Recognition of a traffic situation in real-time

M.Eng Information Technology  
Automation Laboratory  
Syed Muhammad Abdullah Ahsan, Moin Abbas Qureshi 1274385, 1273878  
ahsan@stud.fra-uas.de qureshi@stud.fra-uas.de

***Abstract*— Solution implemented in this project is implemented with python with OpenCV video processing system and TensorFlow machine learning library to recognize traffic participants and traffic situation. Vehicle classification and detection is very important for traffic management and planning. In this project we present Computer Vision based system for traffic recognition and vehicle detection. Images from the video frames are used to detect and identify traffic participants, A pre trained model YOLO is used to detect and classify moving vehicles, After the detection of traffic participants traffic situation is predicted. For traffic prediction we have created and trained a Convolution neural network with three classes Heavy Traffic, Medium Traffic and No Traffic. Experimental results are demonstrated from the real-time street view video taken from single camera .**

Keywords— CNN, YOLO, Object detection, Traffic recognition

# Introduction

Humans look at an image and instantly know what the objects in the image are, where they are, and how they interact. The human visual system is fast and intuitive, which allows us to perform complex tasks such as driving with little thought. Fast, accurate object detection algorithms can allow computers to drive cars without special sensors, enable devices that help transmit real-time information to human users, and open opportunities for common purpose, responsive robotic systems. Current adoption programs are restoring classifiers to recovery. To find an object, these programs take a classifier of the item and scan it at various locations and scales with a test image. Systems like deformable parts models (DPM) use a sliding window approach where the classifier is run at evenly spaced locations over the entire image [1]. With the availability of large amounts of data, faster GPUs, and better algorithms, we can now easily train computers to detect and classify multiple objects within an image with high accuracy. Object Detection involves processes to locate objects in a image/video or a webcam feed. Object detection is now being used everywhere. The applications are endless, be it Video surveillance, Anomaly detection, Face detection or Self driving cars.

(Object Detection) is "recognition target is given its exact location in the diagram", the contents of which can be deconstructed into three parts:

* Identify an object (Classification)
* Give the location of the object in the diagram (Localization)

Identify all the targets and their locations (Detection) in the picture.

# Related work

In this project, we have created a system which can easily detect traffic participants as well as current traffic situation in real time. We have considered three traffic scenarios high traffic, medium traffic and no traffic. Three main algorithms are currently being used in the industry.

* R-CNN - Region-CNN
* YOLO - You Only Look Once
* SSD - Single Shot MultiBox Detector

Also, there are many other machine learning algorithms in use we had to research out which fits our requirements because every algorithm has its own advantages and disadvantages. As we need to train our own model for learning the traffic situation as well as pre trained model for object detection we need to process in real time so after studying above algorithms we found out that YOLO gives us the best results in terms of time and accuracy. On the other hand, R-CNN takes round about 57 seconds for each frame to test in real time and some goes for SSD.

# YOU ONLY LOOK ONCE (yolo)

YOLO (You Only Look Once) is an end-to-end object detection method. Although it is an end-to-end model, it can be roughly decomposed into two parts: regression prediction and frame selection. First, a regression model (such as sliding window CNN) is used to obtain the border annotation of the entire image at once, and the image is gridded. Each grid is responsible for the borders of the object whose center point falls on it; Non-maximum suppression is based on each grid to get the frame output after filtering. The process is as follows:

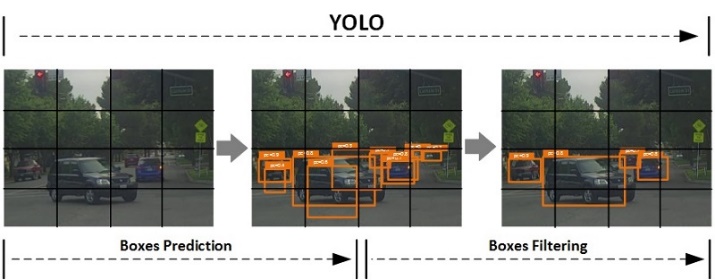


Fig. 1 YOLO Grid

In the above figure, the label prediction (Boxes Prediction) link is responsible for obtaining many large candidate borders. It applies a single forward pass neural network to the whole image and predicts the bounding boxes and their class probabilities as well. This technique makes YOLO a super-fast real-time object detection algorithm.

