A

Project Reporton

Real time traffic management system uging deep learning techniques

Submitted as partial fulfilment for the award of

BACHELOR OF TECHNOLOGY DEGREE

Session 2023-24

in

Computer Science & Engineering

Submitted By:

Km shivani-2000970100058

Name: Abhigyan(2000970100002)

Name: Aditya Singh(2000970100007)

Under the guidance

Dr. Ramveer Singh (Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS COLLEGE OF ENGINEERING AND TECHNOLOGY,
GREATER NOIDA

AFFILIATED TO

DR. A.P.J ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW, UP

MAY 2024

GALGOTIAS COLLEGE OF ENGINEERING & TECHNOLOGY

GREATER NOIDA, UTTAR PRADESH, INDIA- 201306.

CERTIFICATE

This is to certify that the project report entitled "Facial Emotion Detection and Recc
using Convolutional Neural Network" submitted by **km shivani(2000970100058),
Abhigyan(2000970100002), Aditya Singh(2000970100007)** to the Galgotia's College of Er
& Technology, Greater Noida, Utter Pradesh, affiliated to Dr. A.P.J. Abdul Kalam Technical
Lucknow, Uttar Pradesh in partial fulfillment for the award of Degree of Bachelor of Technolc
Computer science & Engineering is a bonafide record of the project work carried out by the
my supervision during the year 2023-2024.

Dr. Ramveer singh	**Dr. Pushpa Choudhary**
Professor	**Professor and Head**
Dept. of CSE	**Dept. of CSE**

GALGOTIAS COLLEGE OF ENGINEERING & TECHNOLOGY

GREATER NOIDA, UTTER PRADESH, INDIA- 201306.

ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend my sincere thanks to all of them.

We are highly indebted to **Dr. Ramveer Singh** for his guidance and constant supervision. Also, we are highly thankful to them for providing necessary information regarding the project & also for their support in completing the project.

We are extremely indebted to **Dr. Pushpa Choudhary**, HOD, Department of Computer Science and Engineering, GCET and Mr. Manish Kumar Sharma, Project Coordinator, Department of Computer Science and Engineering, GCET for their valuable suggestions and constant support throughout my project tenure. We would also like to express our sincere thanks to all faculty and staff members of Department of Computer Science and Engineering, GCET for their support in completing this project on time.

We also express gratitude towards our parents for their kind co-operation and encouragement which helped me in completion of this project. Our thanks and appreciations also go to our friends in developing the project and all the people who have willingly helped me out with their abilities.

Km shivani-2000970100058
Abhigyan(2000970100002)
Aditya Singh(2000970100007)

ABSTRACT

Manual traffic surveillance can be a daunting task as Traffic Management Centers operate a myriad of cameras installed over a network. Injecting some level of automation could help lighten the workload of human operators performing manual surveillance and facilitate making proactive decisions which would reduce the impact of incidents and recurring congestion on roadways. This article presents a novel approach to automatically monitor real time traffic footage using deep convolutional neural networks and a stand-alone graphical user interface. The authors describe the results of research received in the process of developing models that serve as an integrated framework for an artificial intelligence enabled traffic monitoring system. The proposed system deploys several state-of-the-art deep learning algorithms to automate different traffic monitoring needs.Taking advantage of a large database of annotated video surveillance data, deep learning- based models are trained to detect queues, track stationary vehicles, and tabulate vehicle counts. A pixel-level segmentation approach is applied to detect traffic queues and predict severity. Real-time object detection algorithms coupled with different tracking systems are deployed to automatically detect stranded vehicles as well as perform vehicular counts. At each stages of development.interesting experimental results are presented to demonstrate the effectiveness of the proposed system. Overall, the results demonstrate that the proposed framework performs satisfactorily under varied conditions without being immensely impacted by environmental hazards such as blurry camera views, low illumination, rain, or snow.

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Monitoring traffic effectively has long been one of the most important efforts in transportation engineering. Till date, most traffic monitoring centers rely on human operators to track the nature of traffic flows and oversee any incident happening on the roads. The processes involved in manual traffic condition monitoring can be challenging and time-consuming. As humans are prone to inaccuracies and subject to fatigue, the results often involve certain discrepancies. It is therefore, in best interests to develop automated traffic monitoring tools to diminishing the work load of human operators and increase the efficiency of output. Hence, it is not surprising that automatic traffic monitoring systems have been one of the most important research endeavors in intelligent transportation systems. It is worthwhile to note that most present-day traffic monitoring activity happens at the Traffic Management Centers (TMCs) through vision-based camera systems. However, most existing vision-based systems are monitored by humans which makes it difficult to accurately keep track of congestion, detect stationary vehicles whilst concurrently keeping accurate track of the vehicle count. Therefore, TMCs have been laying efforts on bringing in some levels of automation in traffic management. Automated traffic surveillance systems using Artificial Intelligence (AI) have the capability to not only manage traffic well but also monitor and access current situations that can reduce the number of road accidents. Similarly, an AI enabled system can identify each vehicle and additionally track its movement pattern characteristic to identify any dangerous driving behavior, such as erratic lane changing behavior. Another important aspect of an AI-enabled traffic monitoring system is to correctly detect any stationary vehicles on the road. Often-times, there are stationary vehicles which are left behind that impedes the flow of preceding vehicles and causes vehicles to stack-up. This results in congestion that hampers free mobility of vehicles. Intelligent traffic monitoring systems are thus, an integral component of systems needed to quickly detect and alleviate the effects of traffic congestion and human factors.

In the last few years, there has been extensive research on machine and deep learning-based traffic monitoring systems. Certain activities such as vehicle count, and traffic density estimation are limited by the process of engaging human operators and requires some artificial intelligence intervention. Traffic count studies for example require human operators to be out in the field during specific hours, or in the case of using video data, human operators are required to watch man hours of pre-recorded footage to get an accurate estimation of volume counts. This can be both cumbersome and time consuming. Similarly, when it comes to seeing traffic videos from multiple CCTV cameras, it becomes extremely difficult to analyze each traffic situation in real-time. Therefore, most TMCs seek out on deploying automated systems that can in fact, alleviate the workload of human operators and lead to effective traffic management system. At the same time, the costs associated are comparatively lower due to savings associated with not needing to store multiple hours of large video data. In this study, we deployed several state-of-the-art deep learning algorithms based on the nature of certain required traffic operations. Traditional algorithms [1-3] often record lower accuracies and fails at capturing complex patterns in a traffic scene, hence we tested and deployed deep learningbased models trained on thousands of annotated traffic images. Thus, the proposed system as shown in Figure 1 can perform the following:

1. Monitor Traffic Congestion
2. Traffic Accidents, Stationary or Stranded Vehicle Detection
3. Vehicle Detection and Count
4. Manage Traffic using a stand-alone Graphical User Interface (GUI)
5. Scale traffic monitoring to Multiple Traffic Cameras

OBJECTIVE

The proposed system deploys several state-of-the-art deep learning algorithms to automate different traffic monitoring needs. Taking advantage of a large database of annotated video surveillance data, deep learning based models are trained to detect queues, track stationary vehicles, and tabulate vehicle counts.

CHAPTER 2

LITERATURE SURVEY

2.1 Inferences from Literature Survey :

Title: An alternative technique for the computation of the designator in the retinex theory of color vision.
Author: E **H** Land
Year: 1986
Description:

Accepting the first postulate of the retinex theory of color vision that there are three independent lightness-determining mechanisms (one for long waves, one for middle waves, and one for short waves), each operative with less than a millisecond exposure and each served by its own retinal pigment, a basic task of retinex theory becomes the determination of the nature of these mechanisms. Earlier references proposed several workable algorithms. [Land, E. H. (1959) Proc. Natl. Acad. Sci. USA 45, 115-129; Land, E. H. (1959) Proc. Natl. Acad. Sci. USA 45, 636-644; Land, E. H. (1983) Proc. Natl. Acad. Sci. USA 80, 5163-5169; Land, E. H. & Mccann, J. J. (1971) J. Opt. Soc. Am. 61, 1-11; Land, E. H. (1986) Vision Res. 26, 7-21.] The present paper describes a relatively simple alternative technique for the computation of the designator in retinex theory and reports the general operational effectiveness of the new technique, including the competence, not possessed by earlier algorithms, for generating Mach bands.

