IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 8, No. 2, June 2019, pp. 107~119
ISSN: 2252-8938, DOI: 10.11591/ijai.v8.i2.pp107-119

Adaptive real time traffic prediction using deep neural networks

Parinith R Iyer[1], Shrutheesh Raman Iyer[2], Raghavendran Ramesh[3], Anala MR[4], K.N. Subramanya[5]
[1,2,3,4]Department of Computer Science and Engineering, R.V. College of Engineering, India
[5]Principal, R.V. College of Engineering, India

Article Info

Article history:

Received Jan 5, 2019
Revised Mar 6, 2019
Accepted May 9, 2019

Keywords:

Intelligent transportation systems
Neural network
Object detection/classification and count
Traffic signal time
Weighted moving average

ABSTRACT

The ever-increasing sale of vehicles and the steady increase in population density in metropolitan cities have raised many growing concerns, most importantly commute time, air and noise pollution levels. Traffic congestion can be alleviated by opting adaptive traffic light systems, instead of fixed-time traffic signals. In this paper, a system is proposed which can detect, classify and count vehicles passing through any traffic junction using a single camera (as opposed to multi-sensor approaches). The detection and classification are done using SSD Neural Network object detection algorithm. The count of each class (2-wheelers, cars, trucks, buses etc.) is used to predict the signal green-time for the next cycle. The model self-adjusts every cycle by utilizing weighted moving averages. This system works well because the change in the density of traffic on any given road is gradual, spanning multiple traffic stops throughout the day.

Copyright © 2019 Institute of Advanced Engineering and Science.
All rights reserved.

Corresponding Author:

Shrutheesh Raman Iyer,
Department of Computer Science and Engineering,
R.V. College of Engineering,
Bengaluru, Karnataka 560059, India.
Email: shrutheesh.ir@gmail.com

1. INTRODUCTION

The traffic congestion problem, especially in urban areas, has been worsening over the last 20 years. Based on the survey by Economic Times [1], it has been observed that as of 2016, the number of vehicles in Bengaluru, India (66.65 lakhs) has risen to 6.7 times of what it was in 1996 (9.93 lakhs), and 2.4 times that of 2006 (28.02 lakhs). Similar statistics are found in multiple other surveys on India as well [2-3]. The result of this rapid growth has many adverse effects, including deteriorating health conditions, average commute time of over three hours a day and poor air quality. This sharp growth and inadequate means to handle it have left many roads in a state of disrepair. These issues identify the shortcomings of traffic signals used on most roads today, in handling traffic. Most cities have fixed traffic light control systems in place which don't accommodate the dynamic nature of traffic flow.

The fixed-time traffic signals are inefficient, as the duration of time for which signal is open is usually insufficient for busy roads, and excessive for less busy ones (with fewer vehicles). The current system has left much to be desired. A variant of the fixed system is used sometimes. It involves assigning different green-times throughout the day to accommodate rush hours, but they were still "fixed". A viable solution would be to check the change in traffic density frequently (say, every few minutes), determine which road is clogged and decide, in real-time, the time required to clear the traffic on every road. The problem this paper aims to solve is the extended waiting time at traffic intersections, it does so by proposing a novel algorithm to make the allotted green times adaptive with the traffic volume at that point in time. The goal is to create a system that implements the ideas put forth above.

The aim of this paper is to present a novel approach to the dynamic traffic signal system based on intuitive object detection techniques using Convolutional Neural Networks coupled with Single Shot Detectors and provide a predictive model to decide upon the green-time of the traffic signal, for the respective lanes. It can also rectify itself quickly with the ongoing trend of traffic, as fast as one cycle of the signal (one cycle involves all four roads getting green time once). The density of traffic in each lane is used to control traffic signals by predicting the amount of time the signal needs to open. Thus, the duration of "green signal" (signal is open) is dictated by the contribution of the said lane to the traffic build up. The model proposed here can potentially be deployed on a small, portable system like the Nvidia Jetson Kit.

2. RELATED WORK

Previous work in Intelligent Transportation Systems (ITS) have been very promising with a few caveats which can potentially cause problems in the implementation phase in certain locations. Our previous work [4] presents vehicle detection and counting using image processing techniques. This paper explores the deep learning techniques to achieve the same task. Majority of the work tackling this problem has been sensor oriented [5]. There have been propositions involving placing sensors on either side of the road, or in other places almost at the ground level to determine the number of vehicles moving or to generally estimate the density of vehicles and hence decide how much green-time to allocate. The most prominent of these are the loop detectors which are excellent in calculating traffic volume. Yet, they do not perform well in estimating the traffic density. In [6], the use of RFID sensors has been proposed to detect vehicles moving through some portion of the road. Any approach involving deploying sensors approximately at the ground level will be infeasible as not all places enable such deployment of sensors. The developing countries have roads which sometimes are indistinguishable from the footpath as shown in Figure 1.

