Recognition of a traffic situation in real-time

Information Technology
Automation Lab Project

Chandrima Biswas
cbiswas@stud.fra-uas.de
Matriculation number: 1248515

Pretom Kumar Saha
saha@stud.fra-uas.de
Matriculation number: 1276545

***Abstract*—Traffic situation recognition in real time plays an important role to reduce road congestion and accidents. There are many existing systems which can deal with traffic situation by estimating of traffic with the assistance of position of traffic, number of traffic, average speed of traffic etc. In this article we try to build a model which could recognize the traffic situation as human could do by using artificial intelligence. Deep Learning is one of the branches of AI, which is used in computer vision, speech recognition, speech recognition, natural language processing, audio recognition and many other fields. Object detection is a combination of Computer vision and Image Processing. To detect the object in the traffic and make decision You Only Look Once (YOLO) algorithm and ImageAI are given more accuracy. This experimental analysis gives some meaningful conclusion in recognition of a traffic situation in real-time.**

Keywords—Traffic density, Machine Learning, Deep neural network, Computer vision, Object detection.

I. INTRODUCTION

The urbanization has a significant impact on modern life. It provides a lot of benefits to make the life easier but on the other side of coin, it has exponential growth of traffic in urban areas. These causes many problems such as traffic jams, environmental pollution, problem in public health, reduce the productive labor time and also economic losses.

The traffic surveillance is an integral part of the intelligent transportation system network. Providing a reliable, real-time traffic monitoring has a major influence on the efficiency and safety of highways and roadways[7]. Real-time traffic monitoring by using CCTV footages can be used for estimating the traffic density which could solve traffic congestion problem, reduce accidents and thus helping the transport management to face critical aspects of the mobility[3]. At the same time, this information can also be used to plan the mid and long-term roads mobility strategy [3]. This idea is the adequate example of a smart city application which could also have strong impact on citizens' security.

The idea of controlling the automated traffic management system in real time has attracted many researchers to do research in this field with the target of creating automated tool which can estimate the traffic congestion, observe the traffic, also tracking the traffic. Analysis of traffic conditions showed that there are many fluctuations in the quantity of the vehicles approaching to a crossroad for the same period of time [4]. As a result, an automatic real-time traffic situation recognition has been emerging as a new field in image processing, pattern recognition, computer systems, artificial intelligence and neural networks.

The present study demonstrates a simple approach to recognize the traffic situation by counting the vehicles and comparing the changes of the vehicles on a road segment based on video images with no hand-crafted feature engineering. Certain number of studies have dealt collectively with crowd counting using regression approaches [11-13]. Quite a few studies use a regression approach to count vehicles to identify the traffic density on a road. Thus, the success in counting vehicles simply by computer vision and machine learning would be a significant breakthrough in advancing existing traffic control and management [10]. In some previous studies for crowd counting, feed-forward neural network was deployed. On the other hand, the present study is focused on deep convolutional neural network to estimate on counting the vehicles from a video footage.

We try to develop a system which use a video footage of a street and automatically recognize or give a decision of traffic situation . Even if traffic is flowing very slowly, system could recognize the actual traffic situation efficiently. With the help of an intelligent system, the system could able to detect the number of vehicles, the change of vehicles in every second and ultimately adapt the decision for traffic situation accordingly to get the optimal outcome. We implement this system that use a deep neural network to decide on which time of the given video footage has congested street or free street. We also deploy this model with OpenCV in a Computer with an attached camera. Thus, the OpenCV will capture images of the street at brief intervals and will use the Deep neural network to estimate traffic density and then relay this information to the computer, where further processing can be done.

II. OVERVIEW

Real time computer vision-based traffic signal control has drawn recently the interest of many researchers due to the frequent traffic jams at major junctions and its resulting wastage of time. Depending on data created by expensive sensors could be replaced by utilizing easily accessible video cameras in an effective way for adequate traffic congestion system.