Title: Single image haze removal using dark channel prior

In this paper, we propose a simple but effective image prior - dark channel prior to remove haze from a single input image. The dark channel prior is a kind of statistics of the haze-free outdoor images. It is based on a key observation - most local patches in haze-free outdoor images contain some pixels which have very low intensities in at least one color channel. Using this prior with the haze imaging model, we can directly estimate the thickness of the haze and recover a high quality haze-free image. Results on a variety of outdoor haze images demonstrate the power of the proposed prior. Moreover, a high quality depth map can also be obtained as a by-product of haze removal.

Title: A Deep Convolutional Network for Traffic Congestion Classification

We describe the development of a deep neural network for the recognition of traffic congestion state in surveillance camera imagery. A volume of imagery has been collected from Transport for London street cameras at a number of locations. The imagery covers a range of road configurations, times of the day, atmospheric and lighting conditions. A subset of the imagery has been ground-truthed by labelling the images with a perceived traffic congestion state. This labelled dataset is used to characterise a deep neural network for the target congestion state recognition problem. Here a two-phase network is trained using GoogLeNet for the image processing component and a bespoke deep subnet for the congestion recognition step. The subnet is trained on soft class labels, these taking labeller confidence into account. The combined classifier obtains 95% accuracy on held-out test samples, with many of the misclassified images turning out to be borderline cases. Other results include regression-style outputs which rank a collection of images in order of increasing congestion.

Title: Traffic Congestion Detection from Camera Images using Deep Convolution Neural Networks

Recent improvements in machine vision algorithms have led to closed-circuit television (CCTV) cameras emerging as an important data source for determining of the state of traffic congestion. In this study we used two different deep learning techniques, you only look once (YOLO) and deep convolution neural network (DCNN), to detect traffic congestion from camera images. The support vector machine (SVM), a shallow algorithm, was also used as a comparison to determine the improvements obtained using deep learning algorithms. Occupancy data from nearby radar sensors were used to label congested images in the dataset and for training the models. YOLO and DCCN achieved 91.5% and 90.2% accuracy, respectively, whereas SVM's accuracy was 85.2%. Receiver operating characteristic curves were used to determine the sensitivity of the models with regard to different camera configurations, light conditions, and so forth. Although poor camera conditions at night affected the accuracy of the models, the areas under the curve from the deep models were found to be greater than 0.9 for all conditions. This shows that the models can perform well in challenging conditions as well.

Title: Scalable Deep Traffic Flow Neural Networks for Urban Traffic Congestion Prediction

Tracking congestion throughout the network road is a critical component of intelligent transportation network management systems. Understanding how the traffic flows and short-term prediction of congestion occurrence due to rush-hour or incidents can be beneficial to such systems to effectively manage and direct the traffic to the most appropriate detours. Many of the current traffic flow prediction systems are designed by utilizing a central processing component where the prediction is carried out through aggregation of the information gathered from all measuring stations. However, centralized systems are not scalable and fail provide real-time feedback to the system whereas in a decentralized scheme, each node is responsible to predict its own short-term congestion based on the local current measurements in neighboring nodes. We propose a decentralized deep learning-based method where each node accurately predicts its own congestion state in real-time based on the congestion state of the neighboring stations. Moreover, historical data from the deployment site is not required, which makes the proposed method more suitable for newly installed stations. In order to achieve higher performance, we introduce a regularized Euclidean loss function that favors high congestion samples over low congestion samples to avoid the impact of the unbalanced training dataset. A novel dataset for this purpose is designed based on the traffic data obtained from traffic control stations in northern California. Extensive experiments conducted on the designed benchmark reflect a successful congestion prediction.

CHAPTER 3
REQUIREMENT ANALYSIS

3.1 EXISTING SYSTEM:

- Automatic vehicle counting may be slightly less than the actual number of vehicles sometimes due to congestion and heavy traffic flow.
- Current machine-learning-based image processing techniques, a possibility of detecting multiple vehicles as a single object arises when they travel through similar paths and speeds, even though on different road lanes. Hence, it is still difficult to obtain an accurate trajectory of a vehicle by tracking the exact position of the road lane where the vehicle is located
- Sudden illumination changes on the establishment of background, such as the cloud occlusion and dissipation on cloudy days cant predicited the exact accuracy

3.2 PROPOSED SYSTEM:

- YOLO has high applicability to real-time traffic monitoring based on its capability to process multiple images faster than conventional region-based convolutional neural networks (R-CNNs).
- With the aid of deep-learning techniques such as YOLO or faster R-CNN, the performance of detecting vehicles using real-time traffic vision data has been tried to improve in several studies. Their common purpose was to accurately count vehicles for estimating traffic conditions in specific road spots
- Obtaining accurate trajectories of multiple vehicles would be advantageous to traffic managers intending to improve the accuracy of collecting travel volume or queue length values in each traveling direction at an intersection

- we attempt to reduce the error of estimating the center points of the bounding boxes in the images of vehicles to ensure proper performance of the HD map- matching process.

3.3 SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS:

• **System**	Pentium IV 2.4 GHz.
• **Hard Disk**	40GB.
• **Floppy Drive**	1.44 Mb.
• **Monitor**	14' Colour Monitor.
• **Mouse**	Optical Mouse.
• **Ram**	512 Mb.

SOFTWARE REQUIREMENTS:

• **Operating system**	Windows 7 Ultimate.
• **Coding Language**	Python.
• **Front-End**	Python.
• **Designing**	Html,css,javascript.
• **Data Base**	MySQL.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

4.1 SELECTED METHODOLOGY OR PROCESS MODEL

The methodology adopted for implementing an automatic traffic monitoring system is shown in Figure 2. The main components consist of first, a GPU-enabled backend (on premise) which is designed to ensure that very deep models can be trained quickly and implemented on a wide array of cameras in near real time. At the heart of the proposed AI-enabled traffic monitoring system is the development and training of several deep convolutional neural network models that are capable of detecting and classifying different objects or segmenting a traffic scene into its constituent objects. Manually annotated traffic images served as the main source of dataset used for training these models. To enable the system to be situationally aware, different object tracking algorithms are implemented to generate trajectories for each detected object on the traffic scene at all times. The preceding steps are then combined to extract different traffic flow variables (e.g. Traffic volume and occupancy) and monitor different traffic conditions such as queueing, crashes and other traffic scene anomalies. The AI-enabled traffic monitoring system is capable of tracking different classes of vehicles, tabulating their count, spotting and detecting congestion and tracking stationary vehicles in real-time.