Figure 1. A Frequently commuted road in Bengaluru

In [7], the proposed system makes use of Ultrasonic sensor and claims to place it "on top of the road" but the maximum range of the sensor is just 4 meters, which is not feasible. In [8], the system uses IR sensors and Arduino and plans to mount the sensors on either side of the road on the poles. Deploying sensors in such an environment is not possible and such approaches are not best suited to solve the issue. Recently, approaches have shifted to camera-based solutions. These methods attempt to find the vehicle density on each lane and consequently determine the traffic signal time. The early approaches dealt with blob extraction methods such as in [9]. Although they were more versatile than the sensor methods, their accuracy left a lot to be desired. In recent times, research based on detection, tracking and classification of vehicles from video images has become feasible. Thanks to breakthroughs in computer vision technology and incremental advancements in compute power. The results are found to be very promising as well. Convolutional Neural Networks (CNN) provide near human level accuracy in the field of object detection. In [10], a successful camera based counting method was implemented by the means of multi-object tracking. Each vehicle on the road is tracked until it goes out of frame. This hence gives an accurate count. However, this process is extremely computationally expensive since object tracking involves redundancy. This becomes a problem in systems that have constrained hardware resources.

In [11], a deep neural network was used on videos with a regression approach to collectively count the number of vehicles to be counted, thus not allowing room for classification. In majority of the camera-based approaches such as the ones discussed above, attempts are made to calculate the red time in a given traffic junction for a lane. This system is not feasible for a few reasons. From the view of the signal, vehicles

far off in the road, waiting in the signal, will not be identified accurately. As the distance increases, the size of the objects captured by the camera is both small and not entirely visible (as parts may be cut off by cars in front), this, therefore, exposes the constraints of such models. Thus, this paper presents an innovative approach that has borrowed some elements from the aforementioned research to create a predictive and adaptive traffic control signal. In this approach, based on the number of vehicles passing through the junction, the green time for the particular lane is estimated for the next cycle. In this way, it is both self-correcting and a realistic implementation.

The basic principle in any solution for this specific problem is clear, the number of vehicles is the most important parameter one should obtain to proceed further in any way and to ultimately allocate appropriate green-time. It is clear that using sensors is not feasible, so more straight-forward, human-like counting approaches came about. In [12], determining the density of objects in a crowded scene using the Hydra-CNN and the Counting-CNN is proposed which is revolutionary in terms of mass public place surveillance systems and jam-packed roads. This still falls short of ground truth numbers when the scene it evaluates is crowded beyond an upper-threshold or when the scene is very lightly packed below a lower threshold. Adding to this, the model does not provide real-time classification and counting of different classes of objects in the crowded scene. This will be important when processing a live feed from a mounted camera meant for surveillance purposes comes into picture. This paper discusses a few additional features adopted which might help the ITS sector in some ways:

1. Vehicle detection and classification from a camera's video feed.
2. A probabilistic system of predicting the green-time based on the obtained count of different classes of vehicles.
3. Constantly adapting the green-light time based on the changing numbers of vehicles resulting in optimal time efficiency for a cycle.

3. RESEARCH METHOD

The entire procedure is divided into 3 steps. The primary step is the vehicle detection, in which classes of vehicles are detected in a video feed. This is then fed to the second step which involves counting the vehicles in a frame and subsequently counting the total number of vehicles in the video feed. The final step comprises of predicting the 'green time' of the next cycle. In short, the steps are listed as:

a. Vehicle detection
b. Counting of vehicles in a frame and hence from the video feed.
c. Predicting the "green-time" of the next cycle.

3.1. Vehicle detection

Vehicle detection is an instance of object detection in images and videos. Object detection has made great strides in the last 5 years since the advent of AlexNet [13] for image classification. Image classification is the task of assigning or identifying the class of an object in an image. Deep learning approaches, particularly Convolutional Neural Networks have produced near human-level accuracy in this field. Every CNN architecture has four parts – convolution, non-linearity, subsampling followed by classification. Object detection is the procedure of classifying objects in an image and localizing the extent of the object in the image by drawing bounding boxes around it. The first model, Region Based Convolutional Neural Networks (R-CNNs) [14] intuitively begin with the region search and then perform the classification. Since then several models have been developed that have greatly improved performance such as Fast R-CNN [15], Faster R-CNN [16], Region Based Fully Convolutional Neural Networks (R-FCN) [17]. These methods model object detection as a classification problem. These methods are very accurate but come at a big computational cost (low frame-rate), in other words, they are not fit to be used on embedded devices.

Recent approaches combine these two tasks into one network. Instead of having a network produce region proposal, a set of pre-defined boxes are initialized to look for objects. This class of detectors is known as Single Shot Detectors. The algorithm this paper uses is the Single Shot MultiBox Detector (SSD), by Google [18]. SSD performs considerably better than its competitors, both in terms of speed as well as accuracy. The performance in terms of speed is measured on the Frames Per Second (FPS) it processes. For 300 × 300 input, SSD achieves 74.3% mAP on the VOC2007 dataset at 59 FPS on an NVidia Titan X. This is in comparison to Faster R-CNN at 7 FPS with mAP 73.2% and YOLO at 45 FPS with mAP 63.4% [16].

The SSD MultiBox method has 3 aspects:

- Single Shot: The tasks of object localization and classification are done in a single forward pass of the network
- MultiBox: The technique for bounding box regression

– Detector: The network is an object detector that classifies those detected objects

The SSD architecture this paper employs, as shown in Figure 2, is the same as the one presented by [18], which bases itself on VGG architecture. This paper uses an already existing model that suited the needs and have utilized the Keras port of Single Shot MultiBox Detector, by Pierluigi Ferrari [19]. This is built on the Keras framework that runs Tensorflow in the back-end.

