VEHICLE TYPES AND COLORS DETECTION FOR AMBER AND SILVER ALERT EMERGENCIES USING MACHINE LEARNING

by

Uma Kamatchi Kesava Pillai, B.E., M.E

A thesis submitted to the Graduate Council of
Texas State University in partial fulfillment
of the requirements for the degree of
Master of Science
with a Major in Engineering
May 2021

Committee Members:

Damian Valles, Chair

William Stapleton

Harold Stern

FAIR USE AND AUTHOR'S PERMISSION STATEMENT

Fair Use

Duplication Permission

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor Dr. Damian Valles whose understanding, generous guidance, appreciations, valuable feedback, and support made it possible to work on a topic that is passionate to me.

I am grateful to Dr. William Stapleton for accepting to be a thesis committee member and his insights and guidance throughout the thesis work. I am thankful to Dr. Harold stern, for providing valuable feedback and suggestions on thesis report. I am very much thankful to Dr. Vishu Viswanathan for providing guidance throughout graduate study.

I would especially like to thank my family. My husband, Chidambaram has been extremely supportive, providing important instructions to hosting the experiments and has made countless sacrifices to help me get to this point. My son, Kavin who has been very understanding and continually provided the motivation to finish my degree with expediency. My parents, deserve special thanks for their continued support, blessing and encouragement.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

ABSTRACT

The National Center for Missing & Exploited Children estimated that 145 AMBER Alerts were issued in the U.S. involving 180 children in 2019, where 85% had involved vehicles, and in Florida, 136 Silver Alert was issued in (2008-2009). The details of broadcasting in Amber and Silver alerts are color, type of the vehicle, vehicle license plate numbers, and car brands. Vehicle types include six classes as Truck, Bus, van, SUV, Sedan, and Motorcycle. Vehicle colors include eight classes: green, blue, black, white, gray, yellow, cyan, and red. The system works once the Amber alerts are issued; the time taken to recovery is more than twelve hours for children in seventeen cases. More recovery time is due to the time is taken for someone to recognize the vehicle involved in child abduction and authorities to bring back the child or elderly safe back home. This thesis research aims to design a Deep Learning model to classify vehicle colors and types faster and accurately. First, Shallow CNN and Deep CNN, inspired by VGG16, are designed for vehicle colors and classification types. Second, implementation of the pre-trained YOLO object detector to detect vehicles in images. Extraction of license plate characters and numbers using image processing techniques with OpenCV and Python will help identify possible License Plate matches. The whole process will help extract colors and types of vehicles from the child abduction involved, bringing child and elderly person safe home.

1 INTRODUCTION

1.1 Problem Statement

Various research has been done on vehicle classification using Image Processing and Machine Learning in different applications like traffic monitoring, traffic surveillance systems, and vehicle tracking [1]. Roughly 800,000 children are reported missing each year in the United States according to the National Center for Missing and Exploited Children [2]. The US solution to this problem is the Amber Alert program. The Amber Alert program is a collaborative effort between law enforcement agencies, broadcast companies, transportation agencies, and the wireless industry to widely spread an urgent message to the US population in the event of a child abduction. AMBER is officially created acronym for America's Missing: Broadcast Emergency Response. The statistics report of an amber alert in Figure 1 shows the number of children involved in the amber alert from 2015 to 2019 [2], where 85% involved vehicles. In North Carolina, of the 128 silver alerts, 118 seniors were safely recovered in 2008.

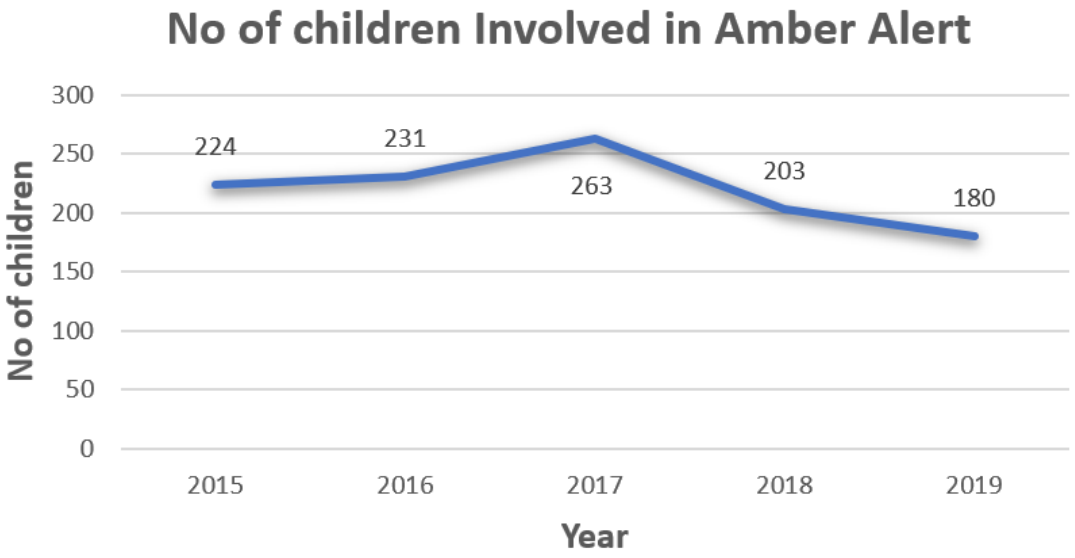


Figure 1. No of Children involved in Amber alert [1]

The Amber Alert systems current functionality has failed to identify the vehicle involved in the Amber/Silver alert. Currently, the system relies upon the observations of community members to spot missing children. This form of crowd sourcing of manual subject identification is ill-suited for the twenty first century where highway cameras are enabled ubiquitous. The system works once the Amber alerts are issued, the time taken to recovery is more than twelve hours for children in seventeen cases [2], as shown in Figure 2. More recovery time is due to the time is taken for someone to recognize the vehicle involved in child abduction and authorities to bring back the child or elderly safe back home. Designing a faster classification and detection model can decrease the time taken to bring back the child safely.

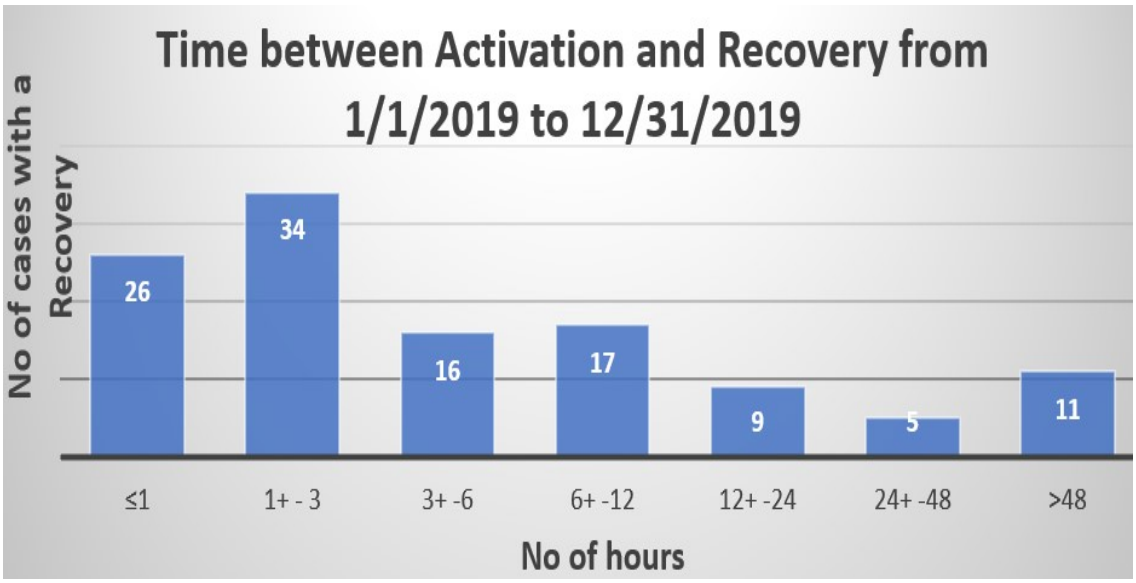


Figure 2. Time between Activation and Recovery [2]

The details of broadcasting in Amber and Silver alerts are color, type of the vehicle, vehicle license plate numbers, and car brands. Vehicle types include six classes as Truck, Bus, van, SUV, Sedan, and Motorcycle. Vehicle colors include eight classes: green, blue, black, white, gray, yellow, cyan, and red. The datasets in [3][4][5] are

imbalanced and challenging due to the illumination variation, haze, and overexposure with different environmental conditions. Haze removal algorithm, dark channel prior (DCP) [6][7][8] to get haze-free images, and data augmentation [9] helps to increase training samples with Keras Image Data Generator. An increase in datasets and clear images helps to improve the accuracy performance of the vehicle classification models. The thesis goal is when an Amber or Silver signal is broadcast, the deep learning model checks with the vehicle's specifications and extracts the vehicle's color and type under different environmental conditions like illumination and overexposure. The model then recognizes the characters and numbers of the vehicle's license using image processing techniques like image filtering, edge detection with OpenCV in Python, and provide notification of the detected vehicle.

1.2 Contributions

The main contributions of this thesis research have been summarized as follows

- Development of image preprocessing steps to provide vehicle information enhancements that will help to find vehicle in a real-world environmental challenge includes hazy, illumination and overexposure.
- Added Data augmentation techniques to obtain balanced datasets.
- Development of CNN techniques to extract vehicle's color and type with higher classification accuracy and faster detection in Amber/Silver emergencies.
- Deep learning architecture YOLO is applied for feature of detection as whole and the relevance in providing the detection feature for these emergencies.

1.3 Thesis Outline

The thesis is structured as follows. Chapter 2 discusses the background that explains the motivation behind the research work is reviewed. Chapter 3 explains the literature reviews related to vehicle types, colors, and license plates. Chapter 4 explains the theories of deep learning Algorithms such as CNN and YOLO object detection and optical character recognition. Chapter 5 discusses the research methodology, including datasets, data pre-processing, and experiment methods of shallow and deep CNN architecture, vehicle detection, license plate characters, and numbers recognition. Chapter 6 then discusses vehicle color and type classification; detection, and the chapter ends with a conclusion and future work.

2 BACKGROUND

The United States Amber Alert program [2] began in Dallas-Fort Worth, Texas, and it has grown into a program at both state and federal levels. This program was eventually taken to the National Center for Missing & Exploited Children (NCMEC) to request a national initiative. The Amber alert system allows the broadcasters and transportation authorities to immediately distribute information about child abduction to the public through the wireless Emergency Alert System. It enables the entire community to assist in searching for the safety and recovery of the child.

A Silver Alert [10] is a public notification system in the United States to broadcast information about missing persons – especially senior citizens with Alzheimer's disease or other mental disabilities to aid in locating them. The United States Silver Alert program began in Oklahoma, and it has grown into a program at both state and federal levels. A Silver Alert works in much the same way as an Amber Alert for missing children. When a senior or someone with cognitive impairment goes missing, their description and last known whereabouts are sent to law enforcement and media to help bring about their safe return. Public information in a Silver Alert usually consists of the missing person's name and description and a description of the missing person's vehicle. As of December 31st, 2018, there have been 956 successful children's recovery by the Amber Alert Program, where 85% involved vehicle identification. The retrieval rates resulting from Silver Alert show more than 90% success level. For example, in North Carolina, 128 Silver Alerts were issued in 2008. Of these, 118 seniors were safely recovered, which results in a 92% success level [11].

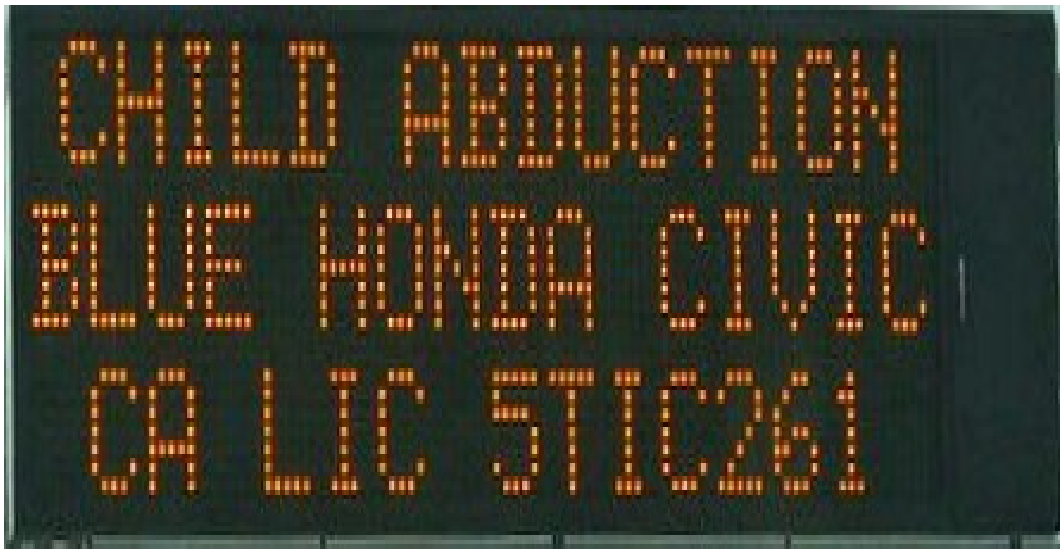


Figure 3. Broadcasting of an Amber Alert in a Digital Highway Sign [12]

Figure 3 shows the details of broadcasting in Amber and Silver alerts are color, type of the vehicle, vehicle license plate numbers, and car brands involved in child abduction. Once the alerts are issued, the time taken to recovery is more than 12 hours for children in 27 cases [11]. More recovery time is due to the time is taken for someone to recognize the vehicle involved in child abduction and authorities to bring back the child or elderly safe back home. Machine Learning makes a computer learn independently and adapt its understanding based on exposure to new data instead of programmed to a specific task. Machine learning models can be trained with millions of labeled images, and they can successfully identify images more quickly than a human. Deep Learning has proven fast vehicle detection and classify vehicle's color and type accurately.

This research aims to develop a deep learning model to extract vehicle colors and types to detect through an image feed to find possible matches in the emergency alert message. The matched images' stored snapshot will help recognize characters and numbers in the License plate from the Optical Character Recognition (OCR) model. The

whole process will help find possible matches for these emergency alerts for child/older adults' safe return.

3 LITERATURE REVIEW

This section discusses the research works in vehicle detection and vehicle classification on various applications like autonomous vehicles, Intelligent Transport systems and Surveillance Systems. Machine Learning and Deep Learning have revolutionized image applications when integrated with computer vision, and it is the core technology behind capabilities for fast object detection. Literature review divided with five sections:

3.1 Vehicle Colors Classification

3.2 Vehicle Type Classification

3.3 License Plate Recognition

3.4 Vehicle Detection

3.5 Conclusion

3.1 Vehicle Colors Classification

The authors in [5] proposed a feature context approach to identify vehicle color. Created their datasets on urban roads, contains 15,601 vehicle images with eight classes of vehicle color. The dataset is very challenging due to the noise caused by illumination variation, haze, and overexposure. Images are taken in frontal view by a high-definition camera. Authors adopted different preprocessing techniques like haze removal to remove haze and color contrast to improve the image's quality. They achieved an average accuracy of 90.68% using the Support Vector Machine (SVM) classifier.