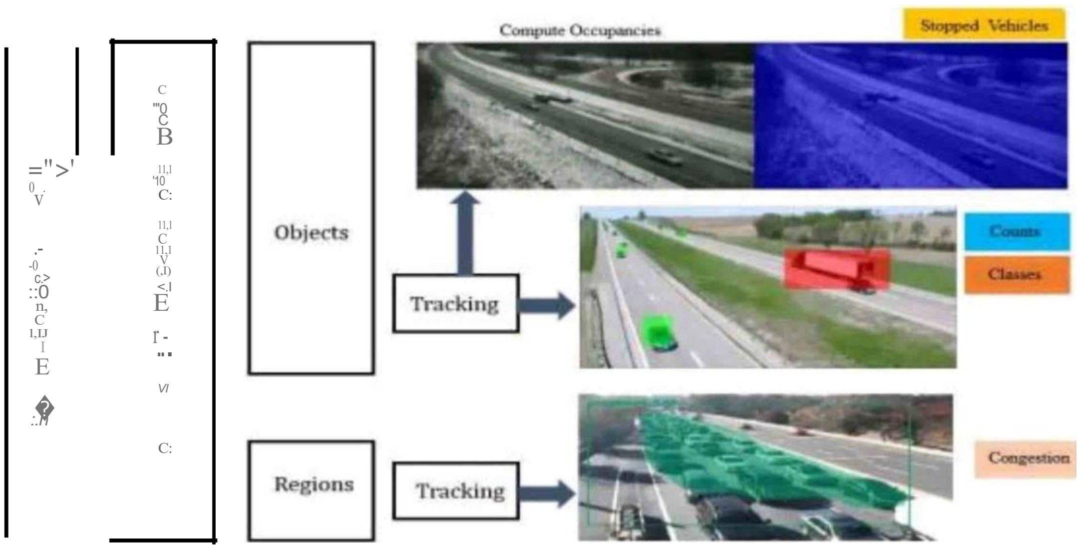


Fig 4.1 Visual Representation of the Proposed AI-Enabled System

4.2 ARCHITECTURE/ OVERALL DESIGN OF PROPOSED SYSTEM

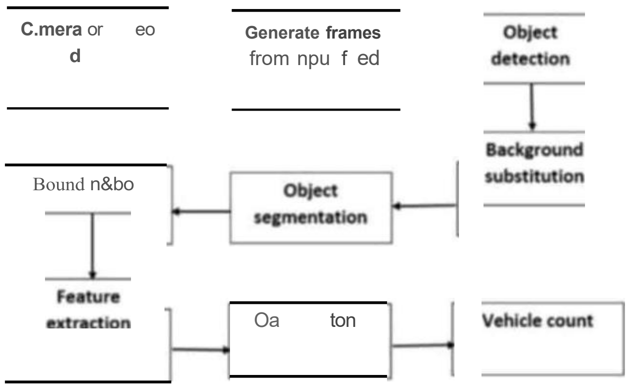


Fig 4.2 System architecture

This section describes the main structure of the vehicle detection and counting system. First, the video data of the traffic scene are entered. Then, the road surface area is extracted and divided. The YOLOv3 deep learning object detection method is used to detect the vehicle object in the highway traffic scene. Finally, ORB feature extraction is performed on the detected vehicle box to complete multi-object tracking and obtain vehicle traffic information.

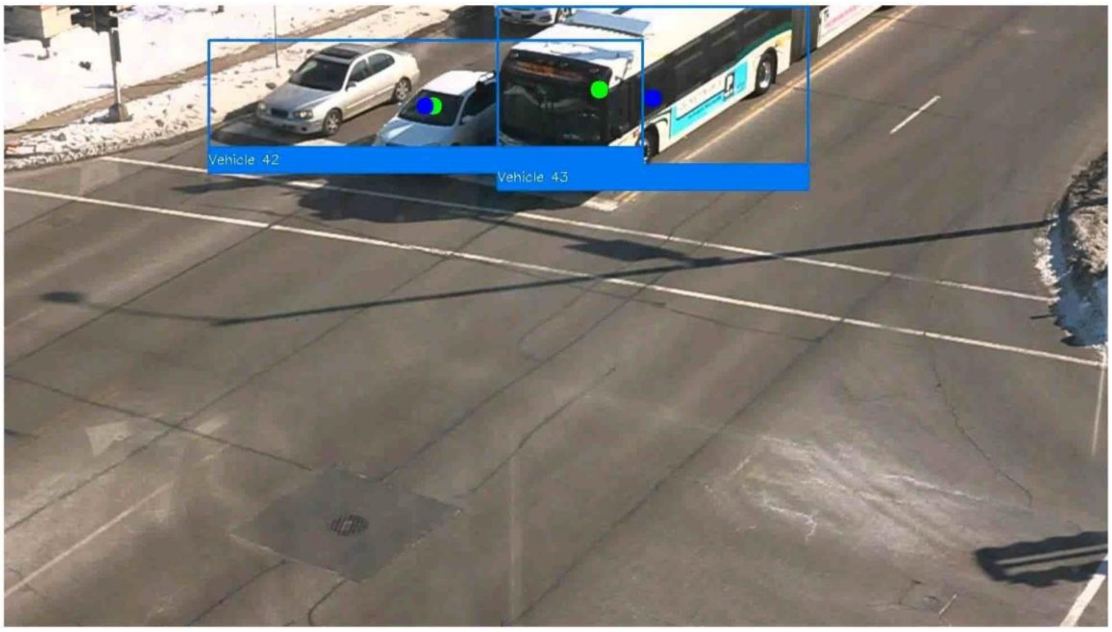


Fig 4.2.1 **Counting of vehicles**

According to the figure, the road surface segmentation method is used to extract the road area of the highway. The road area is divided into two parts based on the position where the camera is erected, a remote area and a proximal area. Then, the vehicles in the two road areas are detected using the YOLOv3 object detection algorithm. This algorithm can improve the small object detection effect and solve the problem that the object is difficult to detect due to the sharp change of the object scale. The ORB algorithm is then used for multi-object tracking. The ORB algorithm extracts the detected box's features and matches them to achieve correlation between the same object and different video frames. Finally, traffic statistics are calculated. The trajectory generated by the object tracking is generated, the vehicle driving direction is determined, and traffic information such as the number of vehicles in each category is collected. This system improves the accuracy of object detection from the highway surveillance video perspective and constructs a detection tracking and traffic information acquisition plan within the full field of the camera view.

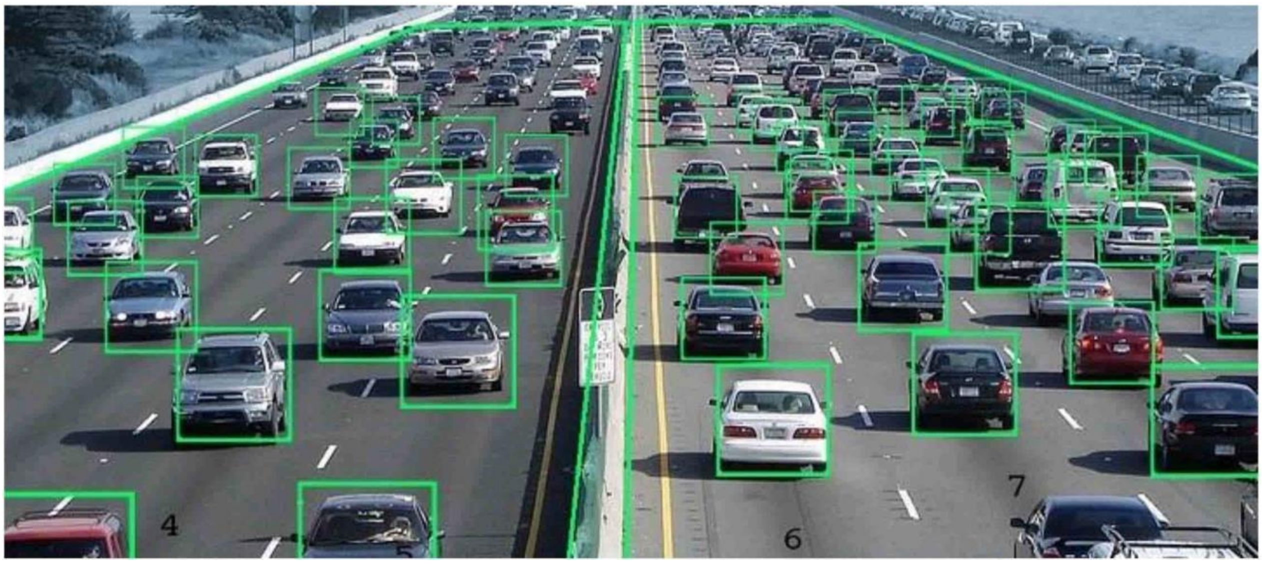


Fig 4.2.2 **Detection of vehicles**

4.3 Technologies:

Faster R-CNN

Faster R-CNN is a two-stage target detection algorithm [13]. In Faster-RCNN, a Region Proposal Network (RPN) shares complete-image convolutional features along with a detection network that enables cost free region proposals. Here, the RPN simultaneously predicts object bounds and their equivalent score values at each position. End to end training of RPN provides high class region proposals which is used by Faster R-CNN to achieve object predictions. Compared to Fast R-CNN, Faster R-CNN produces high-quality object detection by substituting selective search method with RPN. The algorithm splits every image into multiple sections of compact areas and then passes every area over an arrangement of convolutional filters to extract high-quality feature descriptors which is then passed through a classifier. After that the classifier produces the probability of objects in each section of an image. To achieve higher prediction accuracies on traffic camera feeds, the model is trained for 5 classes viz. pedestrian, cyclist, bus, truck and car. Training took approximately 8 hours on NVIDIA GTX 1080Ti GPU. The model processed video-feeds at 5 frames per second.

