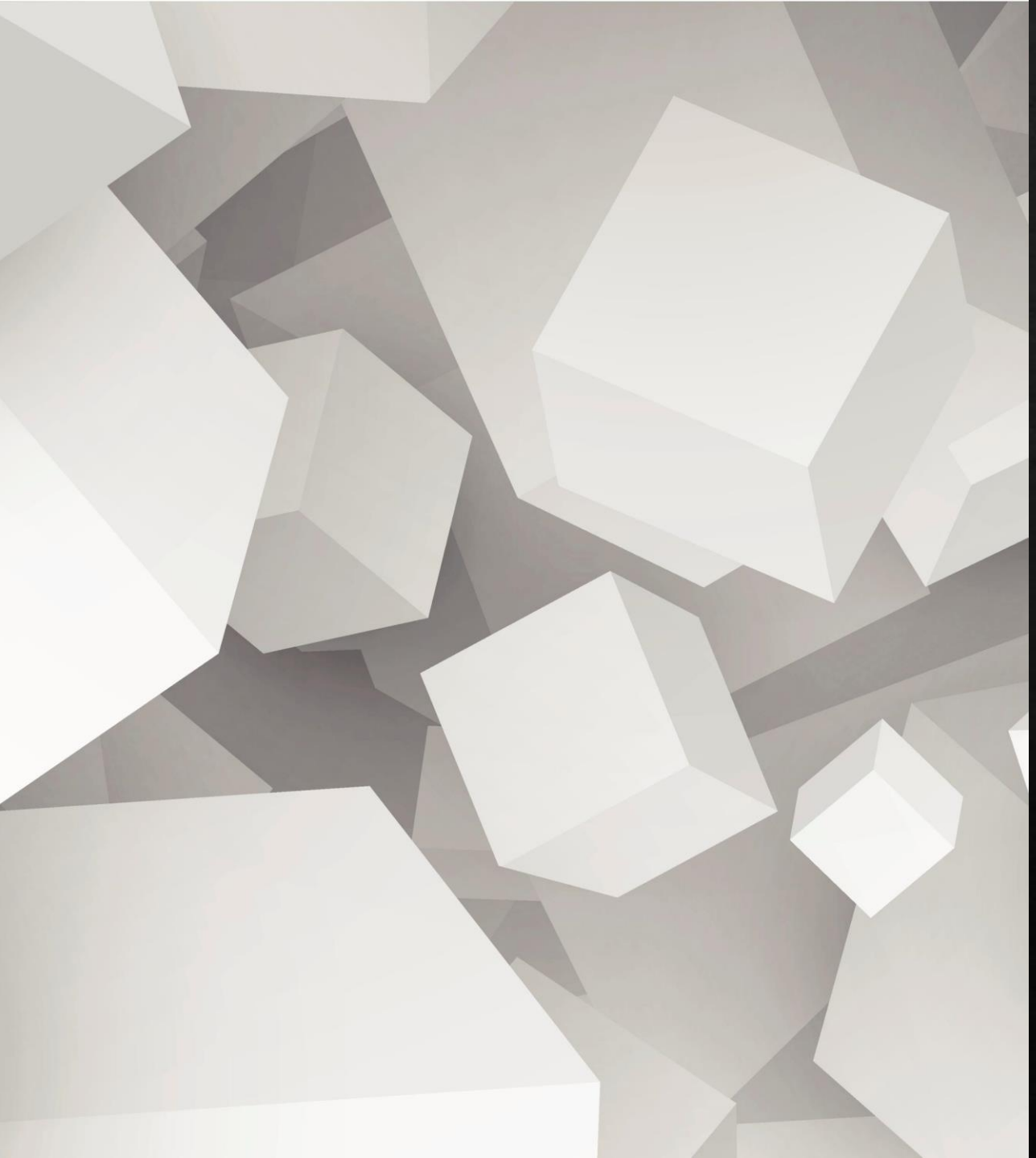
Adaptive Traffic Signal Control System

FS21 ECE830-001

Embedded Cyber-Physical Systems

Aditya Dhaka - Aarush Mathur



Adaptive Traffic Signal Control System

Introduction

Applications

TensorNets & keras-yolov3

YOLO

YoloV3 Car Counter

Working

Sequence of operations performed

Code (Synchronization logic)

Demo

Result

Conclusion

Future Work

References

Introduction

Traffic congestion is increasing day by day on roads due to increase in the number of cars. Vehicles waiting in queue for long time to be processed at the intersection and the traditional traffic lights are not efficient enough to schedule it properly. We can use computer vision and machine learning to have similar details like traffic at the signalized road intersection. This is done by using real-time object detection based on a deep Convolutional Neural Networks called You Only Look Once (YOLO). Traffic signal phases are then calculated and optimized according to collected data about queue density and waiting time per vehicle to enable as much as more vehicles to pass safely with minimum waiting time. YOLO can be implemented on embedded controllers using Transfer Learning technique.