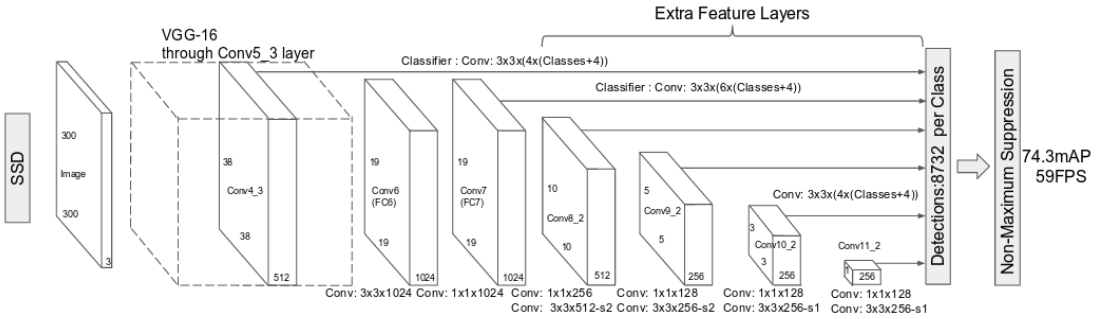

Figure 2. The SSD network architecture [18]

The SSD approach is based on a feed-forward convolutional network that produces a fixed-size collection of bounding boxes and scores for the presence of object class instances in those boxes, followed by a non-maximum suppression step to produce the final detections. Each added layer can produce a set of predictions. MultiBox is a regression technique that starts with the priors as predictions and attempts to regress closer to the ground truth bounding boxes, based on two loss functions, the confidence loss, and location loss. Confidence loss measures how confident it is of finding an object while location loss determines how far away it is from the actual box. Based on this, non-maximum suppression is used, in which thresholding of confidence loss is done and boxes are discarded.

For classification of vehicles on the road, the architecture is altered by subsampling the neurons in the network such that the number of classes included and trained is for cars, motorcycles, trucks, buses, person and background. Every object detected in the frame belongs to one of the 6 classes. Bounding boxes are drawn around them to indicate their position as well. This object detection is applied to the video of vehicles passing through the junction during the green-time. As long as the signal is green for a lane, the moving vehicles are detected by the model. This provides various advantages compared to detecting vehicles during red light, as it not only improves the accuracy of detection due to clear visibility of each vehicle in the frame, but also reduces computation needed since only one lane needs to be observed during single green time instead of observing the 3 lanes waiting in the red-signal. Show in Figure 3 Output of the SSD Object Detection Algorithm.

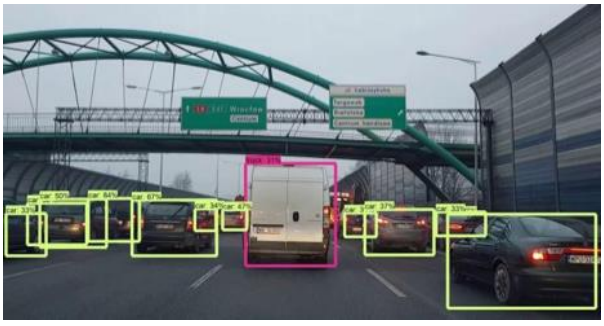

Figure 3. Output of the SSD object detection algorithm [20]

3.2. Counting vehicles

It would be very beneficial to count the number of vehicles while they are passing through the junction during green-light instead of considering a single image during red-light, therefore, it becomes

pivotal to come up with a way to count the number of vehicles of each class that passed through a scene while maintaining real-time speeds of object detection. Counting the number of objects in a still image is simple. It is commonplace now in every object detection algorithm but counting from a video feed is tricky because every frame is treated independently and might count duplicates. In the worst-case scenario, it might keep counting duplicates until the object is out of the frame. The following method is an attempt at alleviating this problem and obtaining an approximate ground truth count.

The frequently used terms are:

- OPT- Optimal Processing Time.
- F_i – The i^{th} frame in the video feed.
- T_i – Time taken to process the i^{th} frame.
- W_i – The contribution of the i^{th} frame.
- VehiclesCounted – Structure holding the number of vehicles of each class counted in the latest cycle.
- SideBySide – Structure describing the number of vehicles of each class capable of simultaneously crossing the intersection side by side.
- TimeTaken – Structure holding the time taken for a vehicle of each class to cross the intersection.
- TimeGiven – Structure holding the ratio of TimeTaken to SideBySide.
- PrevPredict – The green time that was allocated for the present green cycle after feedback from the previous green cycle.
- PresentApprox – The approximate time that had to be given to the present cycle, considering the vehicles that passed.
- NextPredict – The prediction made for the next green cycle.
- a_{min} – The minimum acceleration of a vehicle while crossing the intersection.
- a_{max} – The maximim acceleration of a vehicle while crossing the intersection.

Considering that an object detection algorithm isn't always perfect and might detect non-existent objects in a scene, an intuitive approach is to consider that an object is actually present in the scene only if it persists through some number of frames. Also taking into account the unstable nature of hardware that performs object detection, every frame takes a different amount of time to process depending on conditions at that instant, the number of objects in the scene, etc. This might lead to cases where the object is present in the scene for the required amount of time but is present only in a few frames because only few have been captured and processed. Locking the frame rate won't help either as processing time might end up higher than the defined time interval between frame capture. To solve this, there had to be some kind of a "contribution" parameter or "weight" to every frame that was processed based on which a decision can be made whether to take the vehicles in this frame seriously or not. Ideally, a frame is captured at intervals such that no vehicle is present in two consecutive frames and that all vehicles passing through a scene exist in exactly one frame. This interval is called "Optimal Processing Time" (OPT, the time it takes to process a frame and then fetch the next one from the camera).

