS

3.2.3   Curl

3.2.4   Python 3+ with MiniConda

3.2.5      OpenCV

3.2.6   Performance Tweaks([link](https://microsoft.github.io/ELL/tutorials/Setting-up-your-Raspberry-Pi))

3.2.7 TensorFlow

**

*Raspberry Pi setup used*

4. Related Work

We could readily find various pre-trained models/projects like

### [tatsuyah/vehicle-detection](https://github.com/tatsuyah/vehicle-detection)

### [andrewssobral/simple\_vehicle\_counting](https://github.com/andrewssobral/simple_vehicle_counting)

### [xslittlegrass/CarND-Vehicle-Detection](https://github.com/xslittlegrass/CarND-Vehicle-Detection)

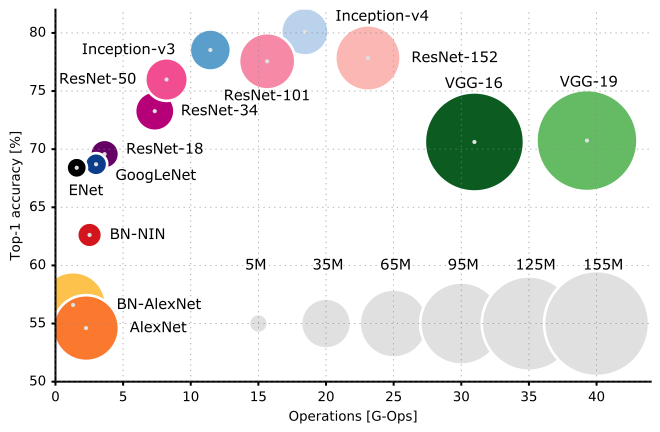
### [JunshengFu/vehicle-detection](https://github.com/JunshengFu/vehicle-detection)

### …. And many more

### But, the major problem was all these models were working and devices with higher specs than Raspberry Pi, so we had to create highly optimised light model to run our program on Raspberry Pi.

5.   Development Cycle

**5.1 ResNet with Embedded Learning Library-Microsoft**



*Comparison of various models*

Host operating system along with the Raspberry Pi3 (B) set-up has been well documented [here](https://github.com/Microsoft/ELL/blob/master/INSTALL-Windows.md). We spent most of week one and three days through week two implementing the requisite set-up for ELL in the native PC, to run ResNet for image classification. We are attaching the anomalies and difficulties faced during installation through the following lines.

**Setting up your windows machine**

1)LLVM 3.9, SWIG 3.0.12, OpenBLAS 0.2.19.3, and Doxygen 1.8.13 via NuGet installation:

* Packages left un-necessary “obj” directory that required to be moved to an external path outside root directory.
* Left-over files needed extra manual installation outside the documented list. This entry might vary according to system settings and current installation scene of the softwares.

FIX: Manually cross check version numbers and requisite packages from the json file in the root repository to identify missing packages and install manually via download and setup(.exe) or NuGet(exhaustive, not recommended).

2) Creating Environment:

* Miniconda installation required. (Active root path set as the environment and all further builds and make in active environment.)

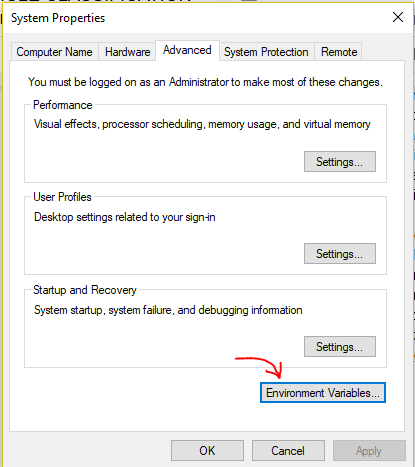
3) Build and Make(Common errors and failed builds):

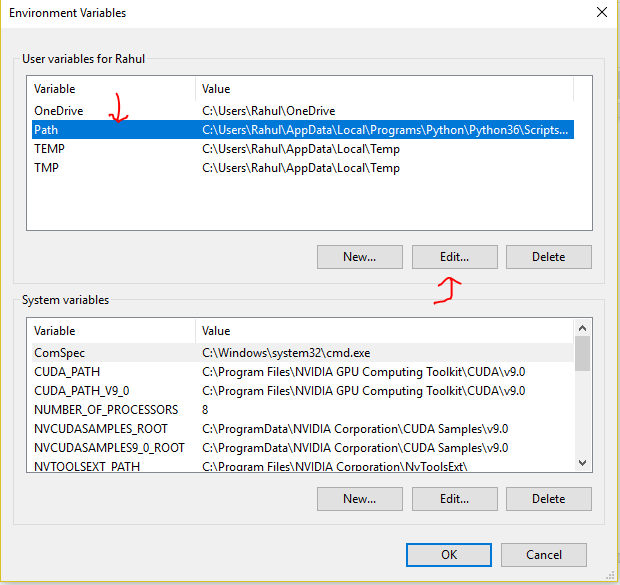
* Cmake fails on linux system in contradiction with windows OS.