The projects is based on –

- Tensor nets
- keras-yolov3 repository

What is TensorNets?

High level network definitions with pre-trained weights in TensorFlow

We installed TensorNets from PyPI

Guiding principles

- Applicability - Many people already have their own ML workflows, and want to put a new model on their workflows. TensorNets can be easily plugged together because it is designed as simple functional interfaces without custom classes.
- Manageability - Models are written in tf.contrib.layers, which is lightweight like PyTorch and Keras, and allows for ease of accessibility to every weight and end-point. Also, it is easy to deploy and expand a collection of pre-processing and pre-trained weights.
- Readability - With recent TensorFlow APIs, more factoring and less indenting can be possible. For example, all the inception variants are implemented as about 500 lines of code in TensorNets while 2000+ lines in official TensorFlow models.
- Reproducibility -We can always reproduce the original results with simple APIs including feature extractions. Furthermore, we don't need to care about a version of TensorFlow beacuse compatibilities with various releases of TensorFlow have been checked with Travis.

A Keras implementation of YOLOv3

- Download YOLOv3 weights from YOLO website.
- Convert the Darknet YOLO model to a Keras model.
- Run YOLO detection.

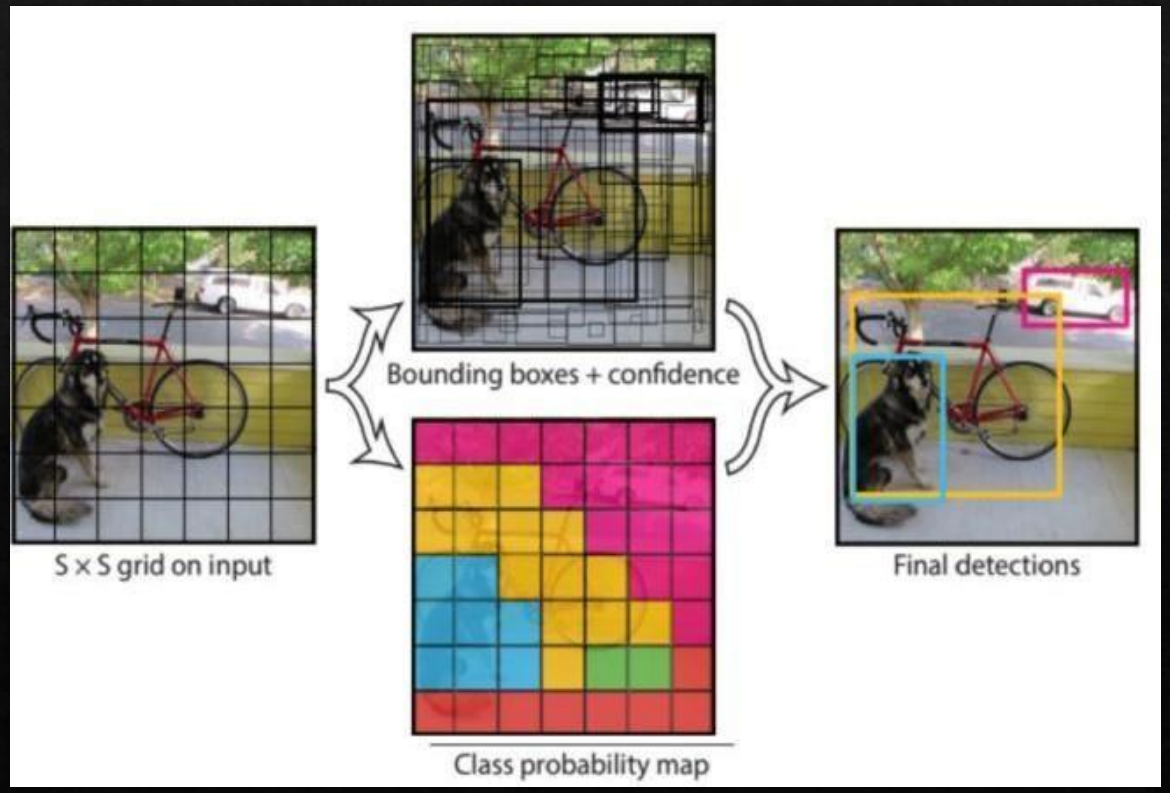