## Working of YOLO

The YOLOv3 network divides an input image into S x S grid of cells and predicts bounding boxes as well as class probabilities for each grid. Each grid cell is responsible for predicting B bounding boxes and C class probabilities of objects whose centers fall inside the grid cell. Bounding boxes are the regions of interest (ROI) of the candidate objects.

The “B” is associated with the number of using anchors. Each bounding box has (5 + C) attributes. The value of “5” is related to 5 bounding box attributes, those are center coordinates (bx, by) and shape (bh, bw) of the bounding box, and one confidence score. The “C” is the number of classes. The confidence score reflects how confidence a box contains an object.

The confidence score is in the range of 0 – 1. We’ll be talking this confidence score in more detail in the section [Non-Maximum Suppression (NMS)](https://machinelearningspace.com/yolov3-tensorflow-2-part-1/#nms-unique).Since we have S x S grid of cells, after running a single forward pass convolutional neural network to the whole image, YOLOv3 produces a 3-D tensor with the shape of [S, S, B \* (5 + C].

The following figure illustrates the basic principle of YOLOv3 where the input image is divided into the 13 x 13 grid of cells (13 x 13 grid of cells is used for the first scale, whereas YOLOv3 actually uses 3 different scales and we're going to discuss it in the section [prediction across scale](https://machinelearningspace.com/yolov3-tensorflow-2-part-1/#unique-identifier)).

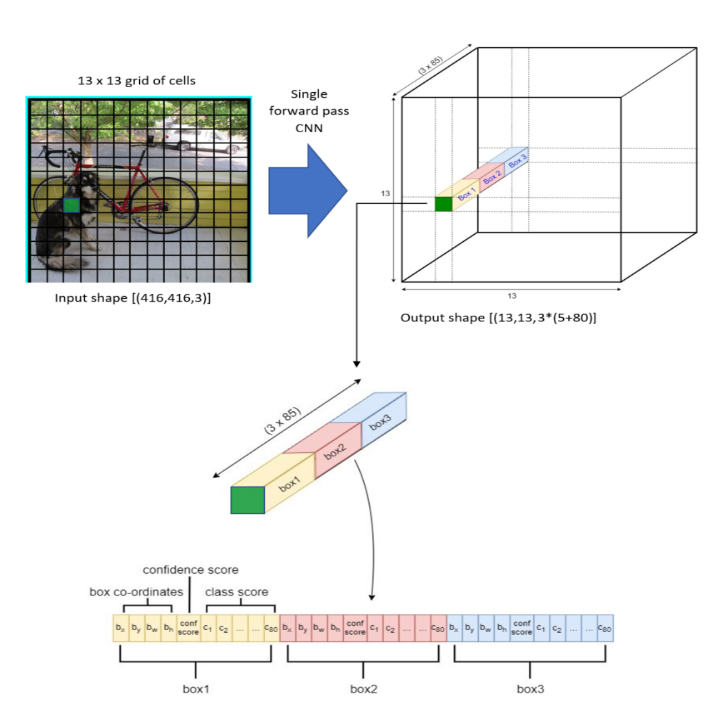


Fig. 2 YOLOv3 Overview

First, YOLOv3 is extremely fast. We run neural network YOLOv3 pretrained on a new image at test time to predict traffic participants as well as current traffic situation. Our second model is Traffic recognition classifier using CNN, Kera’s, and TensorFlow backend trained on 3 different classes of traffic situations. Below is the GUI which shows output of YOLOv3 with bounded boxes and on traffic participants and current traffic situation as well as probability in figure 3. For testing the video is taken from the street view looking towards incoming traffic and can be used for detecting traffic participants and classifying vehicles for any direction of traffic flow. We will not go into detail of YOLOv3 rather we will explain more in detail on our own CNN model that is predicting real time traffic situation.