The authors in [13] used the dataset in [5] to classify eight classes of vehicle color using Convolution Neural Network (CNN). The CNN model learned classification based on color distribution. They convert the input image into two different color spaces using

the Hue-Saturation-Value (HSV) and the International Commission on illumination (CIELAB) transformations. They experimented with different color spaces such as Red-Green-Blue (RGB), HSV, and CIELAB. Achieved an accuracy average of 94.47% in RGB color space, which outperforms the dataset in [5] using the feature context approach.

The authors in [14] trained a model with AlexNet, GoogLeNet, ColorNet, Network in Network (NIN) and showed NIN outperforms the previous models. Collected images from the High Definition (HD) bayonets, cropped from the surveillance videos, and resized them to 256x256 pixel resolution. The dataset consists of 15,016 vehicle images with different cars, buses, SUVs, and trucks. The model recognizes the color of the vehicles with eight classes (black, blue, gray, green, red, cyan, white, and yellow) with an average accuracy of 95%. The drawback is they did not consider environmental conditions.

Researchers in [15] used a Deep Neural Network (DNN), VGG-16 model, to recognize six colors (white, black, red, green, blue, and yellow) by optimizing network structure and parameters. Obtained data from city traffic junction. The dataset contains 2,520 images in the training data set and 1,000 images in the test dataset. Experimental results show the model took 2,240,000 max number of iterations, 0.008ms, to identify each car on an and repeated five times to obtain the correct rate of six colors. The average color recognition rate is 92.68%. The drawback is they did not consider environmental conditions.

Researchers in [16] used a CNN architecture to classify vehicle types with four classes (small, medium, large, and unknown) and vehicle color with seven classes (black,

blue, white, green, yellow, red, and unknown). They extracted 914 vehicle images from surveillance videos. The authors compared their method with previous work that utilized decision trees, random forest, and DNN classifier. The results show that the classification of vehicle type accuracy increased by 1.8%, and vehicle color increased by 0.8%. The drawback is they did not consider the environmental condition and fine-tuning with different hyperparameters.

3.2 Vehicle Type Classification

Researchers in [17] proposed a modified YOLOv2 network structure to classify vehicle types with eight classes by a multi-layer feature fusion strategy. The strategy means global and local features of one layer are fused so that model can distinguish tiny differences among vehicle types and removed repeated convolution layers in the higher ends. The authors used two datasets as the Beijing Institute of Technology (BIT) Vehicle dataset containing 9,580 vehicles and the Comp-cars Dataset containing commercial vehicle images (1,687 vehicles) and road surveillance cameras. It contains only Sedan and SUV types of vehicles with just over 40,000 image samples. It includes day and night scenes, but no noise background, rain, and other weather conditions. The authors compared their model with YOLOv2 and showed a mean average precision of 94.8%.

The authors in [18] proposed an improved YOLOv2 network structure for Multi-scale detection in varying vehicles' sizes. They proposed a Ratio k-means (R k-means++) clustering algorithm that generates anchor boxes to improve location accuracy. They proposed a Focal Loss to reduce the negative influence on training resulting from an imbalance between vehicles and background for training the model to decrease the imbalance between vehicles and background noise by adjusting hyper tuning focusing

parameters such as gamma and alpha. Experimented with the BIT-Vehicle dataset and achieved a 97.3% mean average precision. The main drawback is they did not consider environmental conditions.

The authors in [19] used a CNN and data augmentation techniques to classify vehicle types on the application to automation traffic lights. They created their dataset that includes 2,400 samples with four classes (School bus, Ambulance, Police, and CTM (Moroccan Transport Company)). Evaluated model performance with Precision, Recall, and accuracy of training and testing set. They achieved an average Precision of 0.89 and Recall of 0.83. The main drawback is they did not consider environmental conditions.

3.3 License Plate Recognition

The authors in [20] proposed an automatic number plate recognition system using image processing technology to identify the vehicles. They used various image processing techniques such as Gray-Scale Conversion, Noise Filtering, Image Binarization, Histogram Equalization, and Template Matching. They recognized characters and numbers in the license plate using MATLAB.

Researchers in [21] proposed an Adaptive Technique for Computer Vision-Based Vehicles License Plate Detection System. Three module includes Detection of the license plate, Segmentation of Characters and Text Box Generation. They achieved an average accuracy of 78.2% using MATLAB. The authors in [22] introduced novel CNN for detecting and rectifying multiple distorted license plates in a single image, which is then fed to an Optical Character Recognition (OCR) method to get characters and numbers. The proposed approach outperforms existing methods by considering different challenges includes oblique views and distorted license plates.

3.4 Vehicle Detection

The authors in [23] published the first version of the YOLO algorithm to detect objects in real-time. YOLO aims to convert target detection problems to regression problems by Machine Learning. YOLO predicted directly from full images in one evaluation via a single neural network, which is optimized end-to-end directly on the detection performance. They achieved a real-time detection speed of 45 frames per second. They then published with various improvements like a high-resolution classifier, anchor boxes, Darknet nineteen Convolutional Layers, and multiple usages of datasets like COCO [24] and PASCAL VOC [25]. It is faster and more precise. YOLOv3 [26] trained on the COCO dataset contains 80 labels, which shows fast and accurate results compared to YOLOv2 [27]. It uses logistic classifiers for each class and gives the score to every bounding box's objects instead of the softmax approach.

3.5 Conclusion

This research aims to develop a deep learning model to extract vehicle colors and types with detection of image/camera feed to find possible matches in the emergency alert message. The stored snapshot of the matched images will help recognize characters and numbers in the License plate from the Optical Character Recognition (OCR) model. The whole process will help to find the vehicle involved in the alert. The literature survey discusses the details of data accumulation, the number of classes included in vehicle types and colors, and algorithm implementation to give higher accuracy. Previous works experimented with different classifiers and found that deep CNN showed promising results for vehicle identification and YOLO for vehicle detection. Based on these related research efforts, this thesis work experimented with a CNN classifier under different

environmental conditions. They implemented a pre-trained YOLO model for detection and an Optical Character Recognition (OCR) model to extract characters and numbers.

4 DEEP LEARNING ALGORITHMS

Deep Learning is the extension of Artificial Neural Networks (ANNs), a set of algorithms that learn patterns inspired by the brain's structure and function. Deep-learning architectures such as Convolution Neural Networks (CNN), Deep Neural Networks (DNN) have been applied to fields including computer vision, machine vision, speech recognition, natural language processing, and audio recognition. The Deep Learning model has a robust learning ability, integrating the feature extraction and classification process into an image classification task to improve classification accuracy.

This thesis research aims to design a Deep Learning model to classify vehicle colors and types faster and accurately. First, Shallow CNN and Deep CNN, inspired by VGG16, are designed for vehicle colors and classification types. Second, implementation of the YOLO object detector to detect vehicles in both image and video. Extraction of license plate characters and numbers using image processing techniques with OpenCV and Python will help identify possible License Plate matches. The whole process will help extract colors and types of vehicles from the child abduction involved, bringing child and elderly person safe home. The following sections give an overview of the theories behind CNN and YOLO in detail.

4.1 Convolutional Neural Network (CNN)

Two CNN approaches are considered in this thesis work. The first CNN is designed with a single convolution layer and analyzed with different performance metrics. This Shallow CNN results in lower accuracy and overfitted with validation and testing metrics. The second Deep CNN is designed with multiple CNN layers, which obtained higher classification accuracy and reduced misclassification. Finally, results are

statistically compared, and chose the right CNN to convert to the vehicle detection module. The importance of CNN is discussed in detail as follows.

The CNN technique is a type of forward-feed artificial neural network used to analyze visual images by processing data with grid-like topology. The neurons in the network have learnable weights and biases. Every neuron receives several inputs, takes a weighted sum over them, passes it through an activation function, and responds with an output [28].

Convolutional filters, nonlinear activation functions, pooling, and backpropagation are applied to learn filters that can detect lower-level layers of the network such as edges, corners, and blob-like structures to high-level objects. The types of layer used to build convolutional neural networks are:

- Convolutional
- Activation
- Pooling
- Fully-connected
- Batch normalization
- Dropout

4.1.1 Convolutional Layer

CNN accepts an input, applies a convolution layer, then an activation layer, a fully-connected layer, and finally a softmax classifier to obtain the output classification. It accepts input size volume as height *x* width *x* depth. The critical parameters are the number of filters, which controls the depth of the output volume, the size of the filters

used for convolution, stride, and zero-padding.

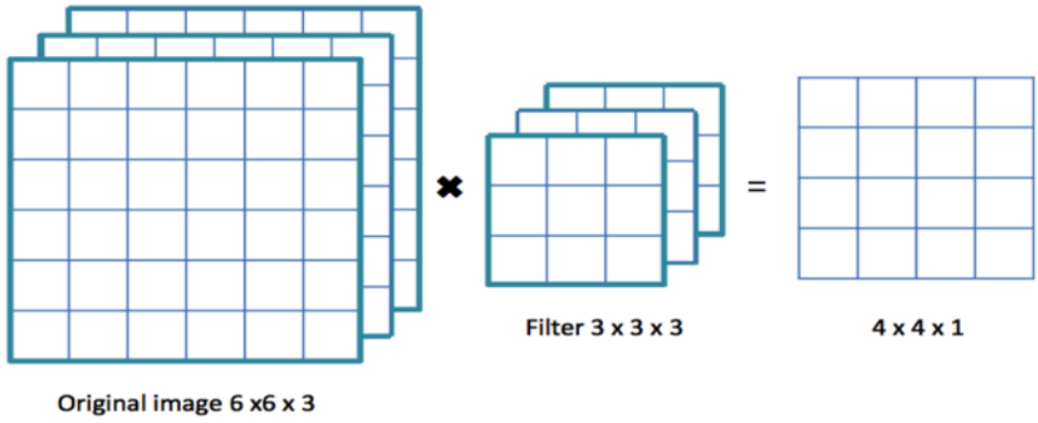


Figure 4. Process of 3D convolution in CNNs [28]

Figure 4 illustrates the process of 3D convolution used in CNNs. For instance, the input size (6 x 6 x 3) is convolved with filter 3 x 3 x 3 separately. Convolution of input with one filter produces one output feature, and with three filters independently produce three features. Each filter is moved from left to right, one element at a time starting from the top-left corner of the input. Once the top-right corner is reached, the filter is moved one element in a downward direction, and again the filter is moved from left to right, one element at a time. This process is repeated until the filter reaches the bottom-right corner.

Here, N = 6 and k = 3, the filter can take four unique positions from left to right and four unique positions from top to bottom. Corresponding to these positions, each feature in the output will contain 4 x 4 (i.e., (N-k+1) x (N-k+1)) elements. The obtained 2D output size is 4 x 4 x 1. The sample visualization of third convolution layers accepts an input size (50 x 50) with 64 filters of size (3 x 3) of Deep CNN implementation is shown in Figure 5. The model extracts the features from the image into more general

concepts, which are used to make a classification better.

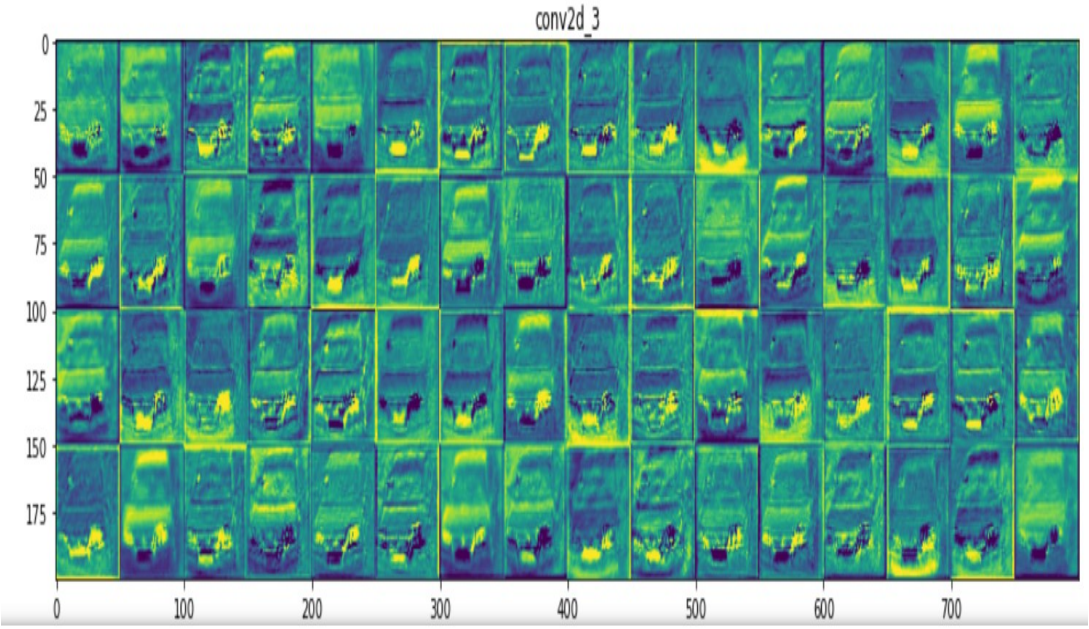


Figure 5. Visualization of the third convolutional layer with 64 filters from Deep Color CNN

4.1.2 Activation Layer

Rectified Linear Unit (ReLU) activation function is a piecewise linear function that will output the input directly if it is positive; otherwise, it will output zero. It overcomes the vanishing gradient problem, allowing models to learn faster and perform better [28]. Figure 6 shows the activation output after the third convolution layer from the Deep Color CNN model. The layer accepts an input of size (50 x 50) with 64 filters of size (3 x 3). The blank image shows the output is zero, and the filter is not learning any features from the image.

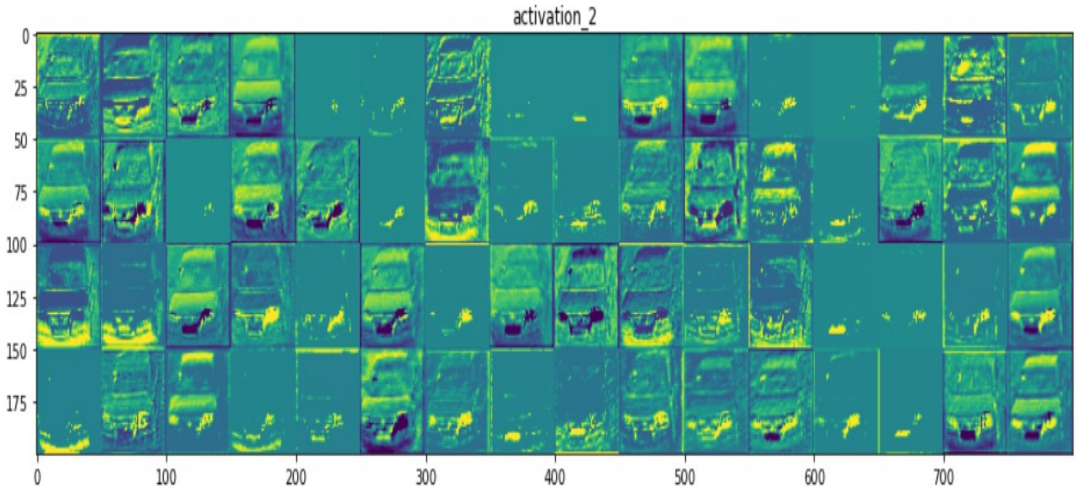
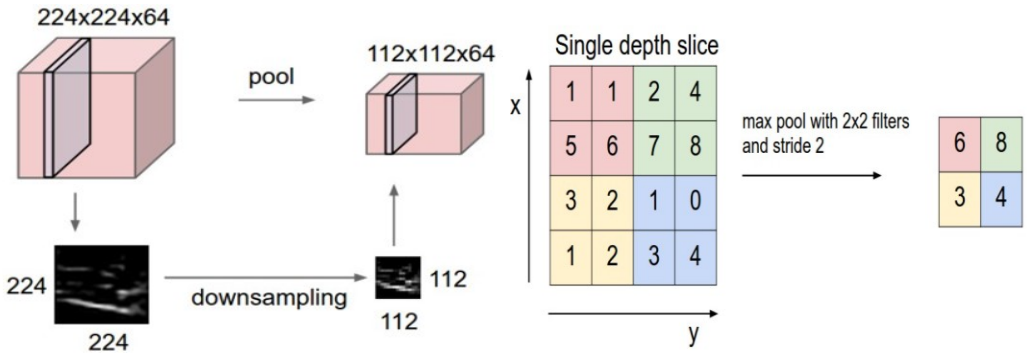


Figure 6. Visualization of Activation layer after the third convolutional layer with 64 filters from Deep Color CNN

4.1.3 Pooling Layer

The pooling layer reduces the feature map's dimensionality, which reduces the number of parameters and computation in the network. Pooling helps to control overfitting. It operates on max or average function. Max pooling is applied in the middle of the CNN architecture to maximize a specific filter region's maximum value. However, average pooling is used as the final layer to takes the average value in a filter region [29]. Figure 7a the input volume of size [224x224x64] is pooled with a filter size of two, stride value of two into output volume of size [112x112x64] by preserving volume depth. Figure 7b shows the down sampling operation max pooling with a stride of two. That is, each max is taken over four numbers (2x2 square). In this thesis work, Max pooling is applied in Deep CNN after the first set of convolutional layers, which the input volume of size [100 x100x 32] is pooled with a filter size of two, stride value of two into output

volume of size [50 x50x32] by preserving volume depth.