Applications

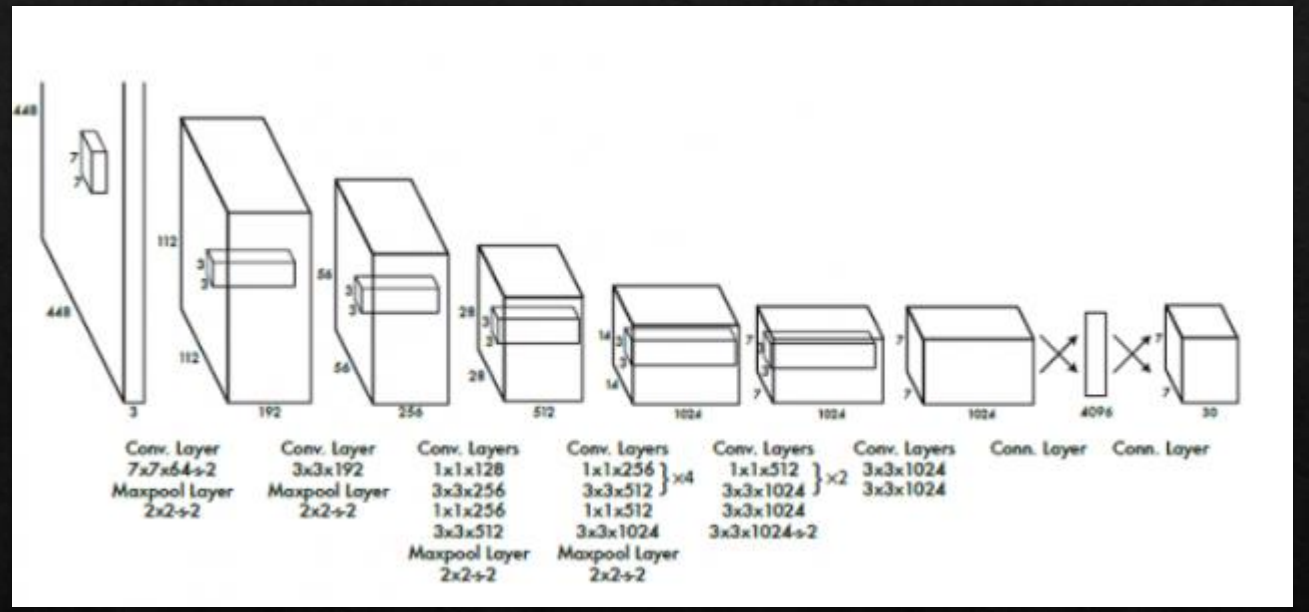
Traffic congestion is a major problem in many cities. Traffic signal controllers with fixed-cycle are not resolving the high waiting time in the intersection. We often see policeman managing traffic light. He sees the cars density on roads based on this he decides the allowed duration of traffic light. Based on this human achievement we are creating a smart Traffic light control taking into account the real time traffic condition that manages traffic at intersection. To implement this we need two main components: eyes to watch the real-time road condition and a brain to process it. A traffic signal system at its core has two major tasks: move as many users through the intersection as possible doing this with as little conflict between these users as possible.

YOLO

You only look once (YOLO) is a state-of-the-art, real-time object detection system YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

The object detection task consists in determining the location on the image where certain objects are present, as well as classifying those objects. Previous methods for this, like R-CNN and its variations, used a pipeline to perform this task in multiple steps. This can be slow to run and also hard to optimize, because each individual component must be trained separately. YOLO, does it all with a single neural network.