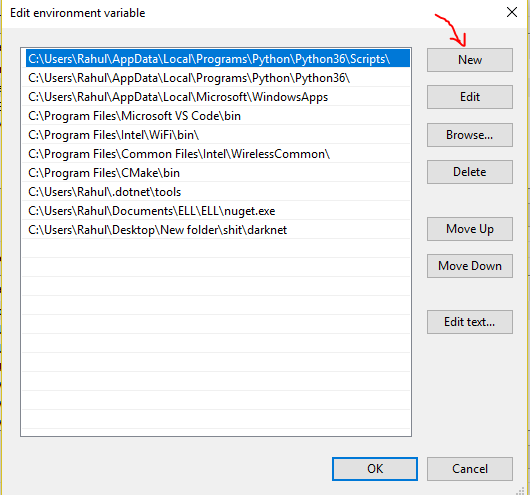
FIX: Common errors are largely attributed to version mismatch and can be fixed by running the same model of cmake(3.9v) as suggested in the json files. Further experimentation resulted in failed builds for both older and newer versions as large amounts of testing was originally carried out in the 3.9 version.

         4) “*Windows cannot access the specified device, path, or file" error when you try to install, update or start a program or file”*

* FIX: adding program to path in windows solves the error. Steps illustrated below.







**Un-resolved issues:**

Import model from CNTK failed : [Link](https://github.com/Microsoft/ELL/issues/160)

5.2) **Setting up Raspberry Pi3 (stretch)**

1. Import module failure : OpenCV

* Working FIX: (Installation guide) → [Link](https://www.pyimagesearch.com/2017/09/04/raspbian-stretch-install-opencv-3-python-on-your-raspberry-pi/)
* Alter:  Installing python-dev and then remaking/reinstalling did the trick.

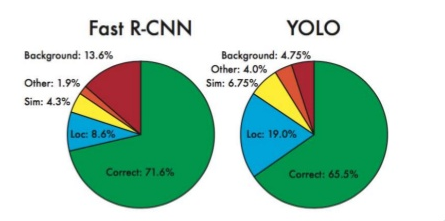
    2) Setting up Apache server and telegram CLI for automated feed response to mobile device:

* Issues opened in associated repository →  [Link](https://github.com/PiSimo/PiCamNN/issues/8)
* Dependencies : *libreadline-dev libconfig-dev libssl-dev lua5.2 liblua5.2-dev libevent-dev libjansson-dev libpython-dev*
* Modified configuration
* Run build on modified configuration.

  3)  Keras with Tensorflow backend (Theano not recommended)

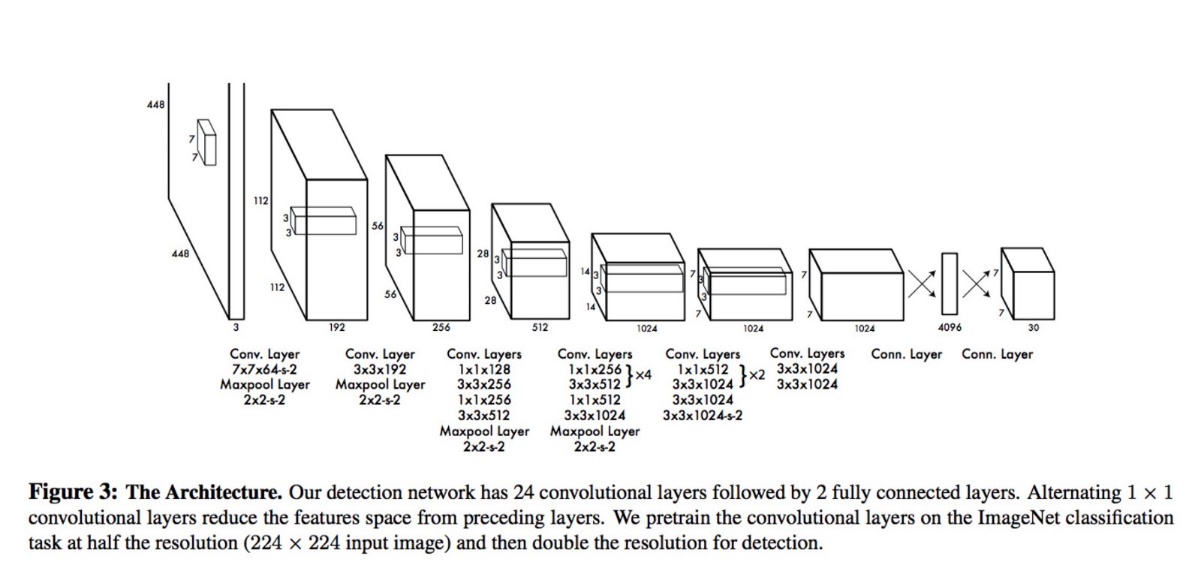
* Installation guide (python 3+) →   [Link](https://www.pyimagesearch.com/2016/11/14/installing-keras-with-tensorflow-backend/)

**5.3) You Only Look Once algorithm (Version 2)**

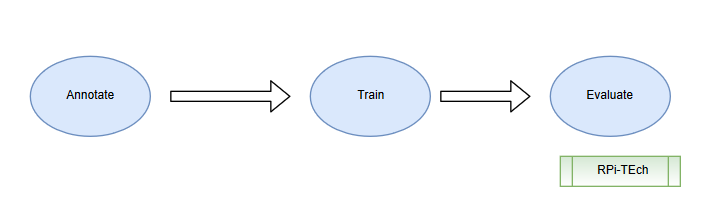


Through week 3 and until the point of writing this document we have shifted focus on developing a overlaying model on top of YOLO architecture and in general Tiny YOLO weights, since arbitrarily large networks like ResNet deemed to be an inefficient wastage of memory on a short and fixated memory block device such as the Pi. Tiny YOLO reduces total space consumption by around 35% with negligible loss in accuracy or correctness of prediction.