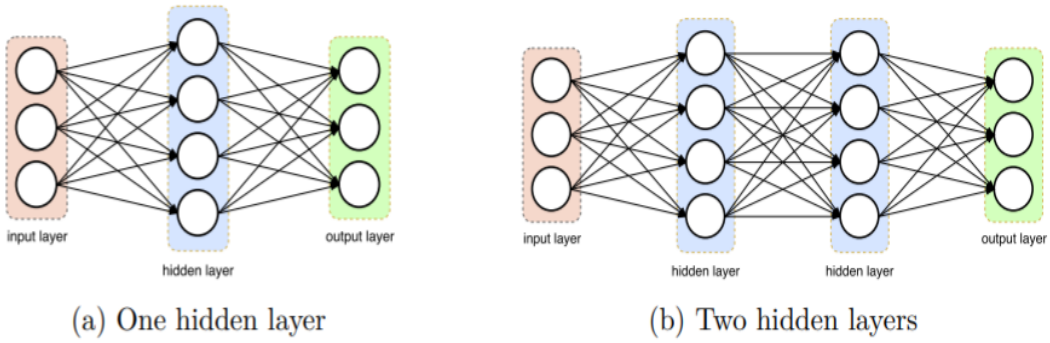


a) b)

Figure 7. a) Down Sampling and b) Maxpool operation [29]

4.1.4 Fully-connected Layer

Neurons in fully-connected layers are fully-connected to all activations in the previous layer, as shown in Figure 8. It is always placed at the end of the network before the softmax classifying layer, which will compute each class's final probabilities. The fully connected layer is responsible for the high-level reasoning in a convolutional neural network and is typically inserted after the convolutional layers [30]. In this thesis work, the fully-connected layer will help to classify eight-color and seven-type classes.



(a) One hidden layer (b) Two hidden layers

Figure 8. Fully connected layers in an artificial neural network [30]

4.1.5 Batch Normalization Layer

Batch normalization layers [28] are used to normalize a given input volume activations before passing it into the network's next layer. If *x* is considered as the mini-batch of activations, then the normalized $\hat{x}$ values can be computed using:

$$\hat{\mathrm{x}} = \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \varepsilon}} \qquad (4.1)$$

Here, μ_β and σ_β^2 are respectively mean and variance over each mini-batch of the training images, β. The error ε is set equal to a small positive value in the range of 10-7 to avoid dividing by zero. Applying (4.1) implies that the activations leaving a Batch Normalization layer will have approximately zero mean and unit variance, i.e., zero-centered. At testing time, the mini-batch $\mu_\beta$and σ_β^2 are replaced with running averages of $\mu_\beta$and σ_β^2 computed during the training process. This ensures that images through the network can be passed and obtain an accurate prediction without being biased by the $\mu_\beta$and σ_β^2 from the final minibatch passed through the network during the training time. The advantages are listed below.

- Prevent overfitting to obtain significantly higher classification accuracy.
- Provides regular training and validation loss curve as it helps to minimize loss.

4.1.6 Dropout Layer

The dropout layer is to reduce overfitting by explicitly altering the network architecture at training time. In the Shallow CNN, a 25% dropout layer is added after a convolutional layer, and a 50% dropout layer is added after a fully-connected layer to reduce overfitting.

The implemented CNNs are summarized as follows. The Shallow CNN architecture is a single convolution layer, followed by a 25% dropout layer to prevent overfitting. The output gets flattened into a single-dimensional network with 512 neurons, a 50% dropout layer, and the softmax classifier's final layer for eight color classification. Table 1 explains the layer configuration of the shallow architecture.

Table 1. The Shallow CNN Architecture

Type of layer	Output Size	Filter Size	Dropout Probability
Convolution	100x100x32	3x3	
Dropout	100x100x32		0.25
Flatten	320,000		
Fully connected	512		
Dropout	512		0.5
Softmax	8		

The Deep CNN model, which is inspired by [30] shown in Table 2, consists of two sets of convolution layer to learn local features like edges and corners in the images with 32 filters followed by nonlinear activation layer ReLU and batch normalization (BN) layer will normalize the activation to have zero mean and unit variance. This helps in effectively reducing the number of epochs to train the network. Then, the added Pooling layer with a size of 2x2 reduces the spatial dimensions of input, followed by a

dropout of 25% is applied to prevent overfitting. The same pattern is repeated for another set of convolution layers to learn more complex patterns in the data with 64 filters; then the network is flattened to a single-dimensional vector with 512 neurons followed by a fully-connected layer, activation layer ReLU, BN, and dropout of 50%. Finally, a softmax classifier is applied to classify the eight color classes.

Table 2. The Deep CNN Architecture

Type of layer	Output Size	Filter Size	Dropout Probability
Convolution => ReLU => BN	100x100x32	3x3	
Convolution => ReLU => BN	100x100x32	3x3	
Pooling=>Dropout	50x50x32	2x2	0.25
Convolution => ReLU => BN	50x50x64	3x3	
Convolution => ReLU => BN	50x50x64	3x3	
Pooling=>Dropout	25x25x64	2x2	0.50
Flatten	4,000		
Fully connected=> ReLU => BN=>Dropout	512		0.25
Fully connected	8		
Softmax	8		

Adding BN, Dropout, and Pooling layers in Deep CNN architecture helps achieve higher classification accuracy and lower loss in color and type classes. This section discussed the building blocks of CNN, and the implementation of CNNs is summarized. The deep learning model with higher accuracy will extract vehicle colors and types from input images/camera feed to find possible matches in the emergency alert message.

4.2 YOLO

YOLO stands for "You Only Look Once," which means that a single network just once is applied to the whole image. It uses convolutional neural networks (CNN) for object detection. YOLO works to predicts class labels and detects the location of objects at the same time. That is why YOLO can detect multiple objects in one image. YOLO [22], shown in Figure 9, divides images into a *S x S* grid and, for each grid cell, predicts bounding boxes, confidence for those boxes, and class probability. Then, bounding boxes are filtered with non-maximum suppression, which excludes some of them if confidence is low or another bounding box for this region with higher confidence.

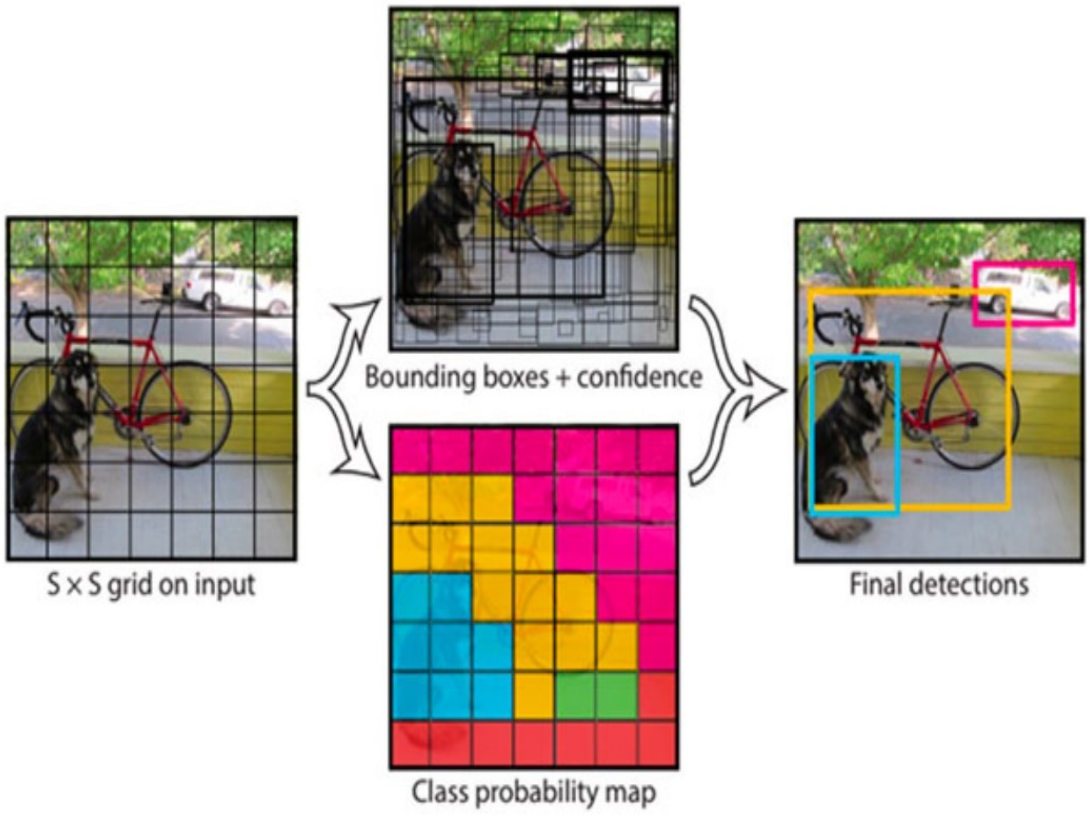


Figure 9. A simplified illustration of the YOLO object detector pipeline [22]

4.2.1 How YOLO Works

The network predicts five bounding boxes at each cell in the output feature map. The network predicts five coordinates for each bounding box, t_x,t_y,t_w,t_h, and t_o. If the cell is offset from the top left corner of the image by (c_x, c_y) coordinates, and the bounding box prior has width and height (p_w, p_h), then the predictions are calculated with (4.2) and (4.3), where, Intersection over Union (IOU) is the overlap rate between the bounding box detected by the system and the ground truth box. The BB_{gt} parameter is the ground truth box on the training labels. The BB_{dt} parameters is the detection bounding box shown in Figure 10.

$$p_r(object) * IOU(b, object) = \sigma(t_o) \tag{4.2}$$

$$IOU = \frac{area(BB_{dt} \cap BB_{gt})}{area(BB_{dt} \cup BB_{gt})} \tag{4.3}$$

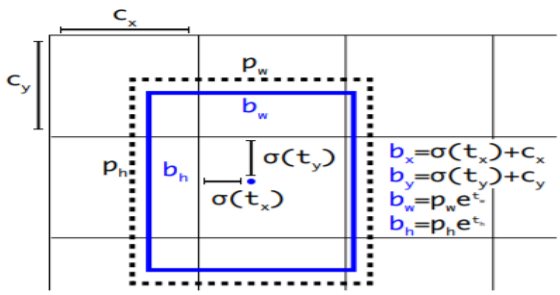


Figure 10. Bounding Box Prediction [28]

YOLOv3 has three different scales; at each scale, predict three anchor boxes. It is trained with the COCO dataset; the tensors at each scale are N × N × [3 × (4 + 1 + 80)] for the four bounding box offsets, one abjectness, and 80 class predictions [26]. YOLOv3 applies the k-means cluster to determine the priors (anchors). There are pre-select nine clusters whose width and height are (10×13), (16×30), (33×23), (30×61), (62×45), (59×

119), (116 × 90), (156 × 198), (373 × 326) that are split up evenly across three scales. Each group is assigned to a specific feature map above in detecting objects. YOLOv3 uses a new network called Darknet-53 for performing feature extraction, as Table 3 shown below. The network uses successive 3×3 and 1×1 convolutional layer. It has some shortcut connections now but relatively larger with 53 convolutional layers than the YOLOv2 with nineteen layers and YOLOv1 with 24 layers. This new network works more accurately on extracting features than Darknet-19 adopted in YOLOv2 [27].

Table 3. Darknet-53 [26]

	Type	Filters	Size	Output
	Convolutional	32	3 x 3	256 x 256
	Convolutional	64	3 x 3 / 2	128 x 128
1x	Convolutional	32	1 x 1	
	Convolutional	64	3 x 3	
	Residual			128 x 128
	Convolutional	128	3 x 3 / 2	64 x 64
2x	Convolutional	64	1 x 1	
	Convolutional	128	3 x 3	
	Residual			64 x 64
	Convolutional	256	3 x 3 / 2	32 x 32
8x	Convolutional	128	1 x 1	
	Convolutional	256	3 x 3	
	Residual			32 x 32
	Convolutional	512	3 x 3 / 2	16 x 16
8x	Convolutional	256	1 x 1	
	Convolutional	512	3 x 3	
	Residual			16 x 16
	Convolutional	1024	3 x 3 / 2	8 x 8
4x	Convolutional	512	1 x 1	
	Convolutional	1024	3 x 3	
	Residual			8 x 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

An input image size of 320x320 pixels YOLOv3 [26] can perform predictions in 22 milliseconds with 28.2 mAP (mean Average Precision). This is as accurate as of the SSD model but much faster. YOLOv3 achieves good performance. The model can perform predictions in 51 milliseconds on a TitanX GPU with 57.9 mAP@50, compared to 57.5 mAP@50 in 198 milliseconds by RetinaNet. YOLOv3 is a multi-scale target detection algorithm that can consider both small and large targets and performs detection for small targets.

Hence, the YOLOv3 is used in this thesis work. The advantages are:

- More straightforward structure of the network
- Much faster than R-CNN and able to process streaming video at 150fps in real-time with less than 25 milliseconds of latency
- Maintaining a proper accuracy range

YOLO helps to classify and localize multiple objects in a single image with bounding boxes surrounded on it. This provides information of the object's name, such as a car, truck, bus, and other vehicles, with a probability score. Detecting vehicles with bounding boxes will help spot the vehicles from the child abduction involved, bringing child and elderly person safe home.

5 METHODOLOGY

This chapter discusses the overall workflow consists of the various processes that include vehicle data collection, Pre-processing, Classification, Detection, and Optical Character Recognition (OCR). When an emergency alert signal is broadcasted, the vehicle color classification model will check for the possible matches of color, and then the matched vehicle color will pass to the vehicle type model to find possible matches of vehicle type. The matched vehicle type and color with bounding boxes snapshot will be saved. License Plate vehicle images will pass to the OCR model to find possible characters and numbers in the license plate to give vehicle identification as shown in Figure 11. The whole process will help find possible matches involved in these emergency alerts for child/older adults' safe return.