Fig. 3 YOLOv3 Detecting Objects

# Convolutional Neural Network

Convolutional Neural Network (CNN) is a variant of standard neural network which is specifically designed to process sequence data such as image [2]. It requires minimal data preprocessing and it can automatically detect the invariant and extract important features [3]. There are two main components of CNN, convolutional layer and pooling layer [4]. The components are mainly inspired by mammalian visual cortex that have two basic cell types: complex cells (convolution layer) which have receptive fields and are locally invariant to exact position of the pattern, and simple cells (pooling layer) that respond maximally to specific edge-like patterns within their receptive field. A basic CNN usually consists of one or more convolution-pooling layers and fully connected layer(s). CNN uses three main idea: local receptive fields, shared weights, and pooling [5]. A typical convolutional neural network is usually composed of the following three-layered structures. Convolution layer (Convolution), down-sampling pooling layer (Pooling), fully connected layer

* 1. Convolutional Layer

The convolution operation is a linear operation, and the neural network needs to fit a non-linear function, so similar to the previous fully connected network, we need to add an activation function, commonly used are sigmoid function, tanh function, ReLU function, etc.

The convolutional layer in the front of the neural network has a small receptive field, which can capture the local and detailed information of the image, that is, each pixel (Activation value) of the output image is only the result of the calculation of the small value of the input image. The receptive field of the convolutional layer behind increases, which is used to capture more complex and abstract information. After the operation of multiple convolutional layers, the abstract representation of the image at various scales is finally obtained.

* 1. Pooling Layer

Through the convolution operation, we have completed the dimensionality reduction and feature extraction of the input image, but the dimension of the feature image is still very high. High dimensionality is not only time-consuming to calculate, but also easy to cause overfitting. To this end, down sampling technology is introduced, also known as pooling, or pooling operation. The practice of pooling is to replace a certain area of ​​the image with a value, such as the maximum value or the average value. If the maximum value is used, it is called max pooling; if the mean value is used, it is called mean pooling. In addition to reducing the image size, another benefit of down sampling is translation and rotation invariance, because the output value is calculated from an area of ​​the image and is not sensitive to translation and rotation. To summarize the role of the pooling layer:

* Reduce dimension, reduce model size, and increase calculation speed
* Reduce the probability of overfitting and improve the robustness of feature extraction
* Insensitive to translation and rotation

The specific implementation of the pooling layer is to divide the obtained feature image after the convolution operation, the image is divided into disjoint blocks, and calculate the maximum or average value in these blocks to obtain the pooled image.

* 1. Fully connected layer

Fully connected layers (FC) play the role of "classifier" in the entire convolutional neural network. If the operations of convolutional layer, pooling layer and activation function layer are to map the original data to the hidden layer feature space, the fully connected layer plays the role of mapping the learned "distributed feature representation" to the sample label space . In actual use, the fully connected layer can be realized by the convolution operation: the fully connected layer that is fully connected to the front layer can be converted into a convolution with a convolution kernel of 1×1; The convolution kernel is the global convolution of h × w, h and w are the height and width of the previous layer convolution result, respectively.

# SYSTEM DESIGN

Vehicles detection must be implemented in a different environment where the light and the traffic status changing. In our proposed system, we accept the traffic flow video from a camera and convert video into frames extract using OpenCV performed the detection of moving objects and performed traffic situation prediction. The system we propose consists of three stages.

* System Initialisation: GUI is created to open image from file system or from web cam that records a continuous stream of data and sends to the system for analysis. Our system works in two modes, pre-recorded video mode and real-time camera mode. We can provide pre-recorded traffic flow video for detection.
* Traffic Participants Detection: In this stage, each frame is taken into focus and on successive analysis and operations background, prediction take place. All the moving vehicles/objects can be tracked and identified.
* Traffic Detection: After detecting traffic participants each frame is also passed to CNN model to predict the traffic situation for example Heavy Traffic, Medium Traffic and No Traffic and probability as well.

After completing all the steps bounded boxes are created to identify objects and traffic situation is displayed on the GUI as shown in fig 3.

* 1. *Collecting the Dataset*

In order to train our machine, we need a huge amount of data so that our model can learn from them by identifying out certain relations and common features related to the objects and classes. We take many videos of traffic and converted into pictures to make our own custom data set of Frankfurt traffic and also downloaded traffic images from online, we created different categories and distributed into different relevant classes as shown in the fig 4.We have divide our data set to ratio of 0.2 e-g if 1000 images are for training 200 will be used for testing and for validation 20% of remaining 800 will be 160 for validation.



Fig. 4 Training Data set

Our deep learning dataset consists total of 2,500 images of Traffic. Our goal is to train a Convolutional Neural Network using Kera’s, TensorFlow and deep learning libraries to recognize and classify each of these Traffic situations. To verify that the dataset looks correct, we have plotted the 9 images from the training set and display the class name above each image.