A. Definitions and Problem Description:

We try to solve the problem of deciding whether there is traffic or not , just as simple as a human would do. By looking at an image and seeing whether there are lots of vehicles and they don't have the change of position. If the answer is yes, then there is heavy traffic, if not, then low traffic. By training with the help of Machine Learning techniques, the computer could to do the same by itself. We chose deep neural networks to use as the machine learning technique. Thus, given a video sequence, the task of computer vision-based traffic situation recognition is:

1) analyze video frame sequences; 2) estimate traffic congestion and 3) predict the traffic situation.

The problems of common traffic Flow control system are mentioned below:

Heavy Traffic:
The number of exponentially increased vehicles on road causes heavy traffic congestion which is also increased in major cities. The problem for heavy traffic usually occurs in some major junctions before and after the office hour. The main effect of the problem is the wasting of time of the people on the road.

Low traffic, but still need to wait:
At certain junctions, often even if there is no traffic, people have to wait. Because the traffic light remains red for the preset time period, the road users should wait until the light turn to green. This problem can be solved by developing a system which detects traffic flow on each road and set timings of signals accordingly.

B. Proposed System (Density Estimation and situation prediction):

The objective of proposed system is to predict the traffic situation system using image processing, measuring traffic load. The prediction will be calculated automatically depending on traffic load for each time change. System will have computer vision with the help of digital camera. This single image of street footage will be processed using image processing techniques to estimate traffic load. Estimated traffic load on particular road will be used to calculate for identification of traffic situation based on experimental results. If the number of vehicles and the change of vehicles will exceed the threshold, then the system will recognize that time would have a heavy traffic and if the variables will not exceed the threshold for low traffic then the system will recognize the situation as low traffic. Python programming environment will be used for simulating and developing the proposed system.

III. LITERATURE SURVEY

Many systems like Google maps use GPS based location broadcast. They work as follows; with help of GPS every car broadcasts their location to a server. Then the server collects information from the cars and measures the number of cars in the same location. It retransmits the decision to every car, which will alert the driver about the traffic situation in the area.
However, there are lots of research going on using intelligent system which could predict the traffic situation. These systems, as opposed to statistical analysis, employ the use of artificial intelligence to 'learn' to detect traffic and predict traffic. The system is trained with a lot of historical traffic data and the trained system will be able to predict the traffic situation in real time. The standard practices in this method are using trained classifiers to decide which class a data point (In this case, a state of traffic) belongs to. A few of the common classifiers used are Feed Forward Neural Networks, Hidden Markov Models, and Support Vector Machines.

vision-based traffic object detection is divided into two methods, they are: traditional machine vision methods and complex deep learning methods. Traditional machine vision methods use the motion of a vehicle to separate it from a fixed background image[2]. This can be done by using three methods: 1) background subtraction, 2) continuous video frame difference and 3) optical flow. The continuous video frame difference method works by calculating a variance according to the pixel values of consecutive two or three video frames. Moreover, the moving foreground region is separated by the threshold [14]. By using this method and suppressing noise, the stopping of the vehicle can also be detected [16]. When the background image in the video is fixed, the background information is used to establish the background model [16]. Then, each frame image is compared with the background model, and the moving object can also be segmented. The optical flow method works as the detection of motion of region in the video. The generated optical flow field represents each pixel's direction of motion and pixel speed [15]. The common methods for vehicle detection using vehicle features such as the Scale Invariant Feature Transform (SIFT) and Speeded Up Robust Features (SURF) methods have been widely used. For example, 3D models have been used to complete vehicle detection and classification tasks [17]. Using the correlation curves of 3D ridges on the outer surface of the vehicle [18], the vehicles are divided into three categories: cars, buses, and minibuses.
The use of deep convolutional networks has achieved amazing success in the field of vehicle object detection. deep convolutional networks have very strong ability to learn the image features and can perform multiple tasks such as classification and bounding box regression [19]. The detection method in deep learning can be divided into two types, they are : 1) The two-stage method and 2) the one-stage method. The two-stages method use many algorithms to generate a candidate box and then classifies the object by a convolutional neural network. On the other hand, the one-stage method directly converts the positioning problem of the object bounding box into a regression problem for processing without generating any candidate box[2]. Among the one-stage methods, the most important are the Single Shot Multibox Detector (SSD) and You Only Look Once (YOLO) frameworks. The YOLO works by dividing the image into certain number of grids. Each grid is responsible for predicting objects whose center points are within the grid [2]. YOLO v2 have an additional Batch Normalization layer, which makes the network normalize the input of each layer and accelerate the network convergence speed[2]. For every ten batches,YOLOv2 uses a multi-scale training method to randomly select a new image size. Yolo v3 uses logistic regression to categorize the object. Moreover, logistic regression is used to regress the box confidence to determine if the IOU of the a priori box and the actual box is greater than 0.5 [2]. If more than one priority box satisfies the condition, only the largest prior box of the IOU is taken. YOLO v3 utilizes three different scales to predict the object of the image for final object prediction.
The traditional machine has advantage of faster speed for object detection but when image changes in brightness it does not make an accurate result. Deep convolutional neural network works better in object detection however, it is sensitive to scale changes in object detection [21,20].