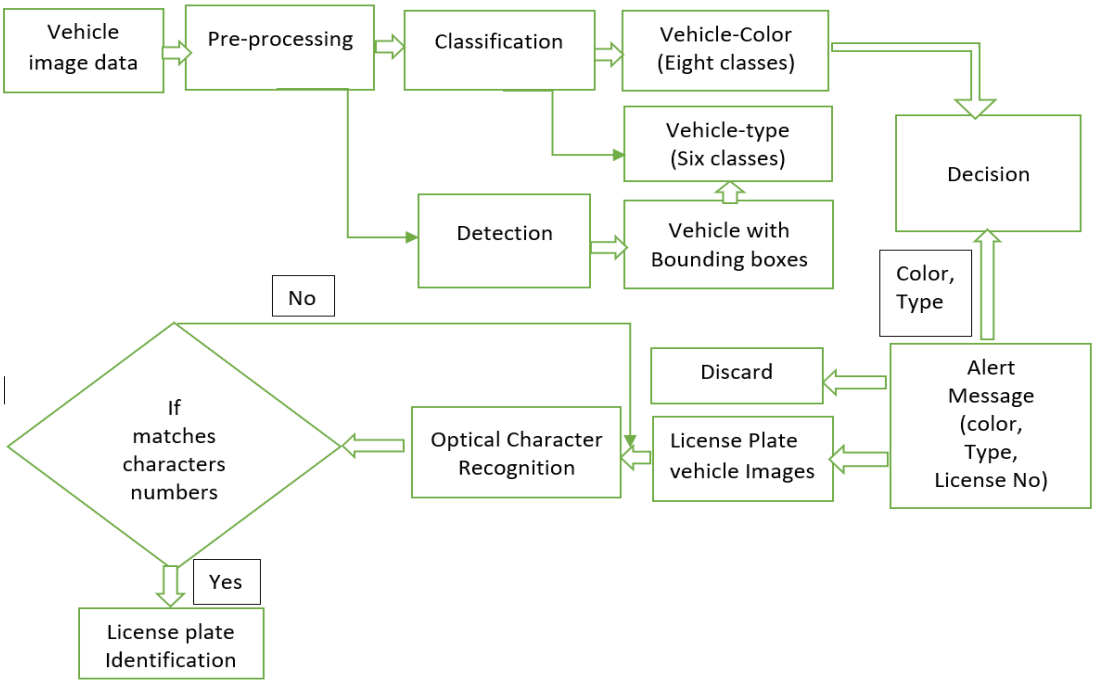


Figure 11. Workflow of the execution model in response to alert emergency signal

5.1 Dataset

The Machine Learning model's first step is to collect the best vehicle image datasets considering different environmental conditions like lighting and illumination with different viewpoints. Datasets used as training for vehicle color and vehicle type classification are discussed in the following subsections.

5.1.1 Vehicle Type Recognition Dataset

Tampere University (TAU) Vehicle Type Recognition Competition Dataset (Kaggle dataset) [31] and CompCars [32] are used to evaluate vehicle type models. The data has been collected from the open images datasets, an annotated collection of over nine million images. The subset of open images contains only vehicle categories among the total of 600 object classes. Vehicle types total seventeen categories, including cars, bicycles, ambulances, and other types that encompass the dataset. Collection of four types includes Van, Motorcycle, Bus, and Truck were used for this work. CompCars [32] dataset is used for SUV and Sedan types. It contains 50,000 surveillance car images suffered from considerable variation in illumination due to differences in traffic imaging conditions, making the recognition of vehicles from the frontal-view surveillance data more challenging. The actual size of the original images is 800x850 pixels in resolution. It also includes twelve types of cars: MPV, SUV, hatchback, sedan, minibus, fastback, estate, pickup, sports, crossover, convertible, and hardtop convertible. Figure 12 shows the number of images collected for vehicle types. The images are taken from different viewpoints, significant variation in illumination and haze making data more challenging.

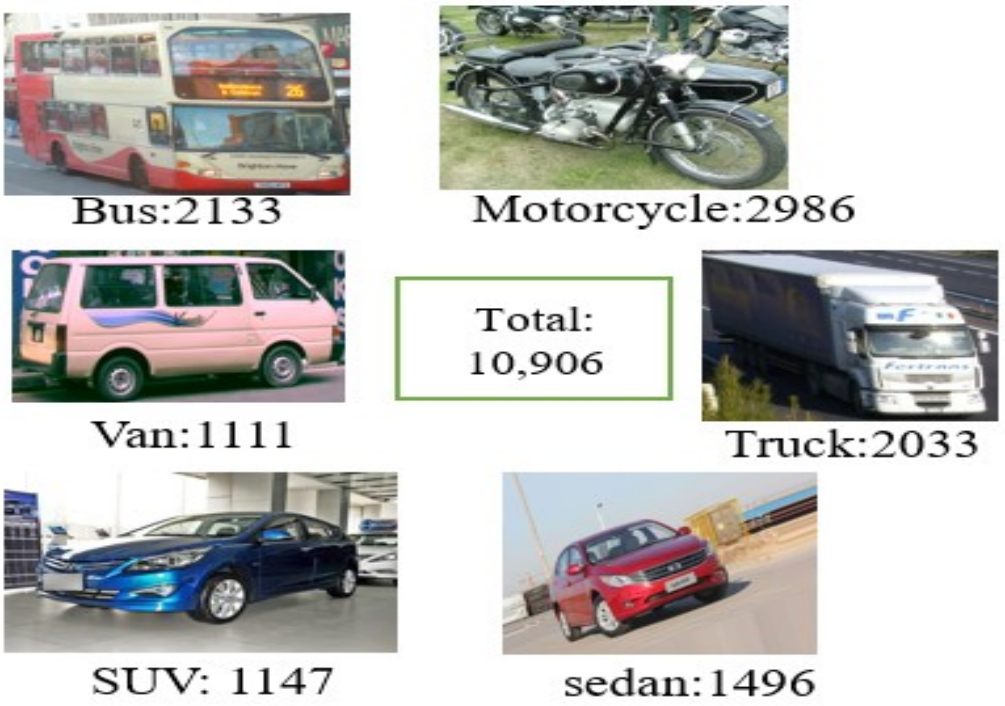


Figure 12. Examples of number of images in each type categories [31] [32]

5.1.2 Vehicle Color Recognition Dataset

The authors in [5] used a dataset consisting of 15,601 vehicle images with eight classes of the vehicle color, which are black, blue, cyan, gray, green, red, white, and yellow. The dataset is very challenging due to the noise caused by illumination variation, haze, and overexposure. Images are taken in front of a high-definition camera, and Figure 13 shows the total number of images in each color category.

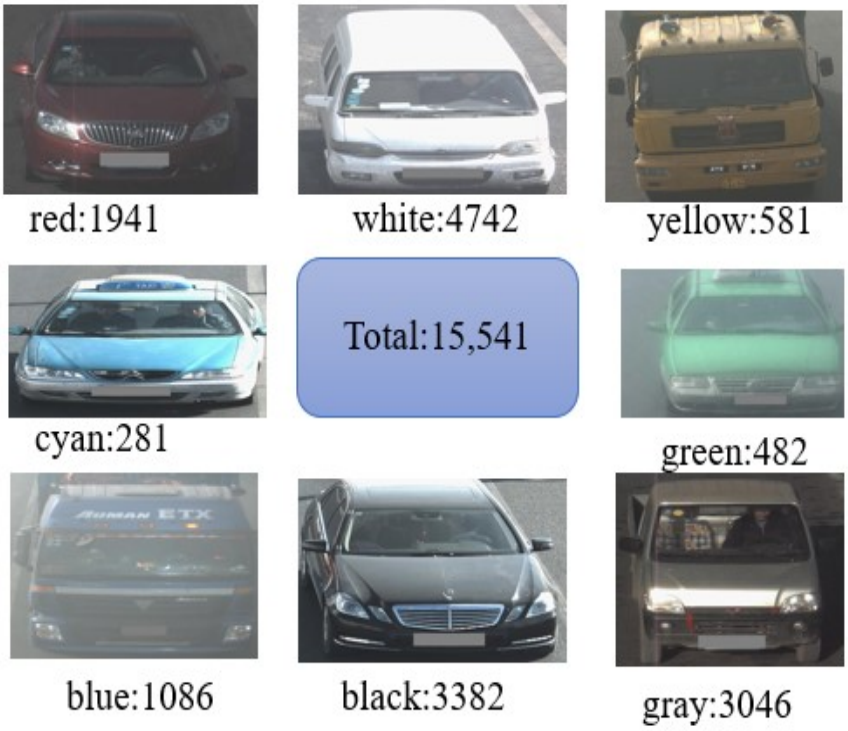


Figure 13. Examples of number of images in each color classes [5]

5.2 Data Pre-processing

Data pre-processing aims to improve vehicle images by removing haze and data augmentation to give balanced training samples, improving the type and color classification performance. The pre-processed image is fed to the Deep learning models to obtain classification results.

5.2.1 Haze Removal

Hazy weather conditions pose the most common problem in surveillance systems, reducing the clarity in traffic images resulting in low accuracy of vehicle recognition and detection. The dark channel prior (DCP) is based on the property of “dark pixels,” which have a very low intensity in at least one-color channel, except for the sky region.

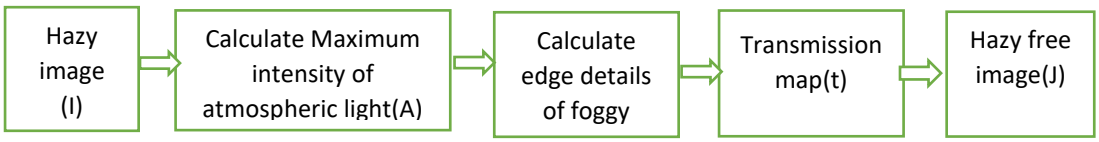


Figure 14. Workflow of DCP algorithm

The workflow of DCP shown in Figure 14 includes calculating the maximum intensity of atmospheric light A, to find the brightest pixel in the dark channel, edge details of the foggy images, and a transmission map to obtain haze-free images. The algorithm is implemented using OpenCV and Python. The formulation of haze images in (5.1) from [6] and [7] as:

$$I(x) = J(x)t(x) + A[1 - t(x)] \quad (5.1)$$

, where x is the pixel coordinate, I denotes the observed image, J is the haze-free images, A is the atmospheric light, and t is the transmission map. Atmospheric light is estimated by identifying the top 0.1% brightest pixel in the brightest dark channel, and the highest intensity is identified as the atmospheric light. The transmission map is estimated by (5.2) as stated in [5] as:

$$t(x) = 1 - \omega \min_{c\in\{r,g,b\}}\left(\min_{c\in\Omega(x)}(D_c(y)A_c)\right) \quad (5.2)$$

, where ω is the defogging parameter in the range of 0 to 1. Adjusted ω as 0.95. By knowing atmospheric color (A) and transmission map t, haze removal result J is given by (5.3) from [7] as:

$$J(x) = I(x) - A\,t(x) + A \quad (5.3)$$

The haze-free images for type bus shown in Figure 15 and black color in Figure 16 using python and OpenCV.

a) b)

Figure 15. Examples of a) hazy bus input and b) haze-free bus image output

a) b)

Figure 16. Examples of a) hazy black color input and b) haze-free black color image output

5.2.2 Data Augmentation

Data Augmentation involves the process of creating new training samples by manipulating the original data. The white color training samples have more samples than other classes such as cyan, green, and yellow. Similarly, the type of motorcycle category's training sample has too many samples and insufficient SUV and Van samples. Images are transformed using Keras Image Data generator [9] to increase the training samples. The Image Transformation shown in Figure 15 shows an input van type image

to be randomly rotated to +/-30 degrees, zoomed in and out of 10%, sheared range of 20%, and flipped horizontally. This will help to increase the generalization and reduce the overfitting of the model. Similar transformations are implemented to increase training samples for color datasets as white color classes have more samples than other classes.

Figure 17. Example of Data augmented formation of van type image sample

5.3 Data Partitioning

In the vehicle color classification experiment, there are 12,115 training samples, and the data is partitioned into training and testing using 70% of the data for training and 30% for testing. The network is trained for 8,480 samples and validates on 3,635 samples. In the vehicle type classification experiment, there are 17,638 training samples, and the data is partitioned into training and testing using 70% of the data for training and 30% for testing. The network is trained for 12,346 samples and validates on 5,292 samples. The summary of data splitting in Table 4 is illustrated below.

Table 4. Summary of Data Splitting

Data splitting	Vehicle Color	Vehicle Type
Total number of samples	12,115	17,638
Training	8,480	12,346
Testing	3,635	5,292

5.4 Shallow CNN Architecture

The shallow CNN Architecture, as shown in Table 5, consists of a single convolution layer, followed by a 25% dropout layer to prevent overfitting, which then gets flattened into a single-dimensional network with 512 neurons, a 50% dropout layer, and the final layer of softmax classifier for eight color classification. This architecture is implemented for the vehicle type classification model, except the final layer of softmax classifier for six vehicle types.

Table 5. Summary of Shallow Color CNN

Type of layer	Output Size	Parameters
Convolution	100x100x32	896
Dropout	100x100x32	0
Flatten	320,000	0
Fully connected	512	163,840,512
Dropout	512	0
Softmax	8	4104
Total Trainable Parameters	163,845,512	

The learnable parameters can be calculated using equation (5.4) and (5.5), which is shown below, and the details of each layer's parameter calculation are illustrated in Table 6. The same calculation is repeated for all the layer's parameter calculations except the shallow type CNN for six vehicle types. The number of parameters in a CONV layer is:

$$\big((m * n * d) + 1\big) * k) \tag{5.4}$$

Here, *m* is the shape of the filter's width, *n* is the shape of the filter's height, *d* is the number of filters in the previous layer, and *k* denotes the previous layer's filters**.** The number of parameters in a fully-connected and Softmax layer is:

$$\big((c * p) + 1\big) * c) \tag{5.5}$$

The *c*-parameter is the current layer's neurons, *p* is the previous layer's neurons, and one is the bias term.

Table 6. Each layer's parameter calculation

Shallow Color CNN		**Parameters**
CONV	$m * n = (3 \times 3)$ $d = 3$ $k = 3$	896
Fully connected	$c = 512$ $p = 320{,}000$	163,840,512
Softmax	$c = 8$ $p = 512$	4,104
Total Trainable Parameters	163,845,512	
Shallow type CNN		
Softmax	$c = 6$ $p = 512$	3,078
Total Trainable Parameters	163,843,590	

5.5 Deep CNN Architecture

The Deep CNN Architecture, which is inspired by [28] shown in Table 7, consists of two sets of convolution layer to learn local features like edges and corners in the images with 32 filters followed by nonlinear activation layer ReLU and batch normalization (BN) layer will normalize the activation to have zero mean and unit variance. This helps in effectively reducing the number of epochs to train the network. Then, the added Pooling layer with a size of 2x2 reduces the spatial dimensions of input, followed by a dropout of 25% is applied to prevent overfitting. The same pattern is repeated for another set of convolution layers to learn more complex patterns in the data with 64 filters; then the network is flattened to a single-dimensional vector with 512 neurons followed by a fully-connected layer, activation layer ReLU, BN, and dropout of 50%. Finally, a softmax classifier is applied to classify eight-color classes. The same

architecture is implemented for the type classification model with a dropout of 25% at the fully-connected layer with the final layer of softmax classifier for six type classification.