448
448
7
7
3
112
112
192
3
3
56
56
256
3
3
28
28
512
3
3
14
14
1024
3
3
7
7
1024
3
3
7
7
1024
4096
7
7
30
Conv. Layer
7x7x64-s-2
Maxpool Layer
2x2-s-2
Conv. Layer
3x3x192
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x128
3x3x256
1x1x256
3x3x512
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x256
3x3x512
×4
1x1x512
3x3x1024
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x512
3x3x1024
×2
3x3x1024
3x3x1024-s-2
Conv. Layers
3x3x1024
3x3x1024
Conn. Layer
Conn. Layer

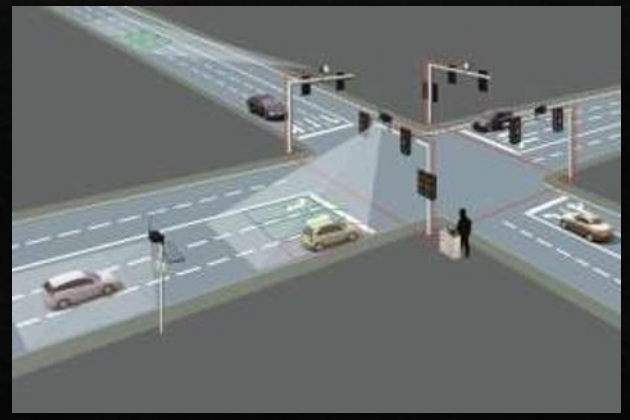
YoloV3 Car Counter

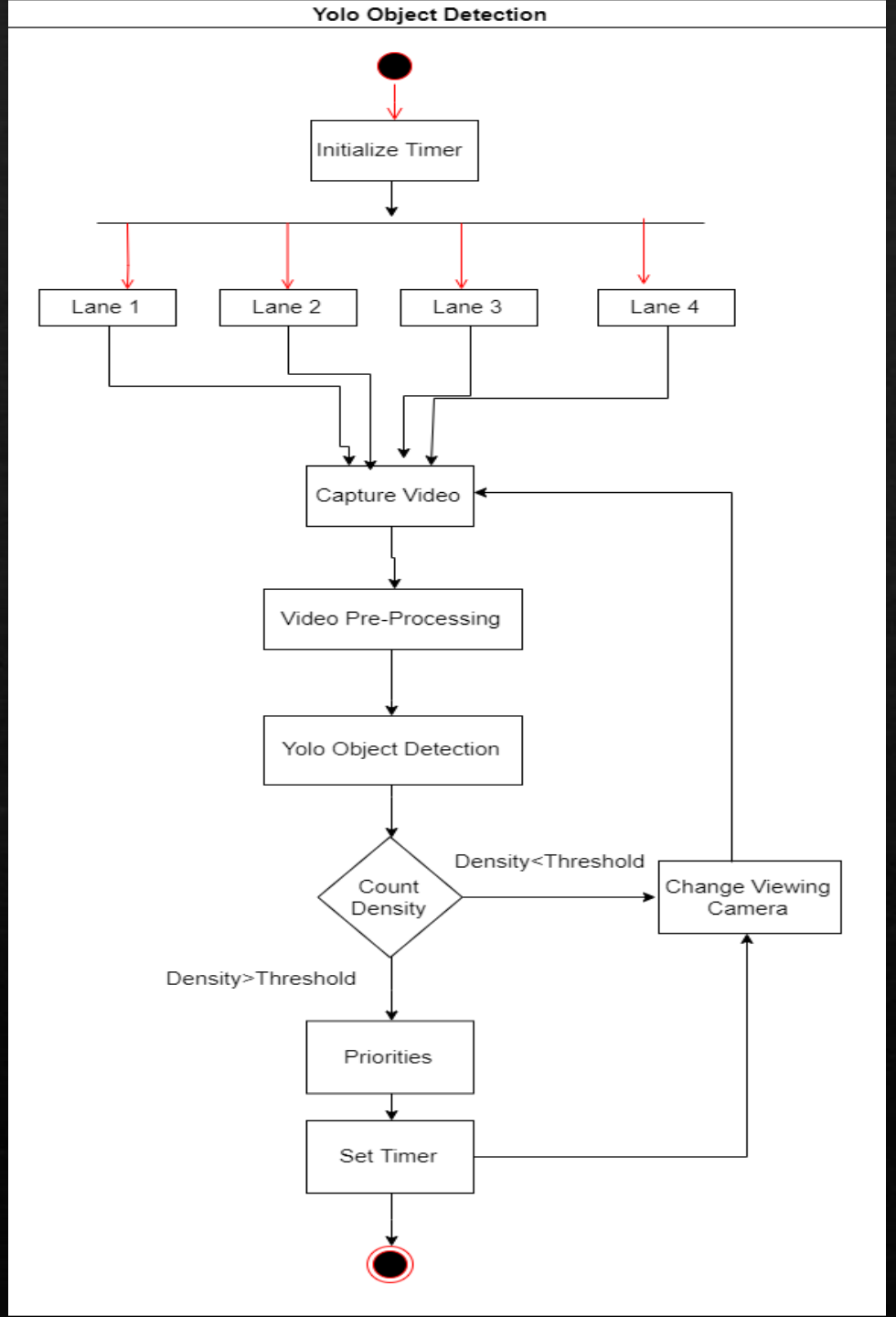
This is a demo project that uses YoloV3 neural network to count vehicles on a given video. The detection happens every x frames where x can be specified. Other times the dlib library is used for tracking previously detected vehicles. Furthermore, you can edit confidence detection level, number of frames to count vehicle as detected before removing it from trackable list and the maximum distance from centroid , number of frames to skip detection (and only use tracking) and the whether to use the original video size as annotations output or the YoloV3 416x416 size.