Obtaining OPT is itself a matter of concern as it is not strictly defined neither can it be measured accurately. This has much to do with large-scale data acquisition and data analysis procedures to figure out how much vehicles accelerate while crossing junctions and if it is related to traffic density itself, how this trend changes across different densities of traffic. The size of the junction also comes into play as the OPT increases as the size does. The size of the junction can be easily calculated based on how much the camera can see in the scene. The acceleration of vehicles is bound to be a data-science problem that needs huge amounts of observations. Show in Figure 4 Illustrating the definition of OPT.

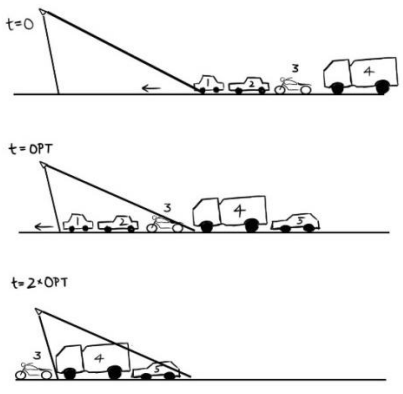

Figure 4. Illustrating the definition of OPT

Considering during OPT, n number of frames were captured and each frame F_i took time T_i to process. Therefore,

$$OPT = \sum_{i=1}^{n} T_i \quad (1)$$

If a frame Fi takes longer to process, the immediate next frame F_{i+1} might be missing out on capturing the vehicle in the scene. So, to compensate for this, F_i should have a higher contribution while considering the vehicles in it. Similarly, if F_i is processed too quickly, there might be more occurrences of the vehicle in the scene. This is compensated by giving a lower contribution to F_i. It is observed that the "weight" W_i of a frame F_i is directly proportional to the time T_i taken to process it.

$$W_i \propto T_i \quad (2)$$

The initial assumption is that the vehicles appear in the scene only once during OPT, then the weight W_i of a given frame is defined by:

$$W_i = \frac{T_i}{OPT} \quad (3)$$

This derivation shifts the requirement from "A vehicle should be present in a specified number of frames captured by the camera" to "A vehicle should be present in the camera's view for a specific interval of time". This ties the calculations to a real-world value free from ambiguity, instead of tying it to the always-changing value of frame-rate. The "contribution" of a frame throughout OPT can be visualized as shown in Figure 5.

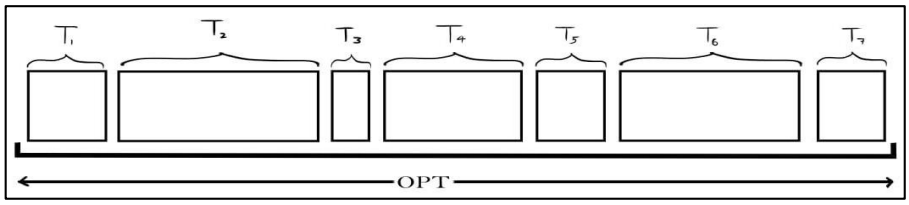

Figure 5. Illustrating the contribution of each frame in a span of OPT

This means, that if the vehicle did appear in the camera's view for as long as OPT and was eventually captured in all the frames that were taken across the time span of OPT, it will ultimately end up being counted as one vehicle, which is the desired outcome. Now, to decide whether a detected vehicle that is a legitimate one, the vehicles can be counted by considering the total number of vehicles of each class in a frame and multiplying them by W_i, i.e. weight of the frame. This is essentially the contribution of all vehicles in a frame to be accepted as a legitimate vehicle as time goes on. Continuing this until any desired time, it is possible to obtain an approximate count of the vehicles. For example, if '$Cars_t$' is the number of cars passing through
the scene in time t, and '$Cars_i$' is the number of cars in a frame F_i, and n frames are captured in time t,

$$Cars_t = \sum_{i=1}^{n} (Cars_i * W_i) \quad (2)$$

Similarly, it is possible to get the approximate counts of every vehicle class in time t.

Considering the actual traffic junction, this time t, is the time given for the vehicles to pass (green-time). Applying the above formula for the time period of the allocated green-time, the total number of vehicles of each class that pass through the junction is obtained. A point to keep in mind is that the camera must have the junction and nothing else in its view. If other roads, or the opposite lanes are captured by the camera, this might lead to discrepancies in obtaining the exact count of the vehicles.

3.3. Predicting green time

The data obtained can be visualized as follows.

```
VehiclesCounted {
   Cars = 12.78;
   2-wheelers = 19.54;
```

```
    Trucks = 3.16;
    Buses = 1.62;
    Bicycle = 5.31;
}
```

This shows the approximate number of vehicles of each class at an assumed time. An array defining the amount of time that every vehicle class needs to cross the junction will be maintained. This again depends on the size of the junction under consideration and the size of the road as multiple vehicles of the same or different class can cross the junction side by side as shown in Figure 6, and the average acceleration with which the vehicles start crossing the junction.