Mask R-CNN

Mask R-CNN abbreviated as Mask-region based Convolutional Neural Network is an extension to Faster R-CNN [11]. In addition to accomplishing tasks equivalent to Faster R-CNN, Mask R-CNN supplements it by adding superior masks and sections the region of interest pixel-by-pixel. The model used in this study is based on Feature Pyramid Network (FPN) and is executed with resnet101 backbone. In this, ResNet101 served as the feature extractor for the model. While using FPN, there was an improvement in the standard feature extraction pyramid by the introduction of another pyramid that took higher level features from the first pyramid and consequently passed them over to subordinate layers. This enabled features at each level to obtain admission at both higher and lower-level characters. In this study, the minimum detection confidence rate was set at 90% and run at 50 validation steps. An image-centric training approach was followed in which every image was cut to the square's shape.

The images were converted from 1024x1024pxx3 (RGB) to a feature map of shape 32x32x2048 on passing through the backbone network. Each of our batch had a single image per GPU and every image had altogether 200 trained Region of Interests (ROIs). Using a learning rate of 0.001 and a batch size of 1, the model was trained on NVIDIA GTX 1080Ti GPU. A constant learning rate was used during the iteration. Likewise, a weight decay of 0.0001 and a learning momentum of 0.9 was used.

The total training time for the model training on a sample dataset was approximately 3 hours. The framework for Mask-RCNN is shown in Figure 3.

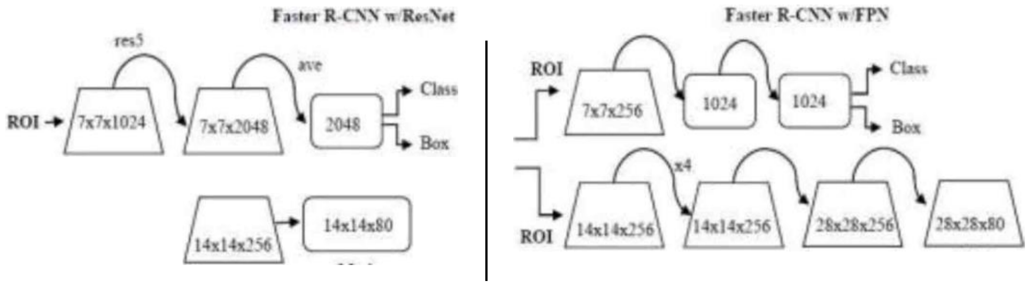


Fig 4.3. Mask R-CNN Framework

YOLO

You Only Look Once (YOLO) is the state-of-the-art object detection algorithm [12]. Unlike traditional object detection systems, YOLO investigates the image only once and detects if there are any objects in it. In this study, YOLOv4 was used to perform vehicle detection, counts, and compare results for traffic queues generation. Most contemporary object detection algorithms repurpose CNN classifiers with an aim of performing detections. For instance, to perform object detection, these algorithms use a classifier for that object and test it at varied locations and scales in the test image. However, YOLO reframes object detection i.e., instead of looking at a single image thousand times to perform detection, it just looks at the image once and performs accurate object predictions. A singe CNN concurrently predicts multiple bounding boxes and class probabilities for those generated boxes. To build YOLO models, the typical time was roughly 20-30 hours. YOLO used the same hardware resources for training as Mask R-CNN.

CenterNet

CenterNet [14] discovers visual patterns within each section of a cropped image at lower computational costs. Instead of detecting objects as a pair of key points, CenterNet detects them as a triplet thereby, increasing both precision and recall values. The framework builds up on the drawbacks encountered by CornerNet [35] which uses a pair of corner-keypoints to perform object detection. However, CornerNet fails at constructing a more global outlook of an object, which CenterNet does by having an additional keypoint to obtain a more central information of an image. CenterNet functions on the intuition that if a detected bounding box has a higher IoU with the ground-truth box, then the likelihoods of that central keypoint to be in its central region and be labelled in the same class is high. Hence, the knowledge of having a triplet instead of a pair increases CenterNet's superiority over CornerNet or any other anchor-based detection approaches. Despite using a triplet, CenterNet is still a single-stage detector but partly receives the functionalities of Roi pooling. Figure 4 shows the architecture of CenterNet where it uses a CNN backbone that performs cascade corner pooling and center pooling to yield two corner and a center keypoint heatmap. Here, cascade corner pooling enables the original corner pooling module to receive internal information whereas center pooling helps center keypoints to attain further identifiable visual pattern within objects that would enable it to perceive the central part of the region. Likewise, analogous to CornerNet, a pair of detected corners and familiar embeddings are used to predict a bounding box. Then after, the final bounding boxes are determined using the detected center keypoints.

The following sections give out a brief description of several traffic operations that could be seamlessly automated.

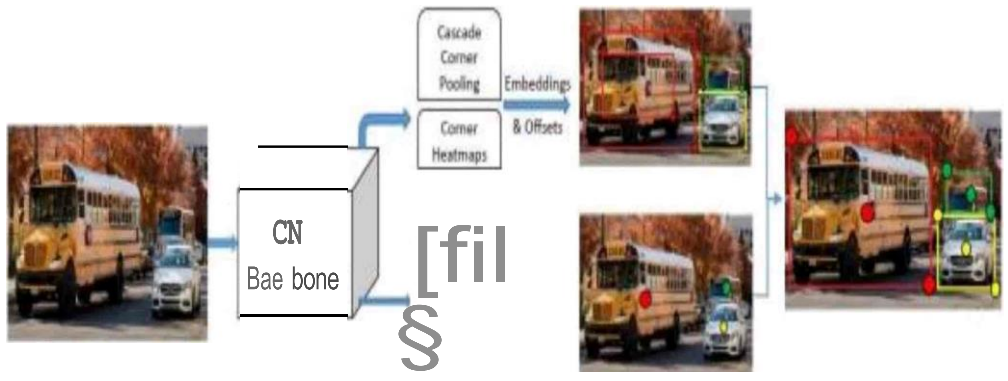


Figure 4.4 Architecture of CenterNet

Monitoring Traffic Queues

The methodology adopted for an automatic queue monitoring system is shown in Figure 5. The first step of performing annotation was achieved using a VGG Image Annotator [36]. In the follow up, annotated images were used to train both Mask R-CNN and YOLO models. The training times for Mask R-CNN and YOLO were approximately 3.5 and 22 hours respectively. After training was done, these models were run on real-time traffic videos to evaluate their performance. The main reason for using Mask R-CNN was due to its ability to obtain pixel-level segmentation masks that made queue detections precise. Since, YOLO uses a bounding box to perform detection, it covers areas that are both congested and non-congested. Therefore, Mask-RCNN has an advantage over YOLO when it comes to precisely predicting classified regions of interest.