IV. SYSTEM DESIGN

This section describes the main structure of the traffic situation recognition in real time by vehicle detection ,counting the vehicles and the make the decision for traffic situation system. In this system, we consider three traffic situations : Heavy traffic ,Medium traffic and Low traffic. First, the video data of the traffic scene are entered. The YOLOv3 deep learning object detection method is used to detect the object in the traffic scene and some other methods used to categorize the traffic which we have discussed in the next section.

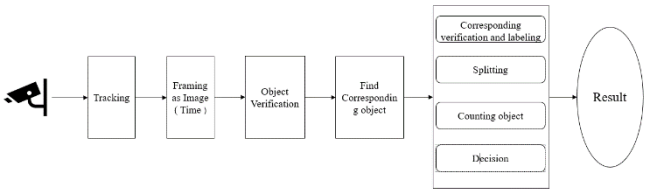


Figure 1: System architecture.

Video Datasets: The data we work with is a raw set of video footage and the data was collected from https://github.com/OlafenwaMoses/IntelliP/raw/master/traffic-mini.mp4. The length of those videos is six second. The traffic density value is calculated from the video footage of a street. . Every traffic density value depends upon some factors such as the time of estimation, the estimated current traffic density, and the predicted traffic density in a certain interval.

Fragmented into image frame: The whole video is fragmented into numbers of frame of image. In one second of time interval, the video is fragmented into 40 image frames.

Object Verification: The algorithm which we have used in that system is already trained by training data sets. Based on that training datasets, the algorithm returns the recognition of the object.

*Find Corresponding objec*t: In previous section, we discussed about object recognition done by the algorithm that is used in the system. In this stage, those object data is identified the corresponding object such as car, bus etc.

Labeling: After identification, the object is then labelled based on the identification of the object.

Splitting: All object in each frame has been divided into five categories. Those are bus, car, bicycle, motorbike and people.

Counting: In this level, the number vehicle object is calculated to make the decision.

Decision: This is the final part of the system. Here, based on the result of previous level decision is made. If a certain interval of time, the number of vehicles is more than 70% , then the decision will be "Heavy Traffic". If the number of vehicles is less than 70% but more than 30% ,then the decision will be "Normal Traffic". And for the number of vehicles is less than 30% the decision will be "Low Traffic"

Result: Based on decision, the current situation of the traffic will be displayed using GUI as the result.

V. TOOLS AND PACKAGES:

Python: Python is a well-known object-oriented programing language which have the capabilities of high-level programming language as well as most popular language for Machine learning and data science. Python offers many effective packages which makes the object detection implementation very easier for programmers. So, to develop the system, we have used Python, as the programming language by using spyder as a IDE, anaconda navigator which allows to launch applications and easily manage conda packages, environments, and channels, Yolo algorithm to detect the object in real time.