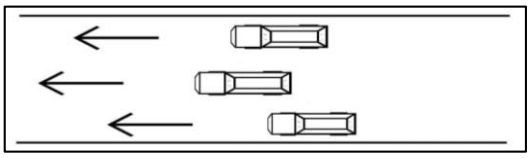

Figure 6. Top View of vehicles crossing an intersection simultaneously

Obtaining the average acceleration can be thought of as a classic data science problem involving large-scale data-acquisition and analysis, and the rest of the parameters will be specific to every junction. It will be a one-time entry of parameters and good to go unless the road width is changed or the junction is completely reconstructed. Assuming the road under consideration is wide enough to occupy the following number of vehicles of the same class side by side,

```
SideBySide {
    Cars = 3;
    2-wheelers = 6;
    Trucks = 2;
    Buses = 2;
    Bicycle = 7;
}
```

Maintaining another structure (arbitrary values for now) defining how much time a single vehicle of a class takes to cross an empty junction (in seconds),

```
TimeTaken {
    Cars = 4.67;
    2-wheelers = 3.5;
    Trucks = 6.83;
    Buses = 6.29;
    Bicycle = 5.16;
}
```

The structure TimeTaken is also a result of large-scale data gathering and data analysis procedures. Once all the above-mentioned data is available, the time to be given to each vehicle to clear the junction can be decided upon.
The TimeGiven for each vehicle class will have to be:

$$TimeGiven[Cars] = \frac{TimeTaken[Cars]}{SideBySide[Cars]} \quad (5)$$

This is to account for a realistic traffic flow scenario. In a road wide enough to hold 3 cars, 3 cars can cross the junction in the same time as it takes for a single car to cross a junction if they are all side by side. So, making this tweak is important while predicting the green-time for the next cycle. Consider a real-life scenario where some value of green-time for the present cycle (PrevPredict) is previously predicted. Now, during a time interval of length 'PrevPredict', the number of vehicles that did cross the junction are stored in 'VehiclesCounted'. The prediction about PrevPredict might have been wrong, or the traffic flow

might be changing, either way, it's important to know how much time should actually have been given to the number of vehicles that were just counted in the present cycle and stored in VehiclesCounted. This data will serve as the "ground-truth" value, like in all other self-correcting systems. This can end up being the feedback that's needed. Therefore, PresentApprox is calculated as:

$$PresentApprox = \sum_{VehicleClass=Cars}^{Bicycle} (VehiclesCounted[VehicleClass] * TimeGiven[VehicleClass]) \quad (6)$$

The value of PresentApprox might be greater or lesser than PrevPredict, nonetheless, it is used as a feedback mechanism to improve our prediction's accuracy for the green-time to be given in the next cycle (NextPredict). Intuitively, NextPredict will have to take the following two values into consideration:

1. The prediction made previously for the present cycle (PrevPredict)
2. The time that had to be given to the number of vehicles that actually crossed the junction in the present cycle (PresentApprox)

Using Weighted Moving Average, NextPredict can be calculated as:

$$NextPredict = (\alpha * PresentApprox) + ((1 - \alpha) * PrevPredict) \quad (7)$$

where α is the truth value given to our observations during the present cycle. The higher the value of α, the faster the system rectifies itself but increasing α also dismisses the contribution of PrevPredict, so conventionally, the value of α is taken to be 0.5. After the value of NextPredict is calculated, the green-time given in the next cycle will be the value of NextPredict and when the next cycle starts, the value of PrevPredict will be made equal to NextPredict, hence maintaining the contribution of our predictions throughout consecutive cycles. As the assumption is that the density of the traffic changes gradually spanning multiple cycles throughout the day, it can be visualized how this approach would always provide appropriate green-time predictions. Let's consider a hypothetical time-varying data simulating the rise in traffic density up until a rush hour and then back down again.

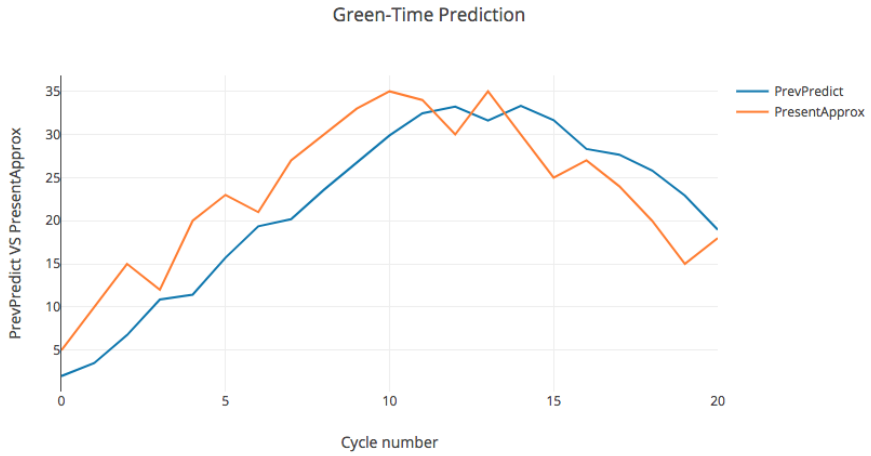

Figure 7. Variation of PrevPredict and PresentApprox illustrating accuracy and self-correcting nature

A typical rise and fall in traffic density during rush hour would span around 50 cycles, but for the sake of testing, a hypothetical less gradual change has been considered. The graph in Figure 7 shows how the prediction corrects itself every cycle bit by bit and manages to keep up with the increasing number of vehicles during rush hour. It also evens out when the traffic density saturates and again reduces as the rush hour comes to an end. Hence, this approach of looking at traffic light automation would result in efficiently utilizing time and cut down a lot of waiting time at signals. This model can easily be scaled up to a 4-road intersection, and even though this model treats every road at an intersection independently, due to its self-correcting nature, compounding effects of such systems used at consecutive traffic signals will also be

accounted for while predicting the next cycle's green-time. In this way, choking of traffic across multiple traffic lights can also be alleviated.