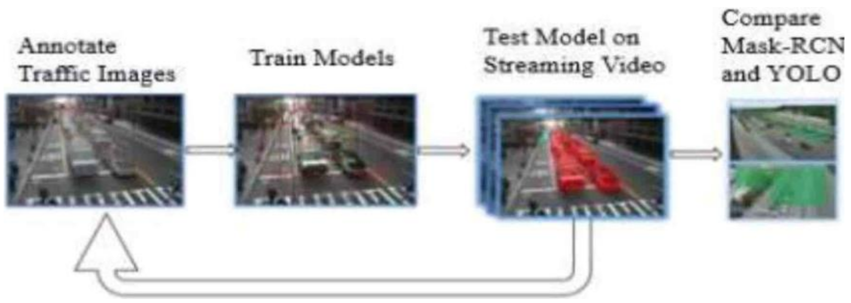


Figure 4.5 Flowchart of step-wise operations

Stationary Vehicle Detection

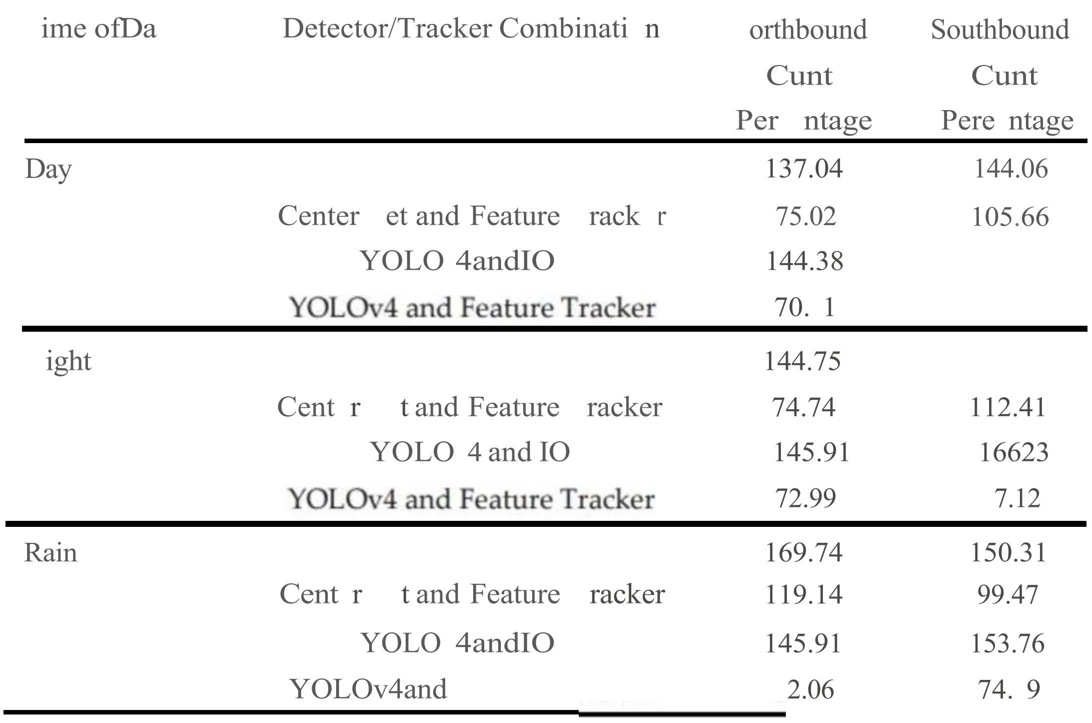
Faster R-CNN and YOLO algorithms were deployed to study stationary vehicles. To comprehend and compare the test results for both Faster R-CNN and YOLO frameworks, confusion matrices and F-1 scores were used. The confusion matrix represents accuracy levels for different sections of image classification. Overall, 25 test results are shown in a 5 x 5 table that is referred to as a confusion matrix. Here, each row shows the actual number of predictions and total number of each row implies the number of targets predicted for that class. Likewise, every column signifies the true number of targets while the total number of each column represents the actual number of targets for that class. Similarly, F-1 score shown in equation (iv) is used to compare the performance of both Faster R-CNN and YOLO models. The results obtained for confusion matrix and F-1 scores are shown in Tables 1 and 2.

$$F-1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

As seen from Table 1, the performance of both Faster R-CNN and YOLO models were similar. Faster R-CNN was relatively inferior in detecting cyclist and bus but was better at detecting trucks when compared to the performance of YOLO. Both models predicted cars and pedestrians with a 99% level of accuracy. From Table 2, it is understood that the cumulative F-1 score of YOLO was lower than that of Faster R-CNN. Also, the recall value for YOLO was lower which implies that YOLO detects fewer objects on a traffic scene compared to Faster R-CNN. After comparing results in Table 2, it appears that Faster-RCNN was slightly better but comparable to YOLO. Therefore, any one of them could be used as an object detector

Vehicle Counts

With the advent of ITS, vehicle counts are often automated using either loop detectors or vision based systems. Although inductive loops give out accurate traffic counts, they often have trouble distinguishing the type of vehicles (i.e. cars, trucks, buses, etc.). Not to forget that these detectors are intrusive. On the contrary, vision based systems' non-intrusive nature enables counts by different vehicle class types with high confidence scores [43, 44]. Since, accurate vehicle count enable TMCs and other transportation agencies apply them in their day to day application areas, the significance of accurate vehicle counts cannot be ignored. Studies such as daily volume counts, travel times calculation, and traffic forecasts are all precursors of an accurate vehicle counting system. These parameters serve as important tools for optimizing traffic at different roadways. Similarly, counting information also enable engineers to obtain future traffic forecasts which in turn helps identify what routes are utilized extensively to lay out affirmative planning decisions. In this study, we aim at developing a single look vehicle counting system that could automatically detect and tabulate the number of vehicles passing through the road. To accurately perform vehicular count, the vehicles are detected using object detectors and then traced through trackers. To obtain vehicle counts, the trackers are set an IOU threshold of 0.5 as shown in equation (vi) which helps correctly track vehicles and avoids multiple counts. $IOU = Intersection\ Union$ *(vi)* To assess the performance of the proposed models, the number of vehicles passing through the north and southbound directions were manually counted and compared against the automatic counts obtained from the combination of two different object detectors and trackers. CenterNet and YOLOv4 were the two different object detectors used in combination with IOU and Feature Tracker. For comparison, these frameworks were tested on a total of 546 video clips each 1 minute in length comprising of over 9 hours of total video length.



ime ofDa	Detector/Tracker Combinati n	orthbound Cunt Per ntage	Southbound Cunt Pere ntage
Day		137.04	144.06
	Center et and Feature rack r	75.02	105.66
	YOLO 4andIO	144.38	
	YOLOv4 and Feature Tracker	70. 1	
ight		144.75	
	Cent r t and Feature racker	74.74	112.41
	YOLO 4 and IO	145.91	16623
	YOLOv4 and Feature Tracker	72.99	7.12
Rain		169.74	150.31
	Cent r t and Feature racker	119.14	99.47
	YOLO 4andIO	145.91	153.76
	YOLOv4and	2.06	74. 9

Fig 4.6 Vehicle Count Performance

Table 4 demonstrates the performance comparison of CenterNet and YOLOv4 models in different conditions. The performance of these detector-tracker frameworks is assessed by dividing the values obtained from them with the manually counted ground truths expressed in per hundredth or percentage. As seen from Table 4, the combination of YOLOv4 and Feature tracker obtained a reasonable counting performance for all the three different environmental conditions specified. For model combinations where a count percentage of over one hundred was achieved, there was clearly some fault in both detector and tracker. The reasoning behind the detector-tracker combination achieving over 100 percent accuracy is largely to do with the object detector generating multiple bounding boxes for the same vehicle. This resulted in overcounting of vehicles. Similarly, IOU at times did not do very well at predicting vehicle trajectories and identified them as disparate vehicles.