To implement our propose system we used different python packages. Those packages are very popular and very effective in machine learning, image processing and on the others field. In the following section we are going to describe about the packages in details:

Packages:

ImageAI: **ImageAI** is a python library built to empower developers, reseachers and students to build applications and systems with self-contained Deep Learning and Computer Vision capabilities using simple and few lines of code[22]. It supports a list of state-of-the-art Machine Learning algorithms for image prediction, custom image prediction, object detection, video detection, video object tracking and image predictions trainings[22]. It provides classes and methods for training YOLOv3 object detection models on custom dataset. This means we can train a model to detect literally any object of interest by providing the images, the annotations and training with ImageAI.

Tensorflow: An end-to-end platform for machine learning and is one of the most widely used frameworks for deep learning[23]. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets developers easily build and deploy ML-powered applications[23].

Pillow: The Python Imaging Library (PIL**)** adds image processing capabilities to Python interpreter. The benefits of the library is to support many file formats, and provides powerful image processing and graphics capabilities.

Keras: Keras is a dynamic and fexible free open source Python library for developing and evaluating deep learning models. It wraps the efficient numerical computation libraries TensorFlow and allows you to define and train neural network models in just a few lines of code[23].

Numpy: NumPy is the fundamental package in Python which used for scientific computing . It provides some features[24]:

- a powerful N-dimensional array object.
- sophisticated (broadcasting) functions.
- tools for integrating C/C++ and Fortran code.
- useful linear algebra, Fourier transform, and random number capabilities.

SciPy: The SciPy library depends on NumPy, which provides convenient and fast N-dimensional array manipulation. The

SciPy library is built to work with NumPy arrays and provides many user-friendly and efficient numerical routines such as routines for numerical integration and optimization[25]. Together, they run on all popular operating systems, are quick to install, and are free of charge. NumPy and SciPy are easy to use, but powerful enough to be depended upon by some of the world's leading scientists and engineers[25].

OpenCV: OpenCV-Python is the powerful Python API of OpenCV. A library of programming functions which mainly use for real-time computer vision.

Matplotlib: Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack[25]. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc[25].

H5py: The h5py package provides both a high- and low-level interface to the HDF5 library from Python. The low-level interface is intended to be a complete wrapping of the HDF5 API, while the high-level component supports access to HDF5 files, datasets and groups using established Python and NumPy concepts[25].A strong emphasis on automatic conversion between Python (Numpy) datatypes and data structures and their HDF5 equivalents vastly simplifies the process of reading and writing data from Python[25].

VI. IMPLEMENTATION

Vision, in which visually perceptible objects that are in pictures of recordings can be identified, localized, and recognized by computers. An picture may be a single outline that captures a single-static instance of a normally happening event.

On the other hand, a video contains numerous instances of static pictures shown in one moment, actuating the impact of seeing a actually happening event.

Technically, a single static picture in a video is called a video frame. In most recordings, the number of frames in one moment of the video ranges between 20 to 32, and this esteem is called the frames-per-second (fps). In this following section we discuss about our implementation process.

A. Object Frameing

1) Frame differencing

Frames are captured from camera at regular intervals of time. Contrast is evaluated from the continuous frames. Utilizing "frames-per-second" to form frame of pictures from video. We use 40 frame per second Fig 1. Because we observe that as much frames-per-second we used, we get more accurate result.

```
video_path = detector.detectObjectsFromVideo(input_file_path=filename,
                                output_file_path=os.path.join(execution_path, "traffic_mini_detected_1"),
                                frames_per_second=40,
                                #per_second_function=forSeconds,
                                #per_minute_function=forMinute,
                                video_complete_function=forFull,
                                minimum_percentage_probability=30,
                                return_detected_frame=False,
                                log_progress=True)
    #
```

Figure 2: object detection from video function

2) Create Object Box

Our framework separates the input picture into an S × S grid. If the center of an object falls into a framework cell, that grid cell is dependable for identifying that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the show is that the box contains an question and also how exact it considers the box is that it predicts. Formally we characterize confidence as Pr(Object) ∗ IOU. In case no object exists in that cell, the confidence scores ought to be zero. Something else we need the confidence score to break even with the intersection over union (IOU) between the predicted box and the ground truth.