Table 7. Deep CNN Architecture

Layers	**Deep CNN Architecture**	
	Output shape	Parameter
Input	(100 x 100 x 3)	0
Conv	(100 x 100 x 32)	896
BN	(100 x 100 x 32)	128
Conv	(100 x 100 x 32)	9,248
BN	(100 x 100 x 32)	128
Max-Pool	(50 x 50 x 32)	0
Dropout	(50 x 50 x 32)	0
Conv	(50 x 50 x 64)	18,496
BN	(50 x 50 x 64)	256
Conv	(50 x 50 x 64)	36,928
BN	(50 x 50 x 64)	256
Max-Pool	(25 x 25 x 64)	0
Dropout	(25 x 25 x 64)	0
Flatten	40,000	0
Dense	512	20,480,512
BN	512	2,048
Dropout	512	0
Dense	8	4,104
Softmax	8	0
Total Learning Parameters	20,484,616	

The learnable parameters can be calculated by using the equations (5.4) and (5.5), and the details of each layer's parameter calculation are illustrated in Table 8. The same calculation is repeated for all the layer's parameter calculations except the shallow type CNN for six vehicle types.

Table 8. Each layer's parameter calculation

Shallow Color CNN		**Parameters**
CONV	$m * n = (3 \times 3)$ $d = 3$ $k = 3$	896
Fully connected	$c = 512$ $p = 40{,}000$	20,480,512
Softmax	$c = 8$ $p = 512$	4,104
Total Trainable Parameters	20,484,616	
Shallow type CNN		
Softmax	$c = 6$ $p = 512$	3,078
Total Trainable Parameters	20,483,590	

5.5.1 Convolutional Layer

- Filters: Filters take an integer value and the most variable parameter of the convolutional layer. The filter count of this layer was 32.
- Kernel Size: It is a tuple or list of two integers, specifying the height and width of the 2D convolution window. In general, a Convolutional layer uses a smaller filter size such as 3x3 or 5x5 [28]. For this layer, a 3x3 kernel size was used.

- Stride: Stride tells the number of pixels by which the window moves after each operation. For this layer, it was kept at the default value 1.
- Padding: The same padding is used to ensure the size of the convolution operation matches the input.
- Activation Function: Each convolutional layer is associated with the ReLU activation function, the most popular activation function used in deep learning architectures [28].
- Dropout Rate: It is defined as the probability at which outputs of the layer are dropped out. The dropout probability of 25% for the convolutional layer and 50% for the fully connected layer was dropped, preventing overfitting.

5.5.2 Max Pool Layer

Max Pool Layer is added for the Deep CNN model to reduce the spatial size, width, and height, of the input volume. The number of parameters and computation in the network pooling reduces and helps to control overfitting.

- Kernel Size: The size of the pooling operation or filter is smaller than the size of the feature map; specifically, it is almost always 2×2 pixels. This configuration was implemented for this work.
- Stride: Stride is set to 2 for the max-pooling layer, which means the filter will move two pixels at a time.
- Dropout Rate: The rate for the dropout layer was given 0.25 for Deep Color CNN work, which means 25% of the output of the layer was dropped out, and it helped prevent overfitting.

5.5.3 Optimizers

Optimizers are algorithms used to change neural networks' attributes, such as learning rate, to reduce the losses [33]. This research experiments with Stochastic Gradient algorithm (SGD) and Root Mean Square (RMS Prop) algorithms, which are some of the most popular optimization algorithms used in Deep Learning. The SGD algorithm helps the neural network initialize and update the weights in all the layers. RMSProp improves AdaGrad by controlling the effect of the accumulating squared gradients. It adds a new coefficient to control the historical gradients' effect to constantly updated in each iteration [34]. RMS prop is used for Shallow CNN and SGD for Deep CNN based on higher training accuracy and lower losses.

5.5.4 Hyperparameters

Three hyperparameters that played a vital role in the learning algorithm's performance are Batch Size, Number of Epochs, and the Learning Rate. Final hyper-tuning parameters are selected based on higher training and validation accuracy and lower training and validation losses.

5.5.5 Batch Size

The batch size is the number of samples to work through before updating the internal model parameters. The Shallow CNN model is tested with batch sizes of 32, 64, and 128, where 128 gives higher training accuracy and lower validation loss values. A batch size of 64 gives the best results for the Deep CNN model.

5.5.6 Number of Epochs

The number of epochs is the number of times the whole training dataset passes through the neural network. The model is tested with several epochs of 20, 40, 60, and

80. Finally, 80 epochs for the Shallow CNN and 30 epochs for the Deep CNN yield higher training accuracy and lower validation loss values.

5.5.7 Learning Rate

The learning rate is one of the critical hyperparameters to train a neural network for gradient descent algorithm. The parameter scales the magnitude of weight updates to minimize the network loss function [33]. If the learning rate is too low, the training process will progress very slowly, resulting in slow updates to the weights. If the value is too large, it may result in learning a suboptimal set of weights too fast or an unstable training process. Experimented the learning rates as 0.01, 0.001, and 0.0001 values, and the learning rate value of 0.0001 performed the best for both Shallow and Deep CNN.

5.6 Vehicle Detection

With the YOLO development from v1 to v3, YOLOv3 has become one of the most popular object detection methods. It can detect relatively small objects from a complex background with less computational consumption [23]. The YOLO model is trained on the COCO dataset (common objects in context) from Microsoft and can detect 80 familiar objects. Using pre-trained models helps avoid unnecessary computational costs and helps take advantage of already biased weights without losing already learned features. Therefore, the pre-trained YOLOv3 was utilized for testing the vehicle detection model, as shown in Figure 18. The input images are passed into function *CV2.dnn.blobFromImage()*, which will convert the input image into a standard size of (416 x 416) that YOLO used to extract the features. Then, feed the YOLO Detector's blob image to extract the bounding box's coordinates and dimensions. The confidence shows

the absence or presence of an object of any class. A low confidence value specifies that the network misinterprets the object.

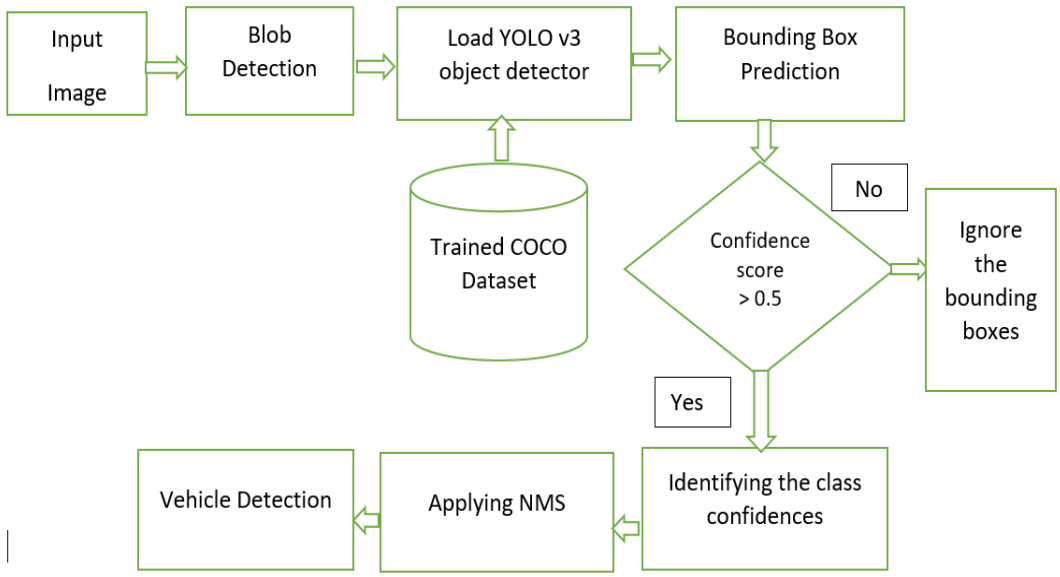


Figure 18. The testing workflow using YOLO

The minimum probability of filtering weak detections has a default value of 50% (0.5). Applying non-maxima suppression suppresses significantly overlapping bounding boxes, keeping only the most confident ones. NMS also ensures that they do not have any redundant or extraneous bounding boxes. Non-maxima suppression threshold with a default value of 0.3 was used. Vehicle images with bounding boxes of types or colors will check with possible matches in the emergency signal to safely return child/older adult.

5.7 Optical Character Recognition

The captured vehicle image is passed to an optical character recognition model to extract characters and numbers to find possible matches in the amber/silver alert signal. Image pre-processing is an essential task in an image analyzing process. Without proper

pre-processing, the recognition will be ineffectual or may give inaccurate results during OCR. Image pre-processing includes image resizing, Grayscale conversion, and image filtering are implemented using OpenCV and Python. The process implementation is discussed in the following sections.

5.7.1 Image Resizing

Resizing images includes changing the dimensions of width and height by preserving the aspect ratio. This will help to avoid bigger resolution images and to maintain the number plate remains in the frame.

5.7.2 RGB to Grayscale Conversion

Grayscale conversion will help identify the number plate's outline easier and make further processing simpler. The gray color is when the red, green, and blue components all have equal intensity in RGB space. Therefore, it carries only intensity information for each pixel [35] as shown in Figure 19.

a) b)

Figure 19. Examples of a) RGB color input and b) Grayscale image output [36]

5.7.3 Bilateral Filtering

The bilateral filter is a non-linear, edge-preserving, and noise-reducing smoothing filter. It replaces the intensity of each pixel with a weighted average of intensity values from nearby pixels. This weight can be based on a Gaussian distribution. The weights depend not only on Euclidean distance of pixels but also on the radiometric differences includes range differences, such as color intensity, depth distance, and other parameters, to preserves sharp edges [37]. This will make the image blurred to remove unwanted details in the image, as shown in Figure 18.

a) b)

Figure 20. Examples of a) RGB color input and b) Grayscale and blurred image output [36]

5.7.4 Canny Edge Detection

The Canny Edge Detection algorithm [38] is the most popular algorithm that aims to find firm edges from the image. The various steps include noise reduction, finding intensity gradient of the image, and non-maximum suppression and hysteresis thresholding. The *cv2.canny()* was used to find the edges with an intensity gradient more

than the minimum threshold value and less than the threshold value, as shown in Figure 21. The minimum value is 30, and the maximum value of 200 was selected after trial and error to find strong edges.

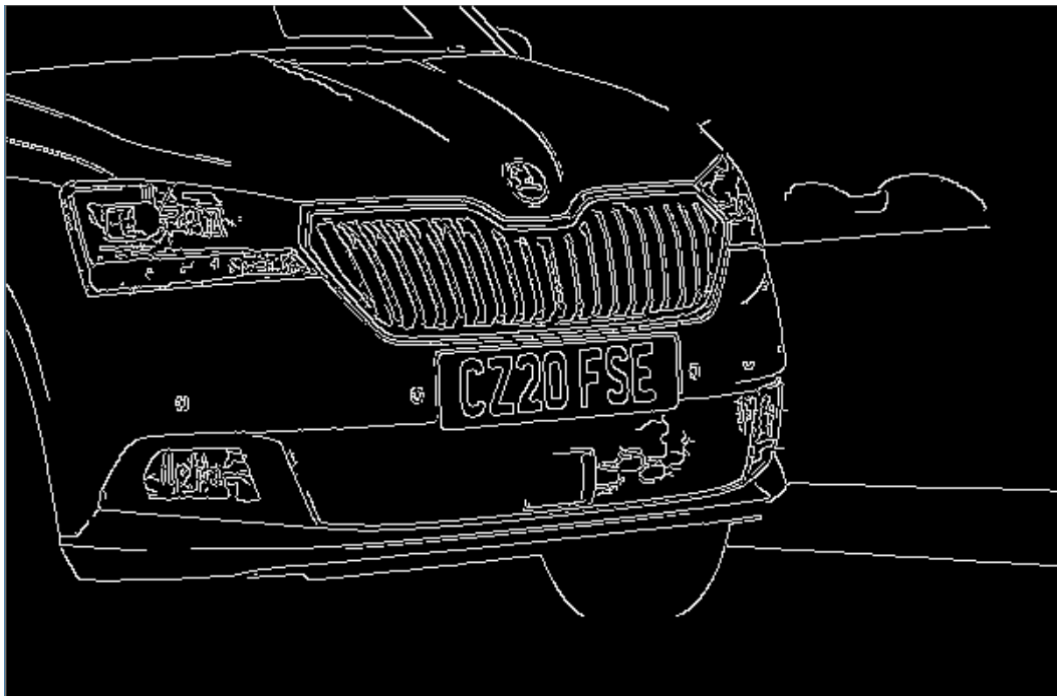


Figure 21. Canny edge detection [36]

5.7.5 Contours

Contours are the line joining all the points along the boundary of an image with the same intensity. Contours help shape analysis, find the size of the object of interest, and object detection [39]. OpenCV *findContour()* function was used in extracting the contours from the image. It helps to check the rectangle shape contour with four sides as the license plate is a four-sided rectangle, as shown in Figure 22.

Figure 22. Contour extraction [36]

5.7.6 Character Segmentation

The detected license plate is passed to a Python-tesseract tool [40], one of the powerful tools used for character recognition. This will recognize and read the text embedded in images, as shown in Figure 23. The final extracted text of all the images will check with possible matches in the emergency signal to safely return a child/elderly person.

a) b)

Figure 23. Example of a) Contour extraction input and b) Detected string output [36]

The vehicle classification, vehicle detection, and OCR models are executed in response to the alert signal. When an emergency alert signal is broadcasted, the vehicle color classification model will check for the possible matches of color, and then the matched vehicle color will pass to the vehicle type model to find possible matches of vehicle type. The matched vehicle type and color with bounding boxes snapshot will be saved. The License Plate vehicle images will pass to the OCR model to find possible characters and numbers in the license plate to give vehicle identification. The whole process will help find possible matches in these emergency alerts for child/older adults' safe return.

6 EXPERIMENTS AND RESULTS

6.1 Programming Environment

The Python 3.7.3 programming language using Anaconda was used to develop the classification and detection models, pre-processing, training, and testing. Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, and other applications) that aims to simplify package management and deployment [41]. Python contains special deep learning libraries such as Keras, a powerful and easy-to-use free open-source platform for developing and evaluating models. It wraps the efficient numerical computational libraries Theano and TensorFlow. Keras with TensorFlow backend was used in this thesis.

- Workstation:
  - Intel (R) Core (TM) i7-8650U CPU Processor running at 1.90 GHz clock frequency
- NVIDIA Tesla V100 specifications are:
  - Architecture: Volta
  - Tensor Cores: 640
  - GPU Cores: 5,120
  - Memory: 16GB HBM2
  - Memory Bandwidth: 900GB/sec
- HiPE Server:
  - Dual Intel Xeon Gold / 2.3GHz / 24.75M Cache
  - Num. of Cores: 18 Cores / 36 Threads

- Memory: 16GB RDIMM x12 Data Width (192GB)

6.2 Training and Evaluation Results

This thesis research aims to develop a Deep Learning model to classify vehicle colors and vehicle types faster and accurately. Experimentation was performed on Shallow and Deep CNN architectures to get the most optimized and efficient architecture with higher accuracy and lower validation losses. Initially, Shallow CNN was designed with a single Convolution Layer as discussed in previous chapter and tested on the pre-processed dataset. Different combinations of hyperparameters, including Learning Rate, Number of Epochs, Image Resolution, Batch Size, were tested, compared, and analyzed to get a better reliable model. The obtained results are evaluated with higher training accuracy and lower validation losses.