**5.3.1. First presentation-Dry run on YOLO v2 (Dated : Week one)** → [Link](https://www.youtube.com/watch?v=L0dR9TxhQSA)



**5.3.1.1. Custom Image detection on Auto-rickshaw data:**



**Annotation :**

After multiple failed attempts at a pre-trained data-set, we had to scrape google search images to assimilate our data-set.

[**Note:** Google API is deprecated]

The major manual task we faced, that took most man hours was allocating annotations or bounding boxes. Major work was completed via an automated script (Credits to [Mark Jay](https://www.youtube.com/channel/UC2W0aQEPNpU6XrkFCYifRFQ))

* Initially, we generated .xml files for the annotations specifications of each images iterated over the entire length of the auto-rickshaw data-set. (approx. 10hrs)
* We trained the model in the NVIDIA DGX supercomputer until our loss function reached it’s stable level (reduction from 108 to 1.2). [Approx. 15 epochs]  
  **Configuration at set-up time (trial run 2)**

|  |  |
| --- | --- |
| Epochs | 15 |
| Batch-size | 16 |
| Final Step | 1550 |

* We then tested it on a script that ran the model from the pre-loaded (newly trained) weights in real-time using a plug and play logitech camera.
* First sample results (76% accuracy) [Notable increase in real-time data over images]



**Key Issues:**

* Run-time error faced: [Link](https://github.com/markjay4k/YOLO-series/issues/16) [We solved the error and updated files and let the repository owner know of the changes.]
* Configuration MisMatch error: Outdated website YOLOv2 has requisite weights for our dark-flow repository model and hence error occurs. Solved by using matching cfg and weights files.
* Running on the NVIDIA DGX: Our native running environment is Windows and a transition to linux via the super-computer raised errors.   
   FIX: We solved this by running the following command and installing cython independently.

[**“pip install -e .”**](https://github.com/markjay4k/YOLO-series/blob/master/part1%20-%20setup%20YOLO.ipynb)

**5.3.1.1. Classes Localization on Custom dataset:**

After getting success in locating auto rickshaws, we combined the auto rickshaw dataset with the pascal VOC dataset we had to get the following training dataset :

|  |  |
| --- | --- |
| **Classes** | **Number of images for trainig** |
| Autorick | 1016 |
| Bicycle | 552 |
| Bus | 421 |
| Car | 1161 |
| Motorbike | 526 |

Considering some images having multiple classes in them we had 3,057 images. The following are the training details.

|  |  |
| --- | --- |
| Epochs | 50 |
| Batch Size | 64 |
| Checkpoint | 4000 |
| Training samples | 3057 |
| Final Loss Function Value | 2.1840163707733154 |
| Accuracy | 93.06 % |

After the training a python code was written for real-time analysis. It basically located classes from video frame and created boundary boxes. The centre of those boxes were tracked real-time to find out whether the object is entering, leaving or is at rest on the part of road surveyed. The real-time results were displayed on top of screen.

* + 1. Future Scope

For better execution of our model, the code could be further optimised by removing the dependencies not required for execution of real-time processing. Further the model could be also trained for number plate detection to get precise data for surveillance. This would not only result to higher precision but we would be able to collect more data.

* + 1. Learnings and reflections from the project

It was really a great learning experience working for this project. Our team members were working on AI for first time, so it was little had for us but we were successful to learn about AI and working on it in a project-based manner. We were also able to learn about its possibilities in future. Coming from different Universities, team members were also able to learn about each other and the Institutions they belong to. Our team was also able to know about the fast developing fields like edge computing. Team work was our key to successfully complete our project.

* 1. References

# We used [**thtrieu**](https://github.com/thtrieu)**/**[darkflow](https://github.com/thtrieu/darkflow) **for transfer learning/fine tuning.**

1. Dataset was derived from [Pascal VOC2012](http://host.robots.ox.ac.uk/pascal/VOC/voc2012/) and  [NCVPRIPG '17](http://cvit.iiit.ac.in/autorickshaw_detection/).
2. This project was conducted under [LeadingIndia.ai](https://leadingindia.ai/).
3. Model used was [tiny Yolo v2](https://pjreddie.com/darknet/yolo/) trained on Pascal VOC2012 dataset.
4. The project and workshop was conducted by [Bennett University](https://www.bennett.edu.in/) .
5. The model was run on Raspberry Pi 3B along with a Logitech webcam