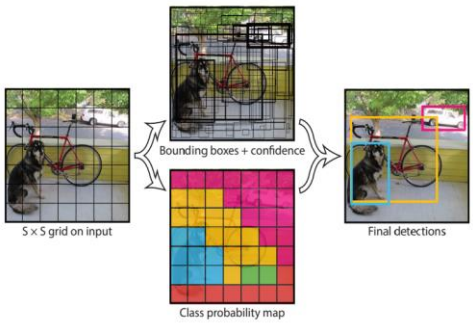


Figure 3: Divide the image into grid and predict the bounding boxes

B. Object Detection

```
def Object_detect():
    filename = filedialog.askopenfilename(initialdir = "/",title = "Select file",filetypes = (("jpeg files","*.mp4"),("all files","*.*")))
    detector = VideoObjectDetection()
    detector.setModelTypeAsYOLOv3()
    detector.setModelPath( os.path.join(execution_path , "yolo.h5"))
    detector.loadModel()

    video_path = detector.detectObjectsFromVideo(input_file_path=filename,
                                    output_file_path=os.path.join(execution_path, "traffic_mini_detected_1"),
                                    frames_per_second=40,
                                    per_second_function=forSeconds,
                                    #per_minute_function=forMinute,
                                    #video_complete_function=forFull,
                                    minimum_percentage_probability=30,
                                    return_detected_frame=False,
                                    log_progress=True)
    #
    #video_complete_function=forFull,
print(video_path)
```

Figure 4: object detection class

From the video sequence of CCTV or other cameras on the traffic signal, we try to figure out the objects. As a result, we are using a Python library called ImageAI that supports YOLOv3 Algorithm. Here's an in-depth explanation of the preceding code:

First, an instance of VideoObjectDetection class is initialized.

Set the model type to YOLOv3 and the path of the model file.

Load the model into the instance of the VideoObjectDetection class.

The detectObjectsFromVideo function is called with the following values:

- input_file_path: This refers to the file path of the video, which is used to detect the objects.
- output_file_path: This refers to the file path to which the detected video will be saved.

- frames_per_second: This refers to the number of image frames that we want the detected video to have within a second.
- minimum_percentage_probability: option to set the minimum percentage probability for nominating a detected object for output.
- return_detected_frame: option to obtain the return the last detected video frame.
- per_second_function: this parameter allows you to parse in a function you will want to execute after each second of the video is detected. If this parameter is set to a function, after every second of a video is detected, the function will be executed.

C. *Decision-making*

```
def traffic(count_arrays):
    total_percent=0
    for i in range(len(count_arrays)):
        if i<len(count_arrays)-1:
            if list(count_arrays[i])==list(count_arrays[i+1]) and (len(list(count_arrays[i]))>1 or len(list(count_arrays[i+1]))>1):
                total=0
                objects=list(count_arrays[i])
                for a in objects:
                    if not a == 'person':
                        if a in list(count_arrays[i]):
                            total=total+abs(count_arrays[i+1][a]-count_arrays[i][a])
                            print("object:", a)
                            print("total:", total)
                    if a == 'person':
                        items=len(objects)-1
                    else:
                        items=len(objects)
                percent=total/items
                percent=int(percent*100)
            else:
                percent=80
        else:
            percent=percent
        total_percent=total_percent+percent
    avg_percent=int(total_percent/len(count_arrays))
    print("avg",avg_percent)
    return avg_percent
```

Figure 5: Function that calculate traffic situation

By checking how my object has moved in different frames of the video, we can calculate the traffic situation. The conditions of the road can be calculated by verifying the object's displacement with the help of different frames taken at different interval of time. This method is actually a flaw where one is not tracking but detecting the object at different frame. By using "per_second_function()", we can get an array of dictionaries, with each dictionary corresponding to each frame in the entire video, and the keys of each dictionary are the name of the number of unique objects detected in each frame, and the key values are the number of instances of the objects found in the frame. Improved method is "detection with object count". In this method estimation of objects like car or bus in every frame. By checking it's quantity at a particular frame "i" and estimating its quantity at another frame let's say 'i+1'.Comparing the changing percentage of objects quantity in frame 'i' and 'i+1',the decision is taken by averaging the changes of object in frame in one second . If the changes are near 30% then traffic is high, and 70% then traffic is low.