6.2.1 Shallow Color CNN Architecture

The designed Shallow CNN Color Architecture is compiled using the sparse categorical cross-entropy as loss function and *RMSprop* as optimizer using a learning rate of 0.0001. Hyper-tuned parameters include changing batch sizes of 32, 64, and 128, several epochs of 20, 40, 60, 80, and a learning rate of 0.001 and 0.0001 with image sizes 100x100 and 200x200. Final parameters are selected based on higher training and validation accuracy and lower training and validation losses. The network is trained for 8,480 samples and validates on 3,635 samples with an image resolution of 100x100, 80 epochs using mini-batch sizes of 128. Results are evaluated using precision, recall, F1-score, confusion matrix, model accuracy, and loss curves. Table 9 shows the precision, recall, F1-score value of each class. The cyan, yellow, and red color indicates the highest value of precision, recall and F1-score, among other classes. The yellow, red and cyan

color looks bright whereas the gray and black color is too light due to the sun reflection on metallic part of the car in most of the data samples. The average accuracy obtained for the Shallow Color CNN is 94%.

The 38 gray color samples are misclassified as white, and 36 samples are misclassified as black are shown in Figure 24. The gray color data samples are overexposure to sunlight during daytime and dark during nighttime. This makes a single convolutional layer difficulty to learn complex patterns, and it makes mistakes in classifying both colors.

Table 9. Classification report of Shallow Color CNN

Color Classes	precision	recall	F1-score
black	0.90	0.93	0.91
blue	0.95	0.95	0.95
cyan	0.98	0.99	0.99
gray	0.85	0.82	0.83
green	0.97	0.92	0.95
white	0.91	0.93	0.92
yellow	0.99	0.98	0.99
red	0.98	0.99	0.99

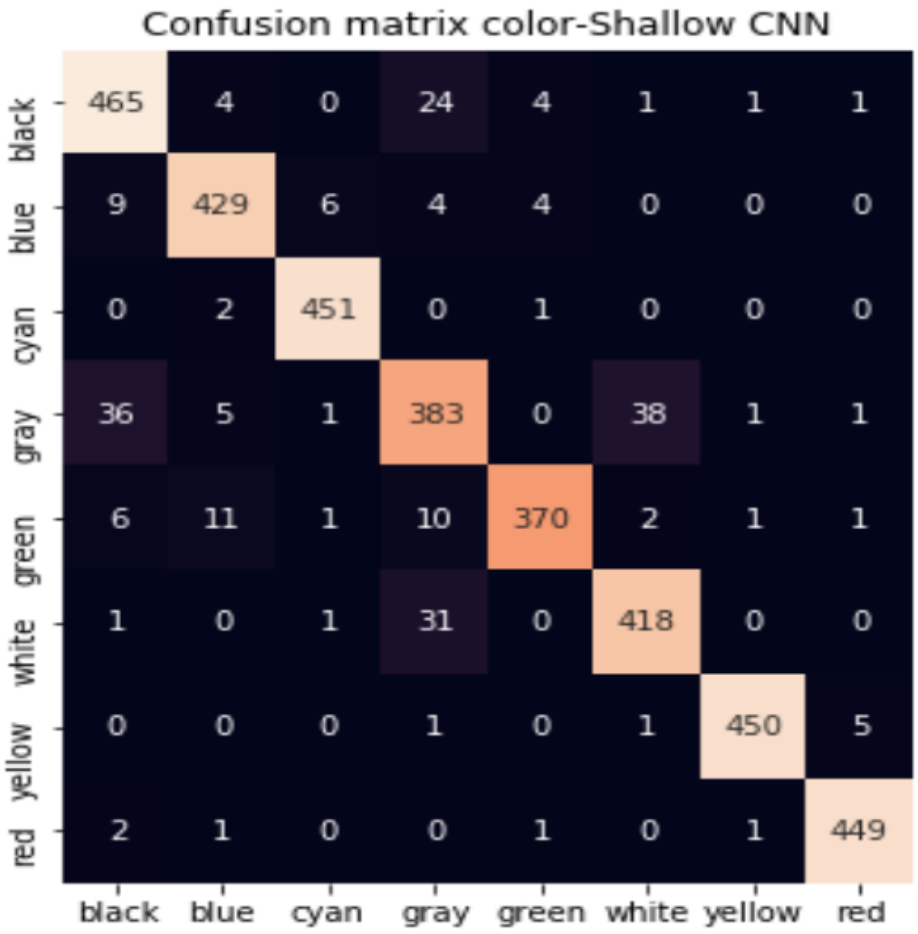
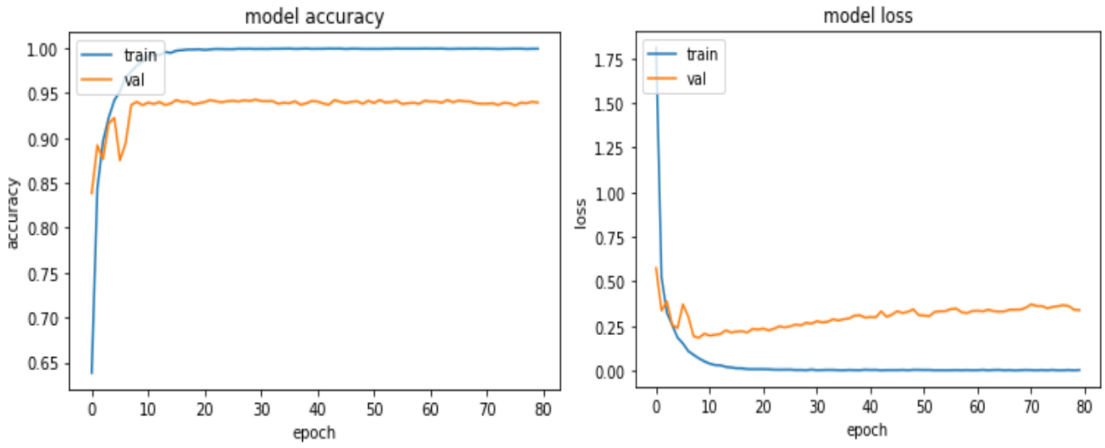


Figure 24. Confusion matrix of test samples for Shallow Color CNN

The Shallow Color CNN model is shown in Figure 25a obtains 94% average classification accuracy; however, the model is overfitting. Figure 25b shows the training loss continues to decrease with experience and the validation loss decreases to a point and begin increasing again and have a large gap with the training loss. This tells the model is overfitted. The overfitting problem can be solved by adding a batch normalization layer and dropout layer, which are used as regulators



a) b)

Figure 25. a) Accuracy curves for Shallow Color CNN b) loss curves for Shallow Color CNN

6.2.2 Shallow Type CNN Architecture

The Shallow Type CNN was designed with a single Convolution Layer and tested on the pre-processed dataset. The network is trained for 12,346 samples and validates on 5,292 samples with an image resolution of 100x100, 30 epochs using mini-batch sizes of 64. Results are evaluated using precision, recall, F1-score, confusion matrix, model accuracy, and loss curves. Table 10 shows the precision, recall, F1-score value of each class. The average accuracy obtained for the Shallow Type CNN is 86%. The truck class shows the lowest precision, recall, and F1-score values when compared to other types of classes.Sedan and SUV data samples are nature web based data samples and less exposed to light so that the model shows higher precision and recall value. The van and truck data samples are high challenging includes blurred and low resolution which makes the single convolutional layer is not able to learn more complex patterns and gives lowest results.

The confusion matrix indicates significant misclassification with the truck and van types. The misclassification of truck and van is shown in Figure 26, indicating the most of the two categories have the same features. Therefore, the model has difficulty in classifying due to changes in weather conditions and overexposure.

Table 10. Classification report of Shallow Type CNN

Type Classes	precision	recall	F1-score
Bus	0.85	0.81	0.83
Truck	0.77	0.75	0.76
Motorcycle	0.85	0.80	0.82
Van	0.81	0.94	0.87
SUV	0.95	0.96	0.96
Sedan	0.97	0.93	0.95

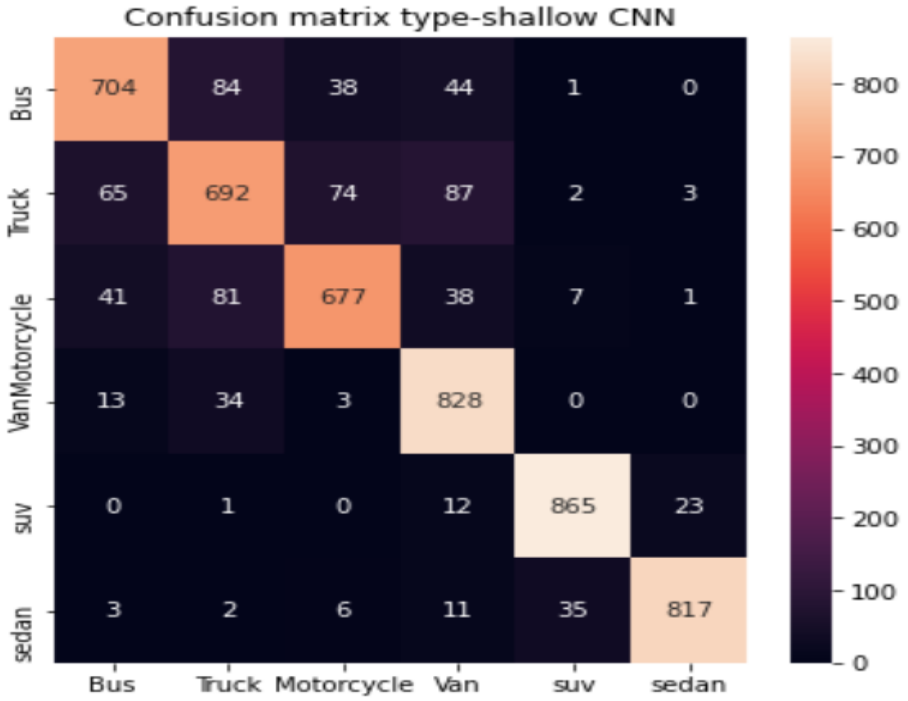
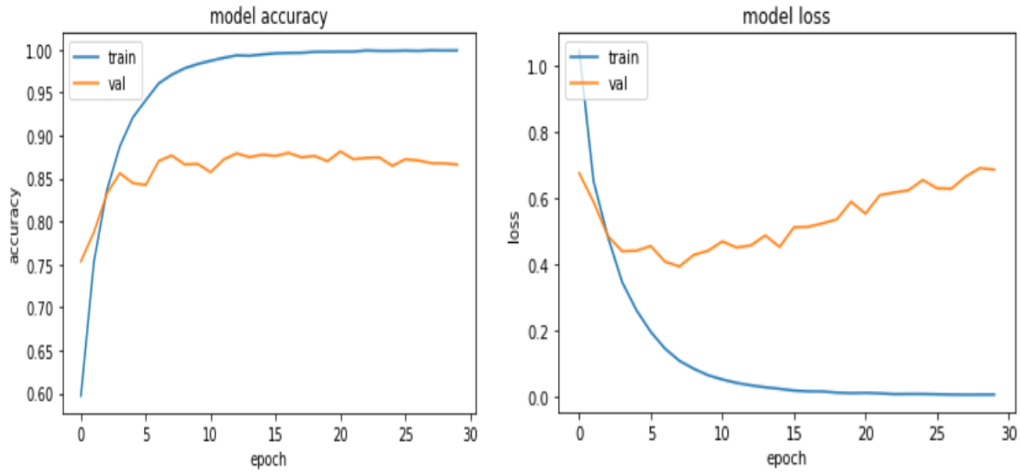


Figure 26. Confusion matrix of test samples for Shallow Type CNN

The accuracy and loss curve showed in Figure 27 large gap between training and validation curves, which indicates that overfitting and the model cannot learn intricate patterns because of a single convolution layer. The overfitting can be prevented by adding batch normalization (BN) for regularization. The model attains an average classification accuracy of 86%. The model performed well on training data but less able to generalize new data.



a) b)

Figure 27. a) Accuracy curves for Shallow Color CNN b) loss curves for Shallow Color CNN

6.2.3 Deep Color CNN Architecture

The Deep CNN architecture is designed to overcome overfitting and reduce misclassification of types and color samples. The detailed design is discussed in the previous chapter 5.4. The network is trained for 8,480 samples and validates on 3,635 samples with an image resolution of 100x100, 30 epochs using mini-batch sizes of 64. The model is compiled using the sparse categorical cross-entropy as the loss function and SGD as the optimizer using a learning rate of 0.0001. Results are evaluated using

Precision, Recall, F1-score, Confusion Matrix, model accuracy, and loss curves. Table 11 shows the Precision, Recall, and F1-score values of each class. The average accuracy obtained for the Deep Color CNN is 95%.

The yellow, red and cyan gives higher values when compared to other classes. Data samples of colors yellow, red and cyan looks bright whereas the gray and black colors are too light due to the sun reflection on metallic part of the car. The 3,635 samples are tested, and misclassification is significantly reduced in the deeper architecture for all colors, but the 93 samples of white color, as shown in Figure 28, are misclassified as a gray color. The gray color in the deeper architecture shows the low precision with high recall value and vice versa for white color. This clearly shows the misclassification between white and gray color due to various scenarios such as a change in weather condition and illumination, which significantly affects the overall accuracy of the classification model.

Table 11. Classification report for Deep Color CNN

Color Classes	precision	recall	F1-score
black	0.93	0.97	0.95
blue	0.97	0.95	0.96
cyan	1.00	0.98	0.99
gray	0.77	0.92	0.84
green	0.96	0.95	0.95
white	0.98	0.79	0.87
yellow	1.00	0.99	0.99
red	0.98	0.99	0.99

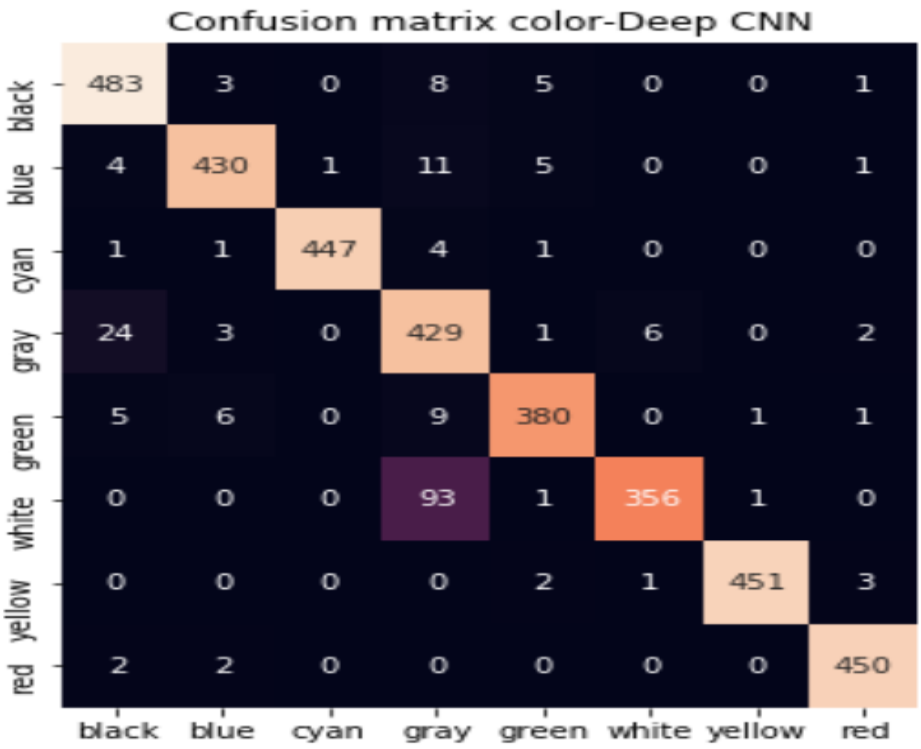
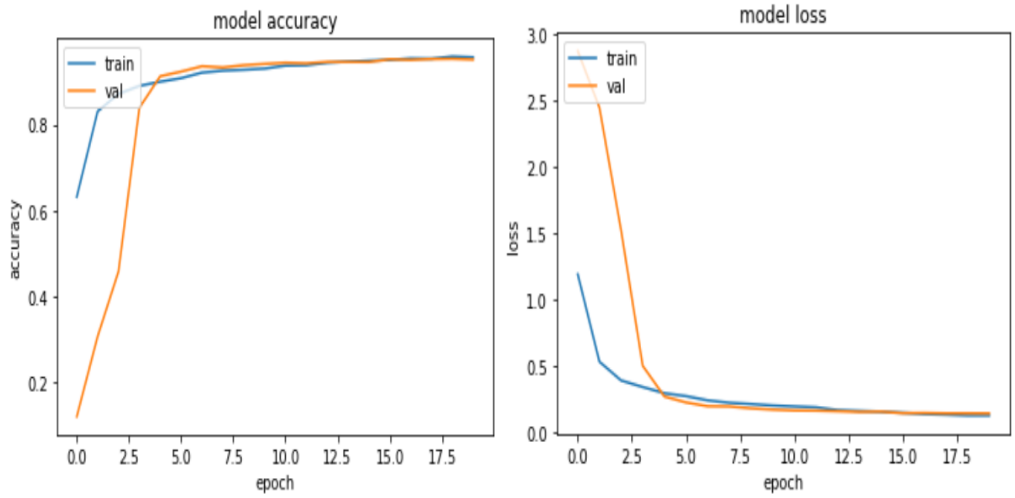