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 ALGORITHM

Convolutional Neural Network

Convolutional Neural Network is a Deep Learning algorithm specially designed for working with Images and videos. It takes images as inputs, extracts and learns the features of the image, and classifies them based on the learned features.

This algorithm is inspired by the working of a part of the human brain which is the Visual Cortex. The visual Cortex is a part of the human brain which is responsible for processing visual information from the outside world. It has various layers and each layer has its own functioning i.e each layer extracts some information from the image or any visual and at last all the information received from each layer is combined and the image/visual is interpreted or classified.

Similarly, CNN has various filters, and each filter extracts some information from the image such as edges, different kinds of shapes (vertical, horizontal, round), and then all of these are combined to identify the image.

Now, the question here can be: Why can't we use Artificial Neural Networks for the same purpose? This is because there are some disadvantages with ANN:

- It is too much computation for an ANN model to train large-size images and different types of image channels.
- The next disadvantage is that it is unable to capture all the information from an image whereas a CNN model can capture the spatial dependencies of the image.
- Another reason is that ANN is sensitive to the location of the object in the image i.e if the location or place of the same object changes, it will not be able to classify properly.

COMPONENTS OF CNN

The CNN model works in two steps: feature extraction and Classification Feature Extraction is a phase where various filters and layers are applied to the images to extract the information and features out of it and once it's done it is passed on to the next phase i.e Classification where they are classified based on the target variable of the problem.

A typical CNN model looks like this:

- Input layer
- Convolution layer + Activation function
- Pooling layer
- Fully Connected Layer

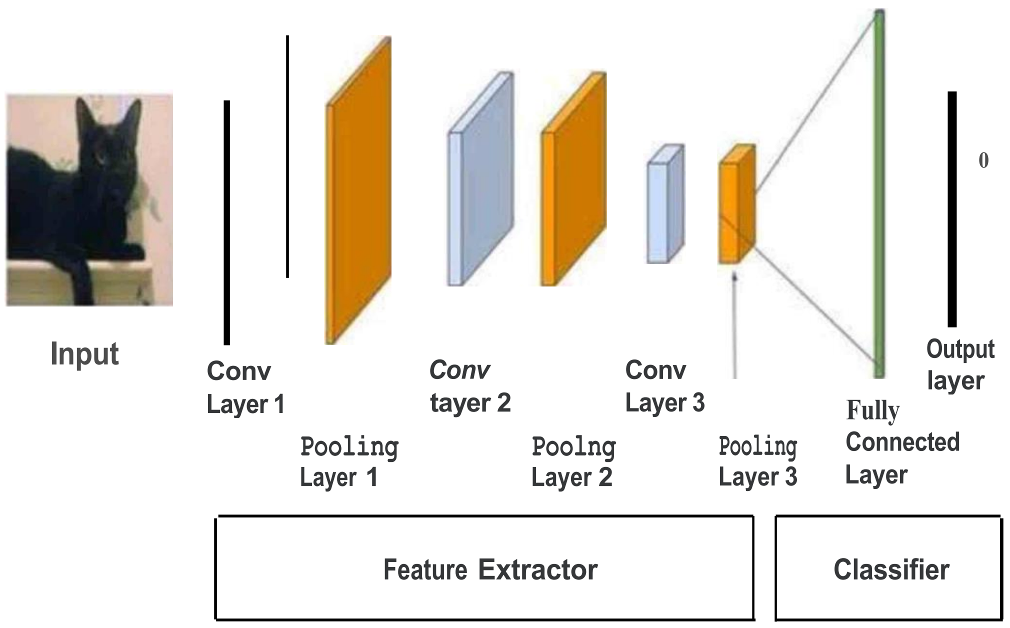
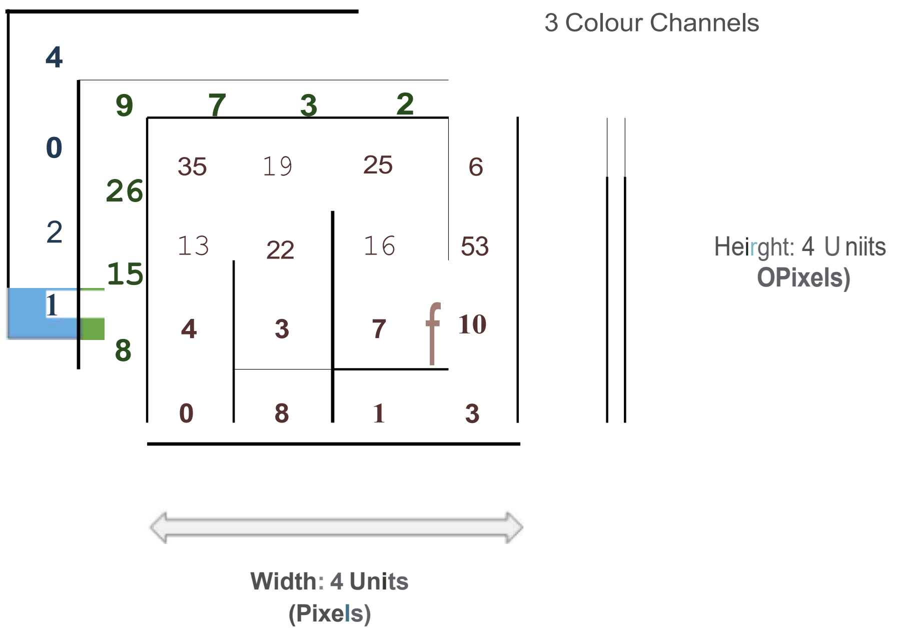


Figure 5.1 Layers of CNN

INPUT LAYER

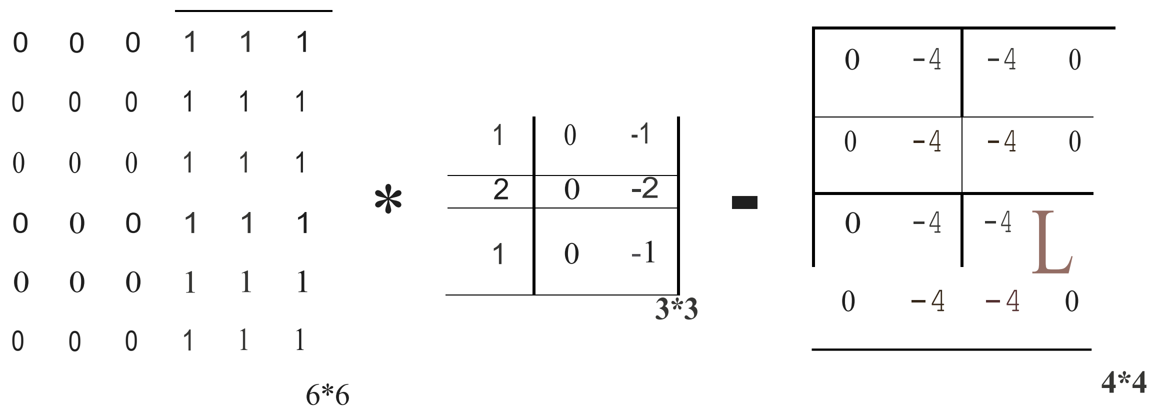
As the name says, it's our input image and can be Grayscale or RGB. Every image is made up of pixels that range from Oto 255. We need to normalize them i.e convert the range between Oto 1 before passing it to the model.

Below is the example of an input image of size 4*4 and has 3 channels i.e RGB and pixel values.