D. *Some Common Difficulties*

To implement this python project, we faced some common difficulties. However, we find the solutions at the end. Here, the difficulties:

- To Install the required dependencies, we faced the problem with the version. Like tensorflow 1.15 is not support keras 2.0.
- Try to use per_minute_function, when we try to calculate the traffic situation with "detection with object count" method. But it too time consuming and does not give accurate result.
- Also decide to use per_frame_function, but it doesn't give the output of next or previous frame.
- Try to make a one GUI to take input and show output. But the data are showing in unsupervised way in the GUI.

VII. Result And User Interface

We are going to show the result of our project work on a set of real images and the images with object detection box.

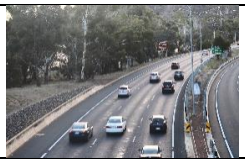
Real Images	Images with object detection box
	
	
	

Figure 6: Detection results.

As a result, it also shows the number of vehicles in the frame per second with percentage probability and the size of the bounding box, the count of the different type same of vehicles in per frame, and the average total number of different vehicle in per second Fig 2.

```
Array for the outputs of each frame
 {'name': 'car', 'percentage_probability': 94.07886266708374, 'box_points': (890, 337, 951, 386)},
 {'name': 'car', 'percentage_probability': 95.1903223991394, 'box_points': (737, 292, 813, 336)},
 {'name': 'car', 'percentage_probability': 96.21223211288452, 'box_points': (854, 292, 909, 337)},
 {'name': 'car', 'percentage_probability': 96.42298817634583, 'box_points': (837, 385, 904, 443)}]

Array for output count for unique objects in each frame :
 { 'car': 2},
 { 'car': 2},
 { 'car': 2},
 { 'car': 2},
 { 'car': 2},

Output average count for unique objects in the last second:  {'truck': 1.0, 'car': 2.025}
```

Figure 7: Output of project execution as array

The confusion matrix is utilized mean average accuracy. Accuracy recall curve is computed for a given class. Recall is characterized as the extent of all positive cases ranked above a given rank. Accuracy is the extent of all cases over that rank which are from the positive class. The average accuracy summarizes the shape of the accuracy-recall bend and is defined as the mean accuracy at a set of n similarly dispersed recall levels. Thus, to get a high score, high accuracy is wanted at all levels of recall. This measure is better than range beneath curve since it gives significance to the sensitivity.

The detections were assigned to ground truth objects and judged to be true/false positives by measuring bounding box overlap. To be considered a redress detection, the range of overlap between the predicted bounding box and ground truth bounding box must exceed a threshold. The yield of the detections allotted to ground truth objects fulfilling the overlap model were positioned in arrange of (diminishing) certainty output. Multiple detections of the same protest in an picture were considered false detections, i.e. 5 location of a single object numbered as 1 true positive and 4 false positives. In case no prediction is made for an image then it is considered a false negative.

The normal accuracy for all the question categories are detailed in Fig. 8.

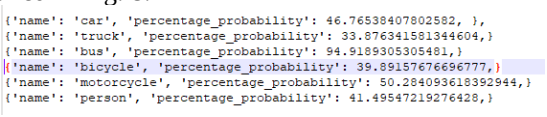


Figure 8: Probability of a certain time

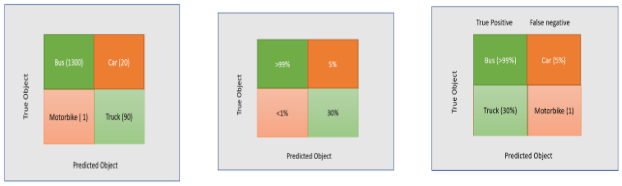


Figure 9: confusion matrix

For the most part, the objective of user interface design is to create a user interface which makes it simple, effective, and agreeable (user-friendly) to function a machine within the way which produces the required result. This generally implies that the operator ought to give negligible input to attain the required output, additionally that the machine minimizes undesired outputs to the human. Here is the user interface:

Figure 10: User Interface to select file

Fig. 10 is the main window to select a file or close the program. When the project will run in every second it will show the traffic situation with 3 different color window Fig. 10.