Figure 28. Confusion matrix of test samples for Deep Color CNN

First two sets of convolution layer to learn local features like edges and corners in the images with 32 filters followed by nonlinear activation layer ReLU and batch normalization (BN) layer helps in effectively reducing the number of epochs to train the network. Then, the added Pooling layer with a size of 2x2 reduces the spatial dimensions of input, followed by a dropout of 25% is applied to prevent overfitting. The same pattern is repeated for another set of convolution layers to learn more complex patterns in the data with 64 filters; then the network is flattened to a single-dimensional vector with 512 neurons followed by a fully-connected layer, activation layer ReLU, BN, and dropout of 50%. This pattern helps the model to be good fit. Figure 29a) tells the model achieves accuracy of 95% and not overfitted. The training loss curves decreases to a point of stability and the gap between training and validation loss values are greatly reduced. The

training and validation loss converges at the fifth epoch and decreases to the point of stability with a minimal gap between two final loss values as shown in Figure 29b.



a) b)

Figure 29. a) Accuracy curves for Deep Color CNN b) loss curves for Deep Color CNN

6.2.4 Deep Type CNN Architecture

The Deep CNN Architecture is designed to overcome overfitting and reduce the misclassification of types and color samples. The detailed design is discussed in the previous chapter 5.4. The network is trained for 12,346 samples and validates on 5,292 samples with an image resolution of 100x100, 30 epochs using mini-batch sizes of 64. Results are evaluated using precision, recall, F1-score, confusion matrix, and learning curves. Table 12 shows the precision, recall, F1-score value of each class. The average accuracy obtained for Deep Type CNN is 89%.

The motorcycle data samples are challenging such as variation in the background, illumination and overexposure. The misclassification is greatly reduced in the deeper architecture for all Types, but the 111 samples of the bus type, as shown in Figure 30, are

misclassified as a van type, and 152 samples of truck type are misclassified as a van type due to their similar features.

Table 12. Classification report for Deep Type CNN

Type Classes	Precision	Recall	F1-score
Bus	0.96	0.71	0.82
Truck	0.89	0.70	0.78
Motorcycle	0.82	0.89	0.85
Van	0.73	0.97	0.93
SUV	0.99	1.00	0.97
Sedan	0.94	1.00	0.97

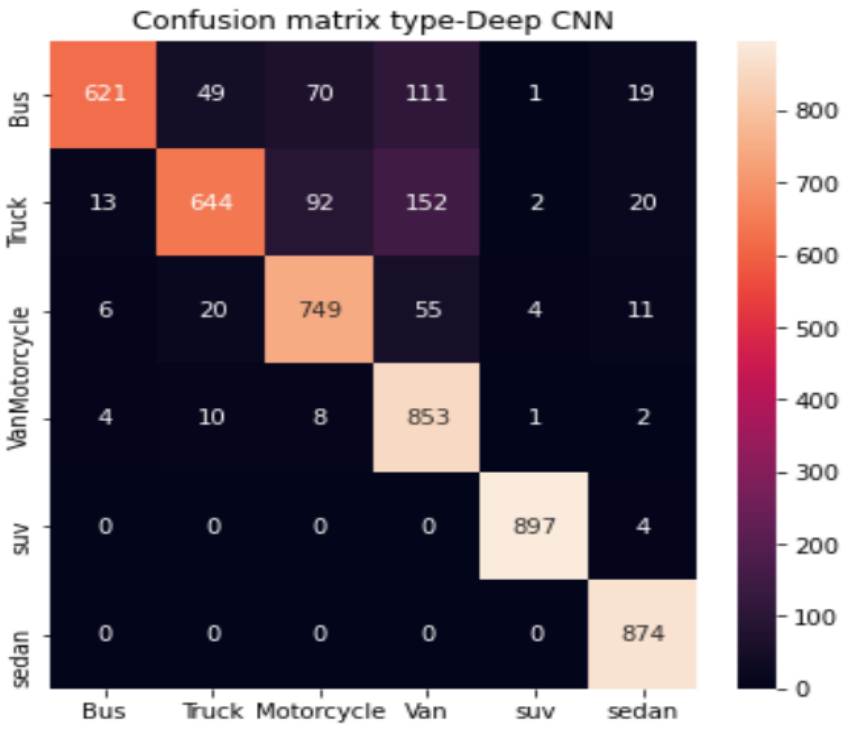
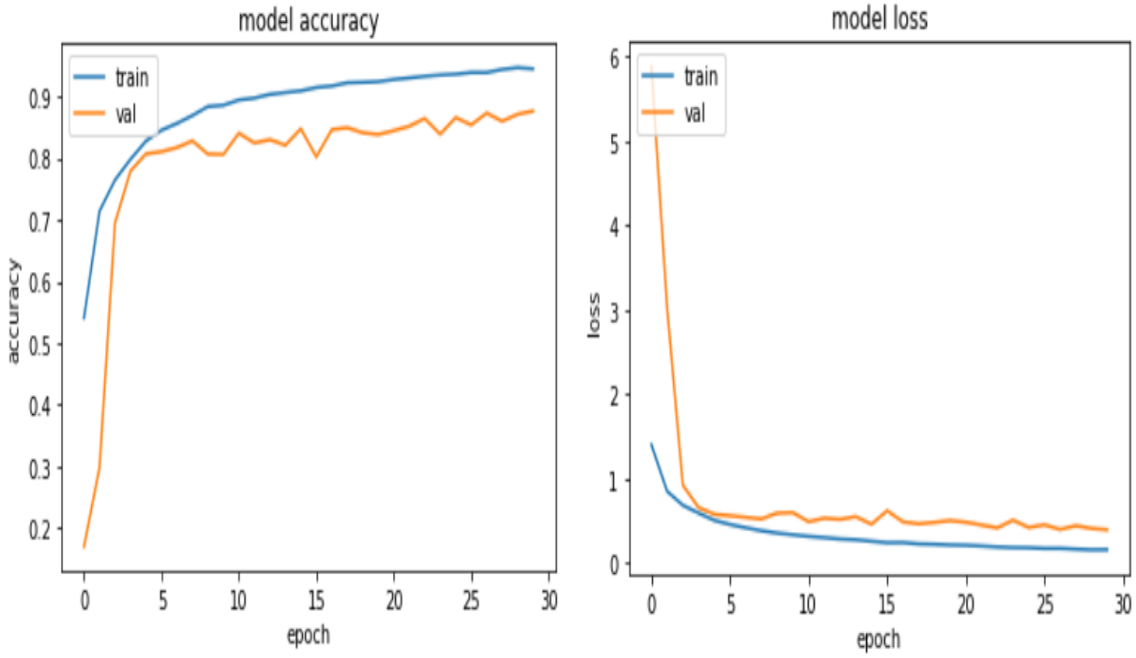


Figure 30. Confusion matrix of test samples for Deep Type CNN

Figure 31. a) tells the model achieves accuracy of 89% and not overfitted. The training loss curves decreases to a point of stability and the gap between training and validation loss values are greatly reduced. The training and validation loss converges at the third epoch and decreases to the point of stability with a minimal gap between two final loss values as shown in Figure 31.b).



a) b)

Figure 31. a) Accuracy curves for Deep Type CNN b) loss curves for Deep Type CNN

6.2.5 Comparison Between Shallow and Deep Architecture

In the Shallow CNN architecture, the model achieved an average accuracy for color classification is 94% and type classification is 86%. The model is overfitted as it is not able to learn more complex patterns. The accuracy is further increased in the Deep CNN. In the Deep CNN, the architecture achieved an average for color classification of 95% and vehicle type classification of 89%. Deep CNN achieves an increase in 1% for vehicle color and an increase of 3% for vehicle type classification accuracy when

compared to the Shallow CNN, as shown in Table 13. The individual Precision, Recall, and F1-score values are increased due to added convolutional layer to learn more complex patterns, batch normalization (BN) layer helps in effectively reducing the number of epochs to train the network and dropout layer to prevent overfitting. The misclassification of color classes includes white, gray, and black mainly due to overexposure and lighting. The misclassification of type classes includes van and truck as it resembles similar features and challenges in the datasets includes different viewpoints such as rear, front with environmental conditions and lighting. The higher accuracy Deep CNN model will help extract colors and types of vehicles from the child abduction involved, bringing child and elderly person safe home.

Table 13. Comparison between Shallow and Deep Architecture

Architecture	Vehicle Color	Vehicle Type
Shallow	94%	86%
Deep	95%	89%

6.2.6 Testing Results

The Deep Color CNN model is tested with sample images are shown in Figure 32, and the Deep type model is shown in Figure 33. The Deep color CNN misclassified black as gray is shown in Figure 34. makes mistakes in classifying due to lighting and white shades.

a) b)

Figure 32. a) Input images b) Output form Vehicle color Classifier model

a) b)

Figure 33. a) Input images b) Output form Vehicle type Classifier model

a) b)

Figure 34. a) Input black image b) misclassified Output form Vehicle color Classifier model.

6.2.7 Vehicle Detection

Pretrained YOLO v3 was utilized for testing the vehicle detection model. The input image shown in Figure 35 is passed into function *CV2.dnn.blobFromImage()*, which will convert the input image into a standard size of (416 x 416) that YOLO used to extract the features. Then, feed the YOLO Detector's blob image to extract the bounding box's coordinates and dimensions. The confidence shows the absence or presence of an object of any class. A low confidence value specifies that the network misinterprets the object. The minimum probability of filtering weak detections has a default value of 50% (0.5). Applying non-maxima suppression suppresses significantly overlapping bounding boxes, keeping only the most confident ones. NMS also ensures that they do not have any redundant or extraneous bounding boxes. Non-maxima suppression threshold with a default value of 0.3 was used. Truck and car detections with bounding boxes are shown

in Figure 36, which will check with possible vehicle type matches in the emergency signal to safely return child/older adult.

Figure 35. Input Truck image to YOLO detection model

Figure 36. Truck detection with bounding boxes using YOLOv3

6.2.8 Optical Character Recognition

The character recognition was tested with sample images as discussed in the previous section V. The detected license plate using the Python-tesseract tool [40] is one of the powerful tools used for character recognition. This will recognize and read the text embedded in images shown in Figure 37.

a) b)

Figure 37. a) Input image b) Text extraction

6.2.9 Execution in Response to Alert Signal

The images are tested with an alert signal “Red Sedan.” First, the vehicle color classification model will check for the possible matches of color in the given input images, and then the matched vehicle color will pass to the vehicle type model to find possible matches of vehicle type Sedan. The matched vehicle will be saved. The saved vehicle will pass to the YOLO Detection model to display images with bounding box and class probabilities are shown in Figure 38.

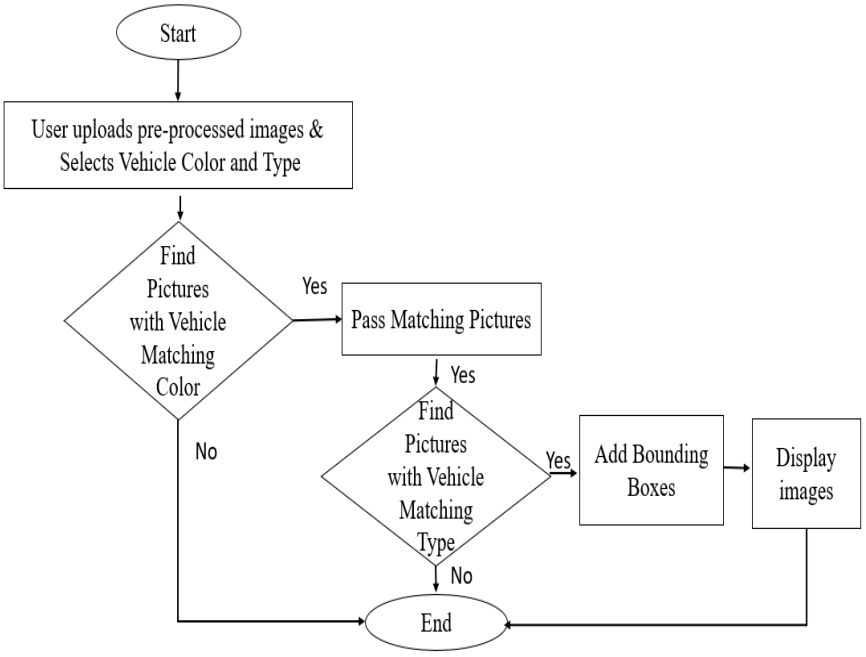


Figure 38. Workflow of the vehicle Classification and Detection model

The images are tested to find possible characters and numbers in the license plate to give vehicle identification is shown in Figure 39.