Fig. 5 Images for training with respect to each class.

In the above figure first row belongs to class 0(No Traffic). Second row belongs to class 1(Medium Traffic) and finally our last class 2 (Heavy Traffic). This diverse mix of training images will allow our CNN to recognize our three Traffic classes across a range of images and as we’ll see, we’ll be able to obtain 90% + classification accuracy.

B. Data augmentation

* 1. *Data Augmentation*

Data augmentation is an efﬁcient process to use with image data to remove overﬁtting and train the model better. It is a process of generating more data from the available data by rotating, scaling, translating, cropping and other transformation. An embedded data loader does the job at train time. When it loads the data, it applies the transformation and generates a slightly different image each time. As the images of our dataset are nicely collected, we carefully applied only a little amount of augmentation [6]. The augmentation types and amount are summarized in the table 1 below:

|  |  |
| --- | --- |
| **Type of Augmentation** | **Values** |
| Rotation | 10° to 20° |
| Scaling Factor | ×1.1 |

Table. 1 Data Augmentation

* 1. *Training and Evaluation*

To train our CNN model, first step is to initialize the model with Sequential (). After that we flatten our data and add our additional 3 hidden layers. This step is fully customizable according to our need. We made several different models with different drop out, hidden layers and activation. It is also best for loss to be categorical crossenthropy but it can be changed. Then after we have created and compiled our model, we fit our training and validation data to it with the specifications we mentioned earlier. Finally, we create an evaluation step, to check for the accuracy of our model training set versus validation set shown in the figure 4.

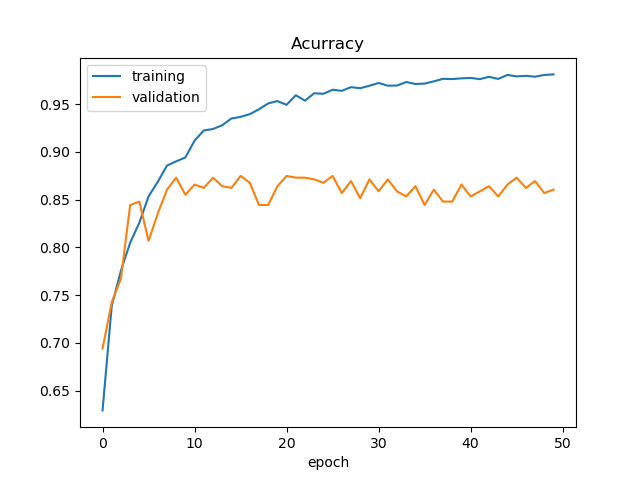


Fig. 6 Training and Validation Accuracy

Our CNN has achieved a accuracy of above 80%. Model accuracy is not a reliable metric of performance, because it will yield misleading results if the validation data set is unbalanced. For example, if there were 90 images of low traffic and only 10 images of heavy traffic in the validation data set and if the model predicts all the images as low traffic. The overall accuracy would be 90%. Model summery is shown in Fig 7.

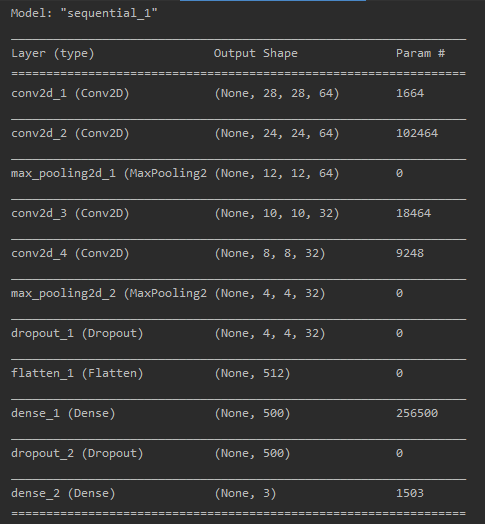


Fig. 7 Model Summery

# Experimental results

We tested this system on a laptop powered by an Intel Core i3 (2.30 GHZ) CPU and 8GB RAM. equipped with web cam. We tested the system on image sequences of street scenes. The system can detect vehicles and classify traffic successfully. Figs. 8-10 show some results of our system.