CONVOLUTION LAYER

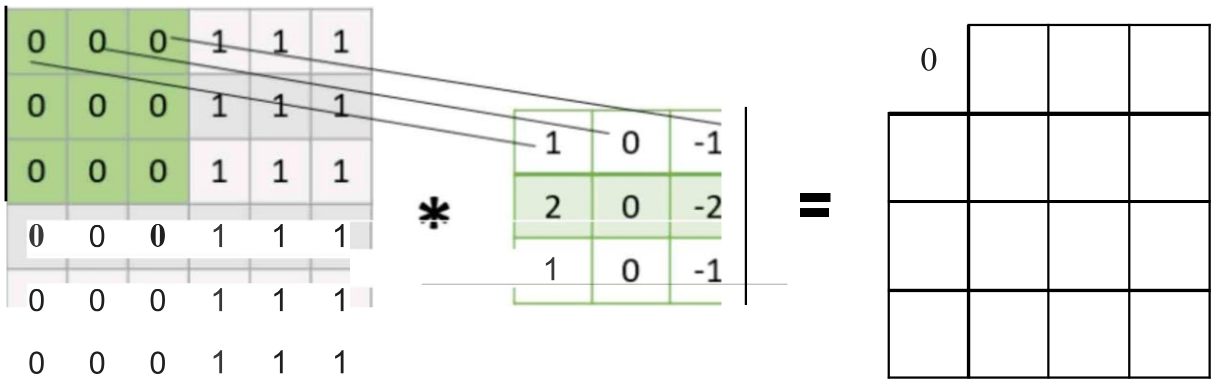
The convolution layer is the layer where the filter is applied to our input image to extract or detect its features. A filter is applied to the image multiple times and creates a feature map which helps in classifying the input image. Let's understand this with the help of an example. For simplicity, we will take a 2D input image with normalized pixels.



In the above figure, we have an input image of size 6*6 and applied a filter of 3*3 on it to detect some features. In this example, we have applied only one filter but in practice, many such filters are applied to extract information from the image.

The result of applying the filter to the image is that we get a Feature Map of 4*4 which has some information about the input image. Many such feature maps are generated in practical applications.

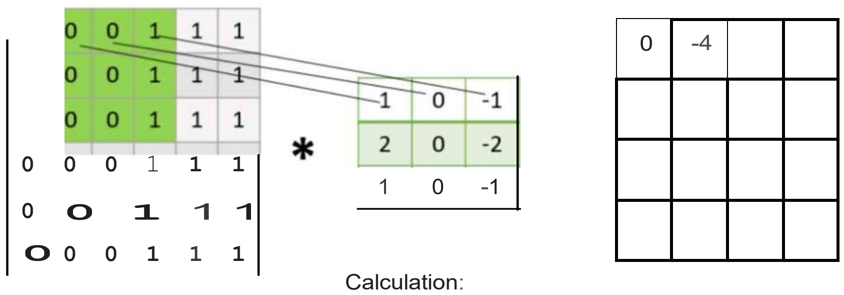
Let's get into some maths behind getting the feature map in the above image.



Calculation:
O*l + 0 0 + 0*-1 +
0*2 + 0 0 + 0*-2 +
O*l + 0 0 + 0 -1

As presented in the above figure, in the first step the filter is applied to the green highlighted part of the image, and the pixel values of the image are multiplied with the values of the filter (as shown in the figure using lines) and then summed up to get the final value.

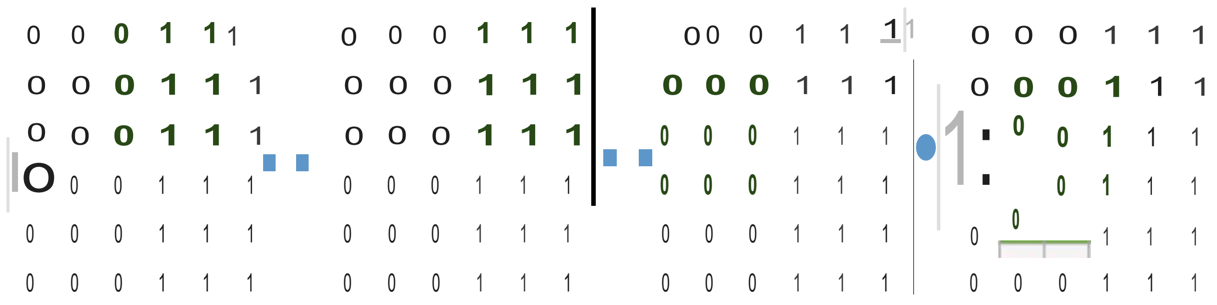
In the next step, the filter is shifted by one column as shown in the below figure. This jump to the next column or row is known as stride and in this example, we are taking a stride of 1 which means we are shifting by one column.



Calculation:
0*l+0*0+l*-1+
0*2 + 0*0 + 1 *-2 +
0*1 + 0*0 + 1 *-1

Similarly, the filter passes over the entire image and we get our final Feature Map. Once we get the feature map, an activation function is applied to it tor introducing nonlinearity.

A point to note here is that the Feature map we get is smaller than the size of our image. As we increase the value of stride the size of the feature map decreases.



This is how a filter passes through the entire image with the stride of 1

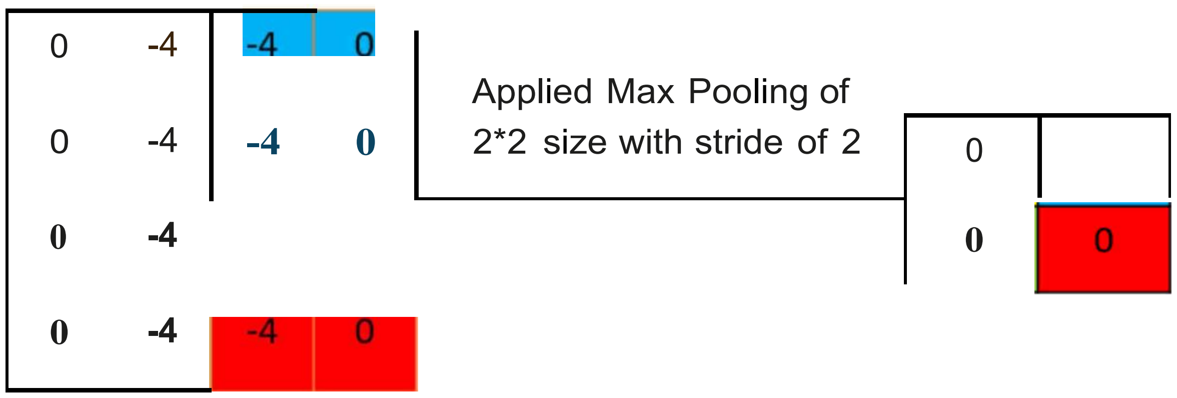
Pooling Layer

The pooling layer is applied after the Convolutional layer and is used to reduce the dimensions of the feature map which helps in preserving the important information or features of the input image and reduces the computation time.

Using pooling, a lower resolution version of input is created that still contains the large or important elements of the input image.

The most common types of Pooling are Max Pooling and Average Pooling. The below figure shows how Max Pooling works. Using the Feature map which we got from the above example to apply Pooling. Here we are using a Pooling layer of size 2*2 with a stride of 2.

The maximum value from each highlighted area is taken and a new version of the input image is obtained which is of size 2*2 so after applying Pooling the dimension of the feature map has reduced.



Fully Connected Layer

Till now we have performed the Feature Extraction steps, now comes the Classification part. The Fully connected layer (as we have in ANN) is used for classifying the input image into a label. This layer connects the information extracted from the previous steps (i.e Convolution layer and Pooling layers) to the output layer and eventually classifies the input into the desired label.

The complete process of a CNN model can be seen in the below image.