3.4. Deciding Hyper-Parameter Values

The SSD model has been trained on the MS-COCO (Common objects in context) dataset [21], and then the weights of the neural network have been subsampled to fit the 6 object classes (and deleting other classes in the last layer) that we're interested in – Car, truck, person, bicycle, motorcycle, bus. The value of OPT, which is dependent on the vehicle's acceleration and velocity profiles at traffic intersections was decided based on previous research in this area [22-24]. In [22], the acceleration of the 1st line of cars was observed over the distance of a zebra crossing, starting from when the vehicles were at rest. The values of accelerations recorded across different models of cars although varied, followed a normal distribution with a mean of around 1.85 ms^{-2}. But this study also stated that the cars in the 3rd or 4th line had negligibly low acceleration values in the order of 0.5 ms^{-2} which makes sense if one imagines driving a car in such a position at a traffic signal. The velocity of vehicles, however, isn't ever increasing and saturates and somewhat reaches a maximum while crossing the intersection. In [19], it is found that the average acceleration of any vehicle steadily decreases with the increase in the speed of the vehicle and assuming the velocity of vehicles doesn't exceed 30 or 40 kph while crossing an intersection, it can be inferred from the research in [22-23] that the average acceleration change while crossing of different classes of vehicles are as follows.

– Car = 1.85 – 0.5 ms^{-2}
– Truck, Bus = 1.0 – 0.29ms^{-2}
– Motorcycle = 0.94 – 0.47ms^{-2}

This acceleration is not strictly a function of time or distance at a traffic intersection. Instead, it depends on factors such as other vehicles on the road, possible interference from people walking, bad drivers, even animals on the roads at times. The closest one can get to define the acceleration profile is by an assumption that it is strictly a function of distance or time. Here, the assumption is that it is a function of distance is constantly decreasing and eventually saturates near zero. This kind of a function resembles an exponential decay function, so fitting the acceleration parameters that were just decided upon,

$$a(x) = a_{max}\, e^{-\lambda x} \tag{8}$$

Where (considering X is the length of the intersection under consideration),

$$\lambda = \frac{ln\left(\frac{a_{max}}{a_{min}}\right)}{X} \tag{9}$$

This acceleration curve corresponding to this equation and a 25-meter intersection for a car is shown in Figure 8.

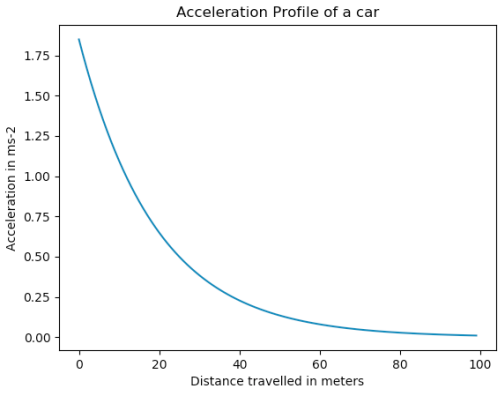

Figure 8. Acceleration profile of a car according to a(x)

This makes sense for short distances such as an intersection where the velocity function is a result of integrating the acceleration function with distance 'x'

$$\frac{1}{2}(v(x))^2 = \int a(x).dx \tag{10}$$

The velocity function obtained is:

$$v(x) = C + \sqrt{\frac{2a_{max}(1-e^{-\lambda x})}{\lambda}} \quad (11)$$

Plotting v(x) against x,
The graph shown in Figure 9 would make sense if one relates self-driving through an intersection and checking the velocity of the car as one crosses and at the very end of it. Although this graph doesn't hold true for long distances, only short distances are relevant to this problem. Time t, can be obtained as a function of distance x by integrating as,

$$t(x) = \int \frac{1}{v(x)} . dx \quad (12)$$

$$t(x) = \frac{ln\left(\sqrt{1-e^{\lambda x}}+1\right)-ln\left(\left|\sqrt{1-e^{\lambda x}}-1\right|\right)}{\sqrt{2a_{max}\lambda}} + C \quad (13)$$

as t(x) is zero when x is zero, a definite equation for t(x) is obtained and by passing the length of the intersection 'X' into t(x), the time that a certain vehicle type takes to cross the intersection is obtained which is nothing but the OPT that is needed. This value can also be interpreted as the 'TimeTaken' for that specific class of vehicles from which 'TimeGiven' can also be derived.

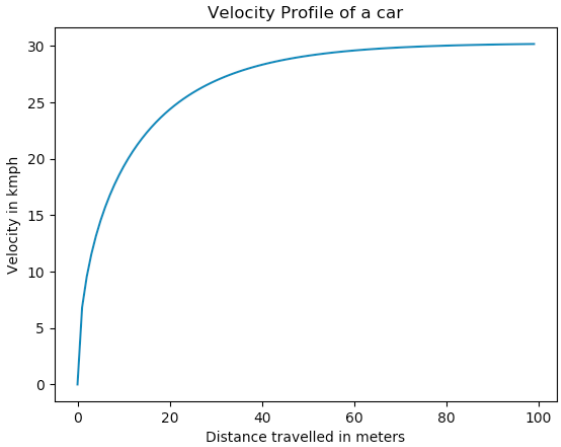

Figure 9. Velocity profile of a car according to v(x)

4. RESULT AND ANALYSIS

To apply the findings from our study outlined above, a traffic footage that is accessible to the public on YouTube was used [25]. The specific requirements for the camera's position, angle, etc. are not met perfectly in any publicly available video footage so the video is cropped and modified to simulate a nearly feasible input to the algorithm. The base video [25] chosen was such that it offered a front-top view towards incoming vehicles which yields the best results for an object detection algorithm with least overlaps. The farthest point in the video is too far away for the object detection system to detect anything that small as shown in Figure 10, and there is another lane of traffic in the frame of the camera which will produce inaccurate results, the video was cropped to a specific part as shown in Figure 11.