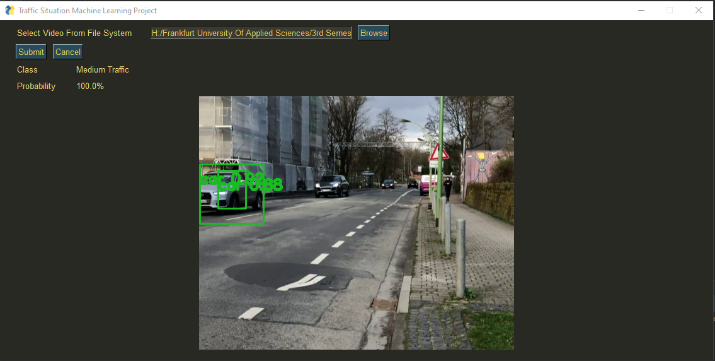


Fig. 8 Medium Traffic

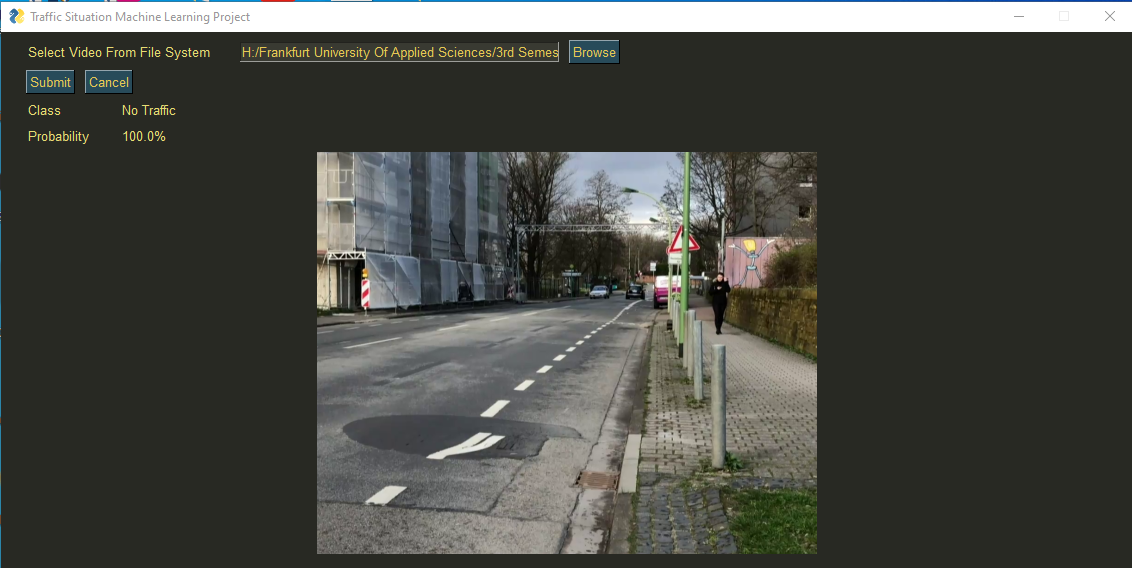


Fig. 9 No Traffic

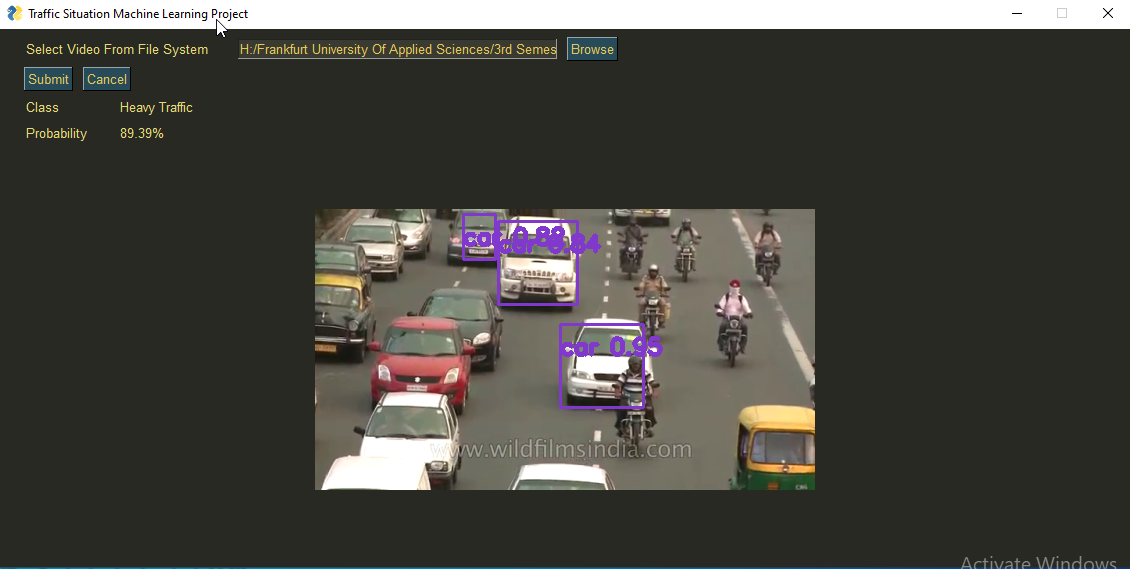


Fig. 10 No Traffic

From the above images from the video sequences we can easily predict what kind of traffic is there as well as the traffic objects.

* 1. *Confusion Matrix*

The confusion matrix allows us to visualize the performance of the trained model. It makes it easy to see if the system is confusing three classes. It also summarizes the results of testing the model for further inspection. The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made. In this section, we have discussed to classify our own dataset digits for visualization confusion matrix. Confusion matrix can be very helpful to see your model drawbacks. There is a very nice graphic to locate the outliers: The Confusion Matrix. The numbers that the model has recognized are shown on the right and the numbers that the model should have recognized are on the top. Thus, the aim of the exercise is to drive all values onto the diagonal and to trim the rest to zero.

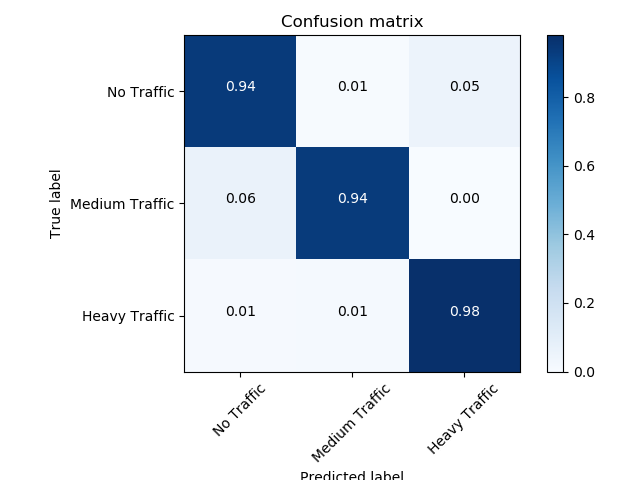


Fig. 11 Confusion Matrix

Above confusion matrix is obtained after training of our model. In which we can see that out system is accurately recognizing no traffic with the probability of 0.94, for medium traffic 0.94 and heavy traffic 0.98.