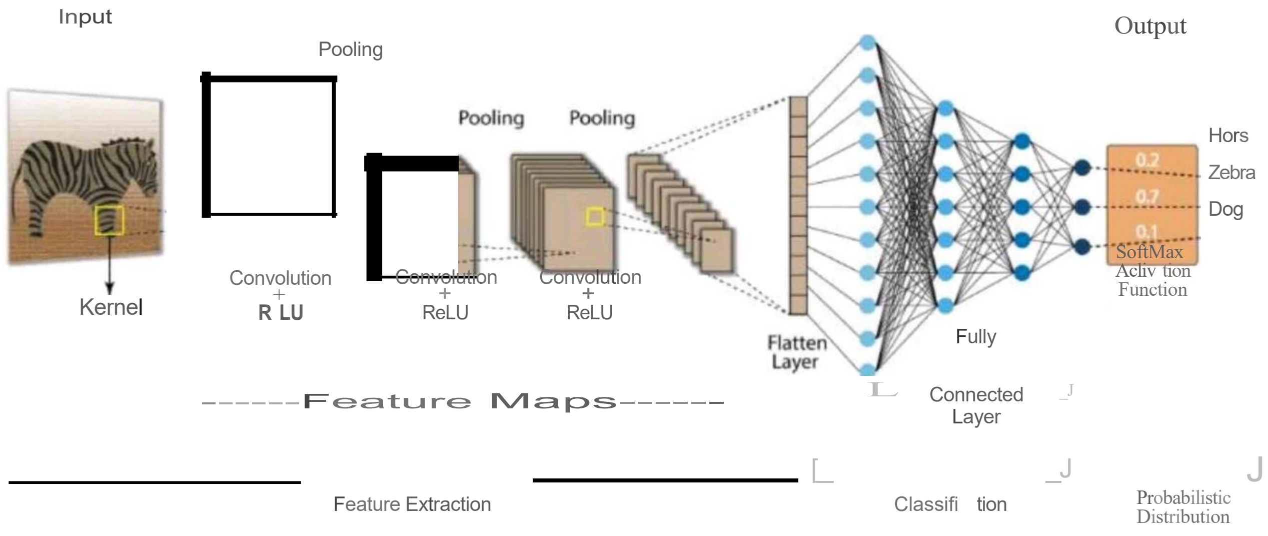


Figure 5.1.1 Convolutional neural network

5.2 TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or - one step up - software applications at the company level - interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER 6

RESULT & DISCUSSION

6.1 RESULT

The AI enabled vehicle counting system is a computer vision-based solution that uses advanced algorithms and deep learning models to detect and count vehicles in real-time. This system can be used for various applications such as traffic management, parking management, and surveillance. The results of the AI enabled vehicle detection system are typically highly accurate and reliable. The system uses advanced machine techniques, such as object detection and tracking, to detect and count vehicles. These techniques are based on deep learning models that have been trained on large datasets of images and videos of vehicles. As a result, the system can accurately detect and track vehicles in various conditions, including different lighting conditions, weather conditions, and different types of vehicles. The AI vehicle counting system can also provide real-time analytics and insights into vehicle traffic patterns. By analyzing the data collected from the system, traffic management authorities can identify peak traffic periods, congestion hotspots, and other traffic-related issues. This info can then be used to ease traffic flow and reduce congestion, resulting in faster and more efficient transportation. However, the effectiveness of the AI vehicle counting system depends on various factors such as camera placement, camera angle, and the quality of the camera used. The system may not be able to accurately count vehicles if the camera is not positioned correctly or if the camera quality is poor. Additionally, the system may have difficulty distinguishing between vehicles of similar sizes and shapes, such as cars, trucks, buses, motorcycles and bicycles. In conclusion, the AI vehicle counting system is a highly effective and reliable solution for counting vehicles in real-time. However, the accuracy and reliability of the system depend on various factors, and it is important to carefully consider these factors when implementing the system for a specific application.

CHAPTER 7

CONCLUSION

7.1 CONCLUSION

The rapid progression in the field of deep learning and high-performance computing has highly augmented the scope of video-based traffic monitoring systems. In this study, an automatic traffic monitoring system is developed that builds up on robust deep learning models and facilitates traffic monitoring using a graphical user interface. Deep learning algorithms, such as Mask R-CNN, Faster R-CNN, YOLO and CenterNet were implemented alongside two different object tracking systems viz. IOU and Feature

Tracker. Mask R-CNN was used to detect traffic queues from real-time traffic CCTVs whereas YOLO and Faster R-CNN were deployed to predict objects which later coupled with object trackers were used to detect stationary vehicles. Mask R-CNN predicted traffic queues with 92.8% accuracy while the highest accuracy attained by YOLO was 95.5%. The discrepancy in correctly detecting queues was mainly due to the poor image quality, queues being distant from the camera and glaring effects. These issues significantly affected the accuracies of the proposed models. Similarly, the F1, RMSE and S3 scores for detecting stationary vehicles were 0.8333, 154.7741, and 0.4034 respectively. It was observed that the model correctly
intersections.

In that case, the system proposed in this paper could be a cheaper and reliable alternative to bringing in some level of traffic automation by supplementing it with some additional low-cost back-up software suites.

7.2 FUTURE WORK

As a result, future studies may focus on determining the minimum needed length between the chamber and the junction region. Furthermore, further study is needed for roads to build vehicle position correction algorithms. Training with the heavy vehicle is also essential to enhance recognization rates. Future research should take these restrictions into account in order to further enhance image processing-based traffic monitoring systems.

REFERENCES:-

1. Land, E. H., An alternative technique for the computation of the designator in the retinex theory of color vision. Proceedings of the national academy of sciences 1986, 83, (10), 3078-3080.

2. Rahman, Z.-u.; Jobson, D. J.; Woodell, G. A. In Multi-scale retinex for color image enhancement, Proceedings of 3rd IEEE International Conference on Image Processing, 1996; IEEE: 1996; pp 1003-1006.

3. He, K.; Sun, J.; Tang, X., Single image haze removal using dark channel prior. IEEE transactions on pattern analysis and machine intelligence 2010, 33, (12), 2341-2353.

4. GitHub, Video Demonstration of a GUI based AI Enabled Traffic Monitoring System. Available Online: https://github.com/titanmu/aienabled. In 2020.

5. Willis, C.; Harborne, D.; Tomsett, R.; Alzantot, M. In A deep convolutional network for traffic congestion classification, Proc NATO IST-158/RSM-010 Specialists' Meeting on Content Based Real-Time Analytics of Multi-Media Streams, 2017; 2017; pp 1-11.

6. Chakraborty, P.; Adu-Gyamfi, Y. O.; Poddar, S.; Ahsani, V.; Sharma, A.; Sarkar, S., Traffic congestion detection from camera images using deep convolution neural networks. Transportation Research Record 2018, 2672, (45), 222-231.

7. Morris, T.; Schwach, J. A.; Michalopoulos, P. G., Low-cost portable video-based queue detection for work-zone safety. 2011.

8. Fouladgar, M.; Parchami, M.; Elmasri, R.; Ghaderi, A. In Scalable deep traffic flow neural networks for urban traffic congestion prediction, 2017 International Joint Conference on Neural Networks (IJCNN), 2017; IEEE: 2017; pp 2251-2258.

9. Ma, X.; Yu, H.; Wang, Y.; Wang, Y., Large-scale transportation network congestion evolution prediction using deep learning theory. PloS one 2015, 10, (3), e0119044.

10. Wang, J.; Gu, Q.; Wu, J.; Liu, G.; Xiong, Z. In Traffic speed prediction and congestion source exploration: A deep learning method, 2016 IEEE 16th International Conference on Data Mining (ICDM), 2016; IEEE: 2016; pp 499-508.

11. He, K.; Gkioxari, G.; Dollar, P.; Girshick, R. In Mask r-cnn, Proceedings of the IEEE international conference on computer vision, 2017; 2017; pp 2961-2969.

12. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y. M., YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv preprint arXiv:2004.10934 2020.

13. Ren, S.; He, K.; Girshick, R.; Sun, J. In Faster r-cnn: Towards real-time object detection with region proposal networks, Advances in neural information processing systems, 2015; 2015; pp 91-99.

14. Duan, K.; Bai, S.; Xie, L.; Qi, H.; Huang, Q