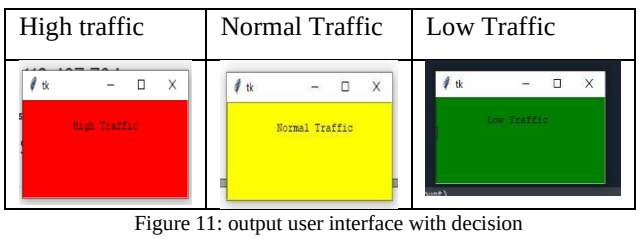


Figure 11: output user interface with decision

VIII. CONCLUSION

The present study was an exhibit of a novel approach to recognize the situation of traffic on a street section in order to precisely evaluate traffic density at an aggregate level for traffic control and management. The smart transportation system has a powerful impact on smart city whereas the traditional systems are not automated to make the decision on the current traffic situation to minimize the traffic congestion efficiently. Hence, a smart traffic congestion control system can be used to manage the automated traffic signal timer by using machine learning technique.

The main goal behind implementing the system is to analyze the traffic and build a model which could be able to participate to reduce traffic congestion to some extent in a specific area. The significance to use machine learning in the system is being performed on random data. The machine learning algorithm will help in training the model for new data which basically means for learning[1].

A pipeline to develop state-of-the-art traffic situation recognition systems from videos are done by three main steps: object detection, feature extraction and classification. A preliminary study shows that YOLO v3 algorithm have 80% accuracy for object detection. It is concluded that the deep learning neural network model is more suitable to measuring the traffic density and later any traffic situation decision could be made.

In the future work on this project:

We could extend the application of the system by creating mobile application where a user interface would be used to provide the user data in the voice format. Many sensors implemented on different adjacent signals which helps to provide the traffic data and send to a controller through IoT devices. Another aspect is non-trivial as many different factors such as weather(fog, rain, wind, wind etc) could have significant impact on the sensor related system.

Further studies will need to consider some other senecios regarding the proposed approach. We ignored vehicle details when counting vehicles in the present model. Of course, the purpose of counting vehicles is considered to evaluating traffic situation at the aggregate level in the present study. However, in the future, advanced counting technology should recognize the details(size, make and type) of each vehicle. The next version of the present study will measure the space mean speed as well as the traffic density. The space mean speed is another important parameter in traffic engineering, and it cannot be measured directly with the existing surveillance systems.

REFERENCES

[1] Aditya Krishna K.V.S, Abhishek K, Allam Swaraj, Shantala Devi Patil, Gopala Krishna Shyam, “Smart traffic analysis using machine learning,” International Journal of Engineering and Advanced Technology (IJEAT), ISSN: 2249–8958, Volume-8, Issue-5S, May, 2019 .

[2] Huansheng Song, Haoxiang Liang, Huaiyu Li, Zhe Dai and Xu Yun, “Vision-based vehicle detection and counting system using deep learning in highway scenes,” European Transport Research Review.

[3] Donato Impedovo, Fabrizio Balducci, Vincenzo Dentamaro and Giuseppe Pirlo, “Vehicular traffic congestion classification by visual features and deep Learning approaches: A comparison,” Dipartimento di Informatica, Università degli Studi di Bari Aldo Moro, 70125 Bari, Italy.

[4] Aya Salama Abdelhady, “Vehicle classifcation for automatic traffic density estimation,” The American University in Cairo School of Science and Engineering, unpublished.