# CONCLUSION

In this work, we have proposed successful machine learning combination of two models including YOLOv3 and our own proposed CNN this is trained on 2232 images, validate on 559 and tested on 698 images with data augmentation. The solution can detect traffic situations and traffic participants on a road. The CNN model has achieved an average accuracy of above 80%.

# References

|  |  |
| --- | --- |
| [1] | A. Rajagede, Recognizing Arabic Letter utterance using convolutional neural network. Proceedings, 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing , SNPD 2017, pp. 181 – 186., 2017. |
| [2] | C. Dewa, Javanese vowels sound classification with convolutional neural network, in: Proceedings of International Seminar on Intelligent Technology and Its Applications (ISITIA), 2016. |
| [3] | R. D. C. a. A. Rajagede, Recognizing Arabic letter utterance using convolutional neural network, in: 2017 18th IEEE/ACIS International Conference on software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), pp. 181-186., 2017. |
| [4] | F. A. A. Dewa C.K., Convolutional neural networks for handwritten Javanese character recognition., Indonesian Journal of Computing and Cybernetics Systems, 12, pp. 83-94, 2018. |
| [5] | M. A. Nielsen, Neural Networks and Deep Learning, 2015 . |
| [6] | M. S. R. Shuvendu Roy, Emergency Vehicle Detection on Heavy Traffic, 2019. |