Figure 10. Base video footage frame

Figure 11. Cropped Video Footage

With the cropped frame of capture, vehicles don't start with zero velocity, neither with the minimum acceleration. Feasible assumptions had to be made about their speed when they enter the frame and the distance they travel in the video to accommodate for the imperfect input footage. Assuming an initial velocity and range of acceleration values, the value of the constant 'C' in the equation for v(x) and the values of a_{max} and a_{min} are set. The vehicles were assumed to have an initial velocity of 20kmph when they entered the frame with an acceleration varying from 1.65 to 1.85 ms^{-2}, the distance covered in the frame was taken to be approximately 16 meters. Calculating OPT according to these approximate hyper-parameters, the following results were obtained when run on a consumer grade NVidia GTX 1060 graphics card.

Table 1. Car count obtained from Nvidia GTX 1060 Graphics Card

Trial number	1	2	3	4	5
Ground Truth (no. of cars)	54	54	54	54	54
Calculated	37.8	37.9	38.6	38.7	39.5

These results demonstrate that the model achieved an accuracy score of 72% based on approximate hyper parameters. Superior results can be achieved while considering accurate parameters for the velocity and acceleration values of various vehicles. Given that the SSD gives an accuracy of 74%, a 72% obtained as a result of using SSD is a very accurate system.

4.1. Statistical inference

Based on the results demonstrated, it can statistically be proven to infer the same. The paper claims to count 72% of all the cars crossing the junction in a confidence interval of 85% i.e. α=0.15. Performing a t-test to prove the claim, from the given data, t=0.97. For a confidence interval of 85%, t=1.19. Therefore, the hypothesis is comfortably accepted, and it is validated to claim that 72% of cars will be detected.

4.2. Limitation and scope

This approach suffers from the limitations associated with the lack of large-scale traffic data about vehicle speed profiles at intersections. Further research in this field will immensely help models such as the one presented in this paper attain higher accuracy values and ultimately help solve the primary traffic congestion problem. It remains unclear whether this model with its specific camera placement requirements is implementable and feasible in a practical implementation. At this stage of understanding, the proposed model, if implemented successfully, provides accurate results for vehicle count and classification, given large enough varied data statistics on vehicle acceleration profiles at intersections.

5. CONCLUSION

A system for detecting, classifying and counting vehicles in traffic footage has been proposed. Previous research about the acceleration data on various vehicles has been used to build mathematical models to reasonably approximate the hyper-parameters that aid in the process of counting of vehicles from a video stream. A predictive model for calculating green time has also been proposed which is based on weighted moving average. Given that the accuracy of SSD Multibox detector is 74%, the accuracy of counting obtained to be 72% is nearly perfect. This approach will utilize no on-road resources like sensors or measuring equipment and overall needs a single camera to fulfill all the requirements which are to be expected of a traffic monitoring and a dynamically-controlled traffic light system.

5.1. Future Scope

Although the above calculations have provided somewhat accurate results for the values of `OPT' and `TimeTaken', due to the limits of the object detection algorithm and the possibility of detection of false positives, specific guidelines must be followed for the placement and the orientation of the camera at an intersection. This specific kind of traffic footage isn't available to the public as of now. As this approach requires very specific camera placements at traffic junctions, future work would involve gathering the required kind of traffic footage with proper angles and coverage coupled with more research and data gathering in vehicle speed and acceleration profiles to test the accuracy of this model systematically for providing formal proof of the accuracy and reliability of this approach.

ACKNOWLEDGEMENTS

We acknowledge the support of NVIDIA Corporation for donating Jetson TX1 and Jetson TX2 kits used to carry out this project.

REFERENCES

[1] Economic Times, "Number of vehicles in Bengaluru rises by a mind-boggling 6099% in 40 years.," https://economictimes.indiatimes.com/news/politics-and-nation/number-of-vehicles-in-bengaluru-rises-by-a-mind-boggling-6099-in-40-years/articleshow/57807459.cms, Bengaluru, 2017.

[2] Q. I. Suneera Tandon, "Traffic jams in just four Indian cities cost $22 billion a year," https://qz.com/india/1255427/traffic-jams-in-delhi-mumbai-bengaluru-and-kolkata-alone-cost-india-22-billion-a-year/, Delhi, 2018.

[3] Dipak K Dash, Times Of India, "Traffic congestion costs four major Indian cities 1.5 lakh crore a year," https://timesofindia.indiatimes.com/india/traffic-congestion-costs-four-major-indian-cities-rs-1-5-lakh-crore-a-year/articleshow/63918040.cms , Delhi, 2018.

[4] R. N. Rithesh, R. Vignesh and M. R. Anala, "Autonomous Traffic Signal Control using Decision Tree," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 3, pp. 1522-1529, June 2018.