YoloV3 model is pretrained and can be downloaded from internet .

Working

- The solution can be explained in four simple steps:
- Get a real time image of each lane.
- Scan and determine traffic density.
- Input this data to the Time Allocation module.
- The output will be the time slots for each lane, accordingly.




Yolo Object Detection
Initialize Timer
Lane 1
Lane 2
Lane 3
Lane 4
Capture Video
Video Pre-Processing
Yolo Object Detection
Count Density
Density<Threshold
Change Viewing Camera
Density>Threshold
Priorities
Set Timer

Sequence of operations performed

I. Camera sends images after regular short intervals to our system.

II. The system determines further the number of cars in the lane and hence computes its relative density with respect to other lanes.

III. Time allotment module takes input (as traffic density) from this system and determines an optimized and efficient time slot.

IV. This value is then triggered by the microprocessor to the respective Traffic Lights.

Code (Synchronization logic)

- f = open("out.txt", "r")
- no_of_vehicles=[]
- no_of_vehicles.append(int(f.readline()))
- no_of_vehicles.append(int(f.readline()))
- no_of_vehicles.append(int(f.readline()))
- no_of_vehicles.append(int(f.readline()))
- baseTimer = 120 # baseTimer = int(input("Enter the base timer value"))
- timeLimits = [5, 30] # timeLimits = list(map(int,input("Enter the time limits ").split()))
- print("Input no of vehicles : ", *no_of_vehicles)

- t = [(i / sum(no_of_vehicles)) * baseTimer if timeLimits[0] < (i / sum(no_of_vehicles)) * baseTimer < timeLimits[1] else min(timeLimits, key=lambda x: abs(x - (i / sum(no_of_vehicles)) * baseTimer)) for i in no_of_vehicles]
- print(t, sum(t))

Result


ID 13
ID 7
ID 9
ID 5
ID 10
ID 12
ID 11
ID 2
ID 4
ID 8
ID 1
BLACK
BLUE!
ashish

Conclusion

The goal of this work is to improve intelligent transport systems by developing a Self-adaptive algorithm to control road traffic based on deep Learning. This new system facilitates the movement of cars in intersections, resulting in reducing congestion, less CO2 emissions, etc. The richness that video data provides highlights the importance of advancing the state-of-the-art in object detection, classification and tracking for real-time applications. YOLO provides extremely fast inference speed with slight compromise in accuracy, especially at lower resolutions and with smaller objects. While real-time inference is possible, applications that utilize edge devices still require improvements in either the architecture's design or edge device's hardware. Finally, we have proposed a new algorithm taking this real-time data from YOLO and optimizing phases in order to reduce vehicle waiting time.

Future Work

We can easily extend this project by changing the classes we are interested in detecting and tracking.

We can also adjust this project for identifying the presence, location, and type of more objects in a given video.

References

[1] https://www.fhwa.dot.gov/innovation/everydaycounts/edc-1/pdf/asct_brochure.pdf

[2] Sven Tomforde, Christian Müller-Schloer "Incremental design of adaptive systems", Journal of Ambient Intelligence and Smart Environments, vol. 6, no. 2, pp. 179-198, 2014

[3] Jason Brownlee, Deep Learning for Computer Vision, May 27, 2019

[4] https://modelzoo.co/model/tensornets

Thanks you!