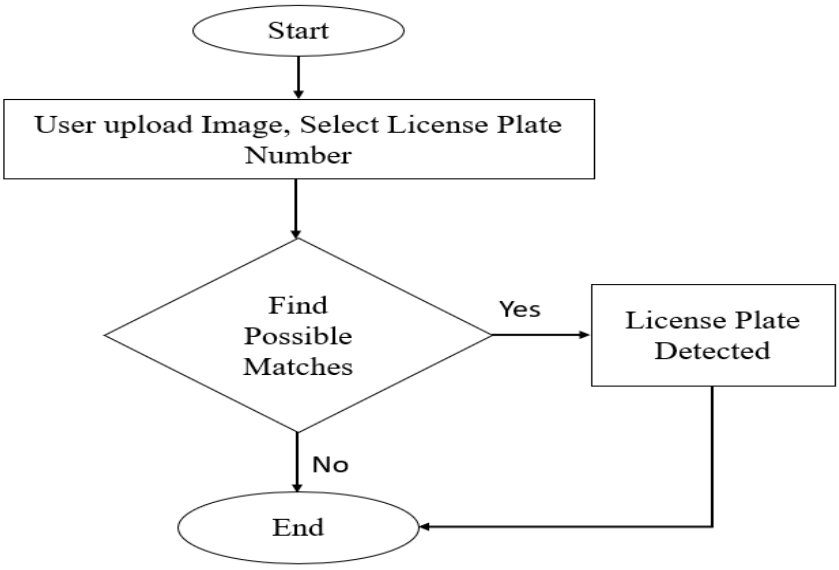
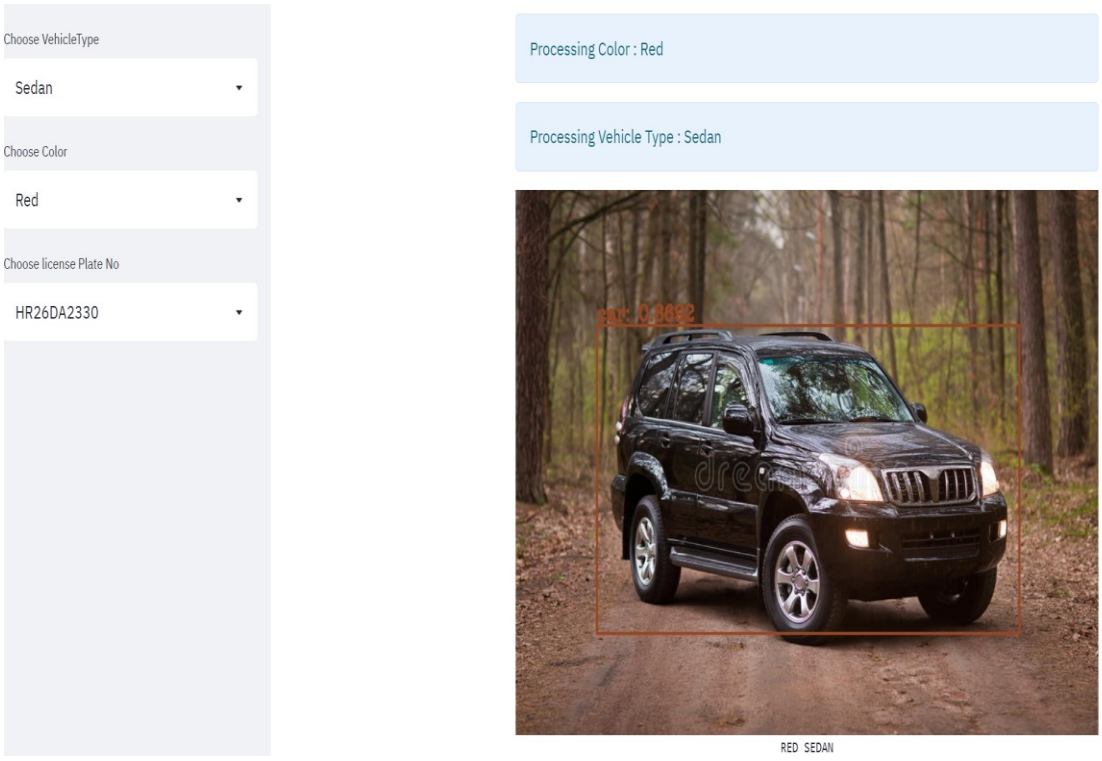


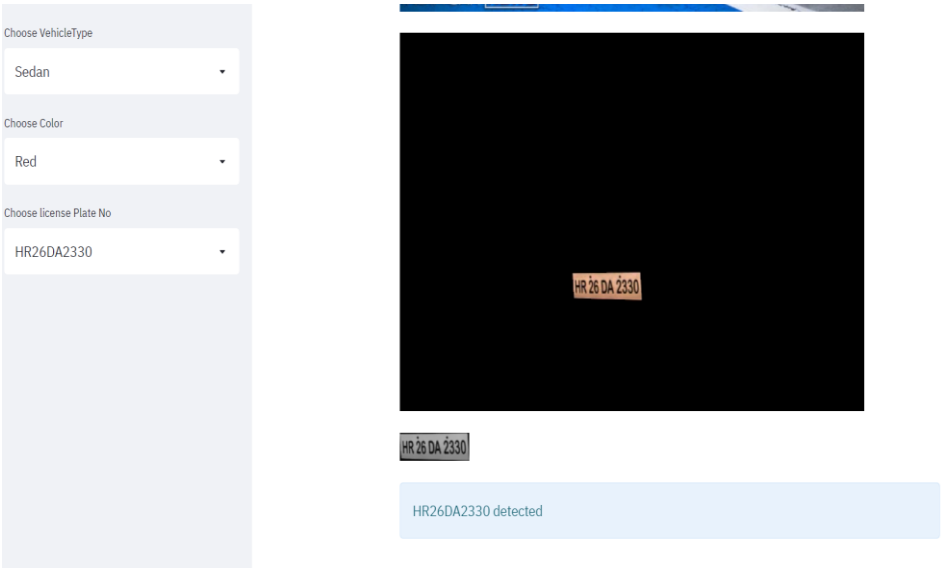
Figure 39. Workflow of OCR model in response to alert signal

The whole process will help find possible matches involved in these emergency alerts for child/older adult's safe return. The hosted app using Streamlit [42] is shown in Figure 40. First, the Figure 40 a) shows the misclassification of Black color as a Red due to several factors like reddish background and bright light. Second, the correctly classified Red Sedan is shown in Figure 40 b) and finally the detected License Plate number according to alert signal is shown in Figure 40 c).



a)

b)



c)

Figure 40. a) Misclassification of Black color as a Red, b) Classification and Detection and c) License plate Detection Hosted Models using Streamlit

7 CONCLUSION AND FUTURE WORK

This thesis research is focused on the application of Amber/ Silver alert emergencies. The details of broadcasting in Amber and Silver alerts are color, type of the vehicle, vehicle license plate numbers, and car brands. Vehicle types include six classes as Truck, Bus, van, SUV, Sedan, and Motorcycle. Vehicle colors include eight classes: green, blue, black, white, gray, yellow, cyan, and red. The system works once the Amber alerts are issued; the time taken to recovery is more than twelve hours for children in seventeen cases. More recovery time is due to the time is taken for someone to recognize the vehicle involved in child abduction and authorities to bring back the child or elderly safe back home.

This research work aims to design a Deep Learning model to classify vehicle colors and types faster and accurately. Datasets used to consider different environmental conditions like lighting and illumination with different viewpoints. First, Shallow CNN and Deep CNN, inspired by VGG16, are designed for vehicle colors and classification types. Accomplished an 89% classification accuracy for vehicle type and a 95% classification accuracy for colors under different weather conditions using the Deep Learning CNN. Deep CNN achieves an increase in 3% vehicle type classification accuracy when compared to the shallow CNN. Results shows by adding several layers like convolution layers, batch normalization layers help to increase the individual color and type metric values such as precision, recall, and F1-Score. The gap between training and validation accuracy and loss curves are reduced with minimal epochs, which clearly shows the model is not overfitted. This classification helps in faster detection.

Second, implementation of the pre-trained YOLO object detector to detect vehicles in images. Pretrained YOLO v3 was utilized for testing the vehicle detection model. detections with bounding boxes will help to check the possible vehicle type matches in the emergency signal to safely return child/older adult. Extraction of license plate characters and numbers using image processing techniques with OpenCV and Python will help identify possible License Plate matches. The whole process will help extract colors and types of vehicles from the child abduction involved, bringing child and elderly person safe home.

The future development of this work is to incorporating color, type classifier models and License plate Recognition models into a single data flow by using YOLO or Splitting dense layers to different path for both colors and types in real time applications. The execution of alert signal will be further analyzed with hardware configuration such as need of cameras, computing power, high performance cluster and hardware required for processing frames per seconds. Integrating single data flow and hardware configuration will help to detect multiple objects in a single image to process faster detection in real time to bring back child/ elderly person safe home.

REFERENCES

[1] Hsieh, Jun-Wei, Yu, Shih-Hao, Chen, Yung-Sheng, Hu and Wen-Fong, "An Automatic Traffic Surveillance System for Vehicle Tracking and Classification," *Intelligent Transportation Systems, IEEE Transactions,* 2006.

[2] "National Center for Missing & Exploited Children," *Wikipedia.* Accessed April 3, 2021. [Online]. Available: https://en.wikipedia.org/wiki/National_Center_for_Missing_%26_Exploited_Children.

[3] TAU Vehicle Type Recognition Competition" Kaggle. Accessed Dec. 10, 2020. [Online]. Available: https://www.kaggle.com/c/vehicle/overview/description

[4] Wu, Zhongyuan, Jun, Qian, Hong, Bin and Xiaofeng, "Multi-Scale Vehicle Detection for Foreground-Background Class Imbalance with Improved YOLOv2," *Sensors*, 2019.

[5] P. Chen, X. Bai and W. Liu, "Vehicle Color Recognition on Urban Road by Feature Context," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2340-2346, Oct. 2014.

[6] M. Zhu, B. He and Q. Wu, "Single Image Dehazing Based on Dark Channel Prior and Energy Minimization," in *IEEE Signal Processing Letters*, vol. 25, no. 2, pp. 174-178, Feb. 2018, doi: 10.1109/LSP.2017.2780886.

[7] L. Shi *et al*., "Image haze removal using dark channel prior and minimizing energy function," *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, Chengdu, 2017, pp. 256-259, doi: 10.1109/ITNEC.2017.8284983.

[8] J. Hu, Z. Li and X. Chen, "Modified Image Haze Removal Algorithm Based on Dark Channel Prior," *2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, Xiamen, China, 2019, pp. 1600-1605.

[9] "Keras's Image data preprocessing" *keras*. Accessed April. 03, 2021. [Online]. Available: https://keras.io/api/preprocessing/image/.

[10] "Silver Alert" *Wikipedia*. Accessed April. 03, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Silver_Alert.

[11] "Statistics | AMBER Alert," *Amberalert.* Accessed April. 03, 2021. [Online]. Available:https://amberalert.ojp.gov/statistics.

[12] Wikimedia.org. Accessed April. 03, 2021. [Online]. Available https://commons.wikimedia.org/wiki/File:Amber_Alert.jpg.

[13] Rachmadi, R. and Purnama.I,"Vehicle Color Recognition using Convolutional Neural Network," arXiv, 2015. [Online]. Available: https://arxiv.org/format/1510.07391.

[14] Zhang, M., Wang, P. and Zhang, X, "Vehicle Color Recognition Using Deep Convolutional Neural Networks", *International Conference on Artificial Intelligence and Computer Science* AICS, 2019.

[15] B. Su, J. Shao, J. Zhou, X. Zhang, L. Mei, "Vehicle Color Recognition in The Surveillance with Deep Convolutional Neural Networks", *2015 Joint International Mechanical Electronic and Information Technology Conference*, 2015.

[16] B. Su, J. Shao, J. Zhou, X. Zhang, L. Mei, “Vehicle Color Recognition in The Surveillance with Deep Convolutional Neural Networks”, *2015 Joint International Mechanical Electronic and Information Technology Conference*, 2015.

[17] W. Maungmai and C. Nuthong, “Vehicle Classification with Deep Learning,”*2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*, Singapore, 2019, pp. 294-298.

[18] Wu, Zhongyuan, Jun, Qian, Hong, Bin and Xiaofeng, “Multi-Scale Vehicle Detection for Foreground-Background Class Imbalance with Improved YOLOv2” *Sensors*. 19. 3336. 10.3390/s19153336.

[19] B. Hicham, A. Ahmed and M. Mohammed, “Vehicle Type Classification Using Convolutional Neural Network,” *2018 IEEE 5th International Congress on Information Science and Technology (CiSt)*, Marrakech, 2018, pp. 313-316.

[20] P.Meghana, S. SagarImambi, P. Sivateja and K. Sairam “Image Recognition for Automatic Number Plate Surveillance,” *International Journal of Innovative Technology and Exploring Engineering (IJITEE)* ISSN: 2278-3075, Volume-8 Issue-4, February 2019.

[21] D. Chowdhury, S. Mandal, D. Das, S. Banerjee, S. Shome and D. Choudhary, "An Adaptive Technique for Computer Vision Based Vehicles License Plate Detection System," *2019 International Conference on Opto-Electronics and Applied Optics (Optronix)*, Kolkata, India, 2019, pp. 1-6.

[22] Linjie Yang, Ping Luo, Chen Change Loy and Xiaoou Tang, “A Large-Scale Car Dataset for Fine-Grained Categorization and Verification**,”** *Computer Vision and Pattern Recognition (CVPR),* 2015.

[23] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified real-time object detection", *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 779-788, Jun. 2016.

[24] "COCO dataset," *cocodataset.* Accessed April. 03, 2021. [Online]. Available: https://cocodataset.org/#home.

[25] "PASCAL Visual Object Classes (VOC) Challenge," Accessed April. 03, 2021. [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/

[26] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement", 2018.

[27] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, 2017, pp. 6517-6525, doi: 10.1109/CVPR.2017.690.

[28] A. Rosebrock, Deep Learning for Computer Vision with Python, PyImageSearch, 2017.

[29] "Max-Pooling/Pooling," Accessed April. 03, 2021. [Online]. Available: https://computersciencewiki.org/index.php/Max-pooling_/_Pooling

[30] "CS23 In Convolutional Neural Networks for Visual Recognition," Accessed April. 03, 2021 [Online]. Available: http://cs231n.github.io/convolutional-networks/.

[31] "TAU Vehicle Type Recognition Competition" Kaggle. Accessed Dec. 10, 2020. [Online]. Available: https://www.kaggle.com/c/vehicle/overview/description.

[32] Wu, Zhon gyuan, Jun, Qian, Hong, Bin and Xiaofeng, "Multi-Scale Vehicle Detection for Foreground-Background Class Imbalance with Improved yolov2," *Sensors*, 2019.

[33] "Various Optimization Algorithms For Training Neural Network," Accessed April. 03, 2021 [Online]. Available: http://cs231n.github.io/convolutional-networks/.

[34] I. Goodfellow, Y. Bengio and A. Courville, "Deep learning" in Adaptive Computation And Machine Learning, Cambridge, MA, USA: MIT Press, pp. 775, 2016.

[35] Y. Miao, F. Liu, T. Hou, L. Liu and Y. Liu, "A Nighttime Vehicle Detection Method Based on YOLO v3," *2020 Chinese Automation Congress (CAC)*, Shanghai, China, 2020, pp. 6617-6621, doi: 10.1109/CAC51589.2020.9326819.

[36] "license-plate-recognition-using-OpenCV-python" Accessed April. 03, 2021 [Online]. Available: https://medium.com/programming-fever/license-plate-recognition-using-opencv-python-7611f85cdd6c

[37] C. R. Rashmi and C. P. Shantala, "Vehicle Density Analysis and Classification using YOLOv3 for Smart Cities," *2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, Coimbatore, India, 2020, pp. 980-986, doi: 10.1109/ICECA49313.2020.9297561.

[38] "Canny edge detector" Accessed April. 03, 2021 [Online]. Available: https://en.wikipedia.org/wiki/Canny_edge_detector

[39] "Draw Contours" Accessed April. 03, 2021 [Online]. Available: http://cs231n.github.io/convolutional-networks/.

[40] "pytesseract-PyPI" Accessed April. 03, 2021 [Online]. Available: https://pypi.org/project/pytesseract/.

[41] "Anaconda," *wikipedia*. Accessed April. 03, 2021 [Online]. Available: https://en.wikipedia.org/wiki/Anaconda.

[42] "Streamlit," Accessed April. 14, 2021 [Online]. Available: https://streamlit.io/.