[5] J. Pang, "Review of microcontroller based intelligent traffic light control.," in *2015 12th International Conference & Expo on Emerging Technologies for a Smarter World (CEWIT)*, Melville, NY, 2015.

[6] K. A. Khateeb and J. A. Johari, "Intelligent dynamic traffic light sequence using RFID," in *International Conference on Computer and Communication Engineering*, Kuala Lumpur, Malaysia, 2008.

[7] H. Chaudhuri and N. P. Raikar, "Traffic Control Management with help of State of Control Algorithm using Ultrasonic Sensors & GSM Technology," *International Research Journal of Engineering and Technology (IRJET)* , vol. 5, no. 4, April 2018.

[8] S. Sahu, D. Paul and S.Senthilmurugan, "Density Based Traffic Signal Control Using Arduino And IR Sensors," *IJNRD*, vol. 3, no. 4, April 2018.

[9] S. Shams-Ul-Haq, "AUTONOMOUS REAL TIME TRAFFIC MONITORING AND DATA ANALYSIS," in *The 2012 World Congress in Computer Science, Computer Engineering, and Applied Computing*, Las Vegas, USA, 2012.

[10] F. A. Afif, F. M. Rachmadi, A. Wibowo, W. Jatmiko, P. Mursanto and M. A. Ma'sum, "Enhanced adaptive traffic signal control system using camera sensor and embedded system," in *2011 International Symposium on Micro-NanoMechatronics and Human Science*, Nagoya, Japan, 2011.

[11] J. Chung and K. Sohn, "Image-Based Learning to Measure Traffic Density Using a Deep Convolutional Neural Network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 5,May 2018, pp. 1670 - 1675, 2017.

[12] D. Oñoro and R. J. López-Sastre, "Towards Perspective-Free Object Counting with Deep Learning," in *Computer Vision – ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VII*, Amsterdam, Netherlands, 2016, pp. 615-629.

[13] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in neural information processing systems*, vol. 1, pp. 1097-1105, 2012.

[14] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, 2014.

[15] R. Girshick, "Fast R-CNN," in *IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, 2015.

[16] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137 - 1149, 2017.

[17] J. Dai, Y. Li, K. He and J. Sun, R-FCN: Object Detection via Region-based Fully Convolutional Networks, NIPS, 2016.

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *Computer Vision – ECCV 2016*, 2016.

[19] pierluigiferrari, "A Keras port of Single Shot MultiBox Detector," available at: https://github.com/pierluigiferrari/ssd_keras.

[20] K. Majek, "SSD Mobilenet Object detection FullHD S8#001," available online at: https://www.youtube.com/watch?v=7p2XL8wApfo, 19 Feb 2018.
[21] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick and P. Dollár, "Microsoft COCO: Common Objects in Context," in *European conference on computer vision*, Cham, Germany, 2014.
[22] V. Bogdanović, N. Ruškić, Z. Papić and M. Simeunović, "The Research of Vehicle Acceleration at Signalized Intersections," *Traffic & Transportation*, vol. 25, no. 1, 2013.
[23] P.S.Bokarea and A.K.Mauryab, "Acceleration-Deceleration Behaviour of Various Vehicle Types," *Transportation Research Procedia*, vol. 25, pp. 4733-4749, 2017.
[24] G. Long, "Acceleration Characteristics of Starting Vehicles," *Transportation Research Record Journal of the Transportation Research Board*, vol. 1737, no. 8, pp. 58-70, 2000.
[25] Cisco C, "2.1M Full HD 1080P CCTV Footage," *available online at: https://youtu.be/WxgtahHmhiw*.

BIOGRAPHIES OF AUTHORS

Mr. Parinith R Iyer is an Undergraduate student in his 3rd year studying in the CSE department at R V College of Engineering. He has publications in IEEE funded magazine in SJCE, Mysore. He has carried out 4 UG projects and 1 funded Industry Project while at R V College of Engineering.

Mr. Shrutheesh Raman Iyer is an Undergraduate student in his 3rd year studying in the CSE department at R V College of Engineering. He heads the technical drone club Jatayu at RVCE. He has carried out 3 UG projects and 1 funded Industry Project while at R V College of Engineering.

Mr. Raghavendran Ramesh is an Undergraduate student in his 3rd year studying in the CSE department at R V College of Engineering. He has publications in IEEE funded magazine in SJCE, Mysore. He has carried out 3 UG projects and 1 funded Industry Project while at R V College of Engineering.

Dr. Anala M R is now an associate professor in the CSE department at R V College of Engineering. Her academic qualification includes: BE, MSc(IT), M.Tech, Ph.D. She was also awarded the Second Rank in M.Tech from VTU, Belgaum. She has guided over 30 UG and PG projects and 5 funded research projects. She has 25 publications in national and international journals and conferences.

Dr. K N Subramanya is the Principal and Professor, Department of Industrial Engineering and Management at R.V. College of Engineering. His academic qualification includes: Ph.D. in Supply Chain Management from Coimbatore, MBA with HR specialization (5th Rank), M.Tech. in Industrial Management (IITM, Chennai) and B.E in Industrial and Production Engineering (Bangalore University). The professional expertise include Lean Manufacturing System, Operations Management, Intelligent Systems in Supply Chain and Logistics Management, e-Enterprise solutions, Simulation Modeling and Analysis, Systems Engineering, Operations Research, Probability & Statistics, Applied Ergonomics and Business Process Management.