[5] ParthoSharothi Paul, “Traffic density estimation and fõow control for video surveillance system”, unpublished

[6] Domen Tabernik and Danijel Skoˇcaj, “Deep learning for large-scale traffic-sign detection and recognition,” University of Ljubljana.

[7] Walid Balid, Hasan Tafish, and Hazem H. Refai, “Intelligent vehicle counting and classification sensor for real-time traffic surveillance” IEEE Transactions on Intelligent Transportation Systems · September 2017

[8] Kh Tohidul Islam, Ram Gopal Raj and Ghulam Mujtaba, “Recognition of traffic sign based on bag-of-words and artificial neural network”.

[9] Julian Nubert, Nicholas Giai Truong, Abel Lim, Herbert Ilhan Tanujaya, Leah Lim and Mai Anh Vu, “Traffic density estimation using a convolutional neural network machine learning project,” National University of Singapore, April 2018.

[10] Jiyong Chung and Keemin Sohn, “Image-based learning to measure traffic density using a deep convolutional neural network,” T-ITS-16-09-0605.R1.

[11] S. Y. Cho, T. W. S. Chow, and C. T. Leung, “A neural-based crowd estimation by hybrid global learning algorithm,” IEEE Trans. Syst., Man, Cybern. B, vol. 29, no. 4, pp. 535–541, Aug. 1999.

[12] D. Kong, D. Gray, and H. Tao, “A viewpoint invariant approach for crowd counting,” in International Conference on Pattern Recogn., Santa Cruz, CA, 2006, pp. 1187–1190.

[13] A. N. Marana, S. A. Velastin, L. F. Costa, and R. A. Lotufo, “Estimation of crowd density using image processing,” in Proc. IEE Colloquium Image Processing for Security Applications, Mar. 1997., pp. 11/1–11/8.

[14] Qiu-Lin, L.I., & Jia-Feng, H.E. (2011), “Vehicles detection based on three-frame-difference method and cross-entropy threshold method,” Computer Engineering, 37(4), 172–174.

[15] Liu, Y., Yao, L., Shi, Q., Ding, J. (2014), “Optical flow based urban road vehicle tracking,” In 2013 Ninth International Conference on Computational Intelligence and Security https://doi.org/10.1109/cis.2013.89: IEEE.

[16] Park, K., Lee, D., Park, Y. (2007), “Video-based detection of street-parking violation,” In International Conference on Image Processing. https://www.tib.eu/en/search/id/BLCP%3ACN066390870/Video-based-detection of street- parking-violation, vol. 1 (pp. 152–156). Las Vegas: IEEE.

[17] Ferryman, J.M., Worrall, A.D., Sullivan, G.D., Baker, K.D. (1995). “A generic deformable model for vehicle recognition,” British Machine Vision Conference 1995,in Procedings. https://doi.org/10.5244/c.9.13: British Machine Vision Association.

[18] Han, D., Leotta, M.J., Cooper, D.B., Mundy, J.L. (2006). “Vehicle class recognition from video-based on 3d curve probes,” IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance. https://doi.org/10.1109/vspets.2005.1570927: IEEE, 2005.

[19] Zhao, Z.Q., Zheng, P., Xu, S.T., Wu, X. (2018), “Object detection with deep learning”: A review. arXiv e-prints, arXiv:1807.05511.

[20] Cai, Z., Fan, Q., Feris, R.S., Vasconcelos, N. (2016), “A unified multi-scale deep convolutional neural network for fast object detection,” In 2016 European conference on computer vision. https://doi.org/10.1007/978-3-319-46493-0_22 (pp. 354–370): Springer International Publishing.

[21] Hu, X., Xu, X., Xiao, Y., Hao, C., He, S., Jing, Q., Heng, P.A. (2018), “A scale-insensitive convolutional neural network for fast vehicle detection,” IEEE Transactions on Intelligent Transportation Systems, PP(99), 1–10.

[22] .https://imageai.readthedocs.io/en/latest/

[23] https://www.tensorflow.org/

[24] https://numpy.org/

[25] https://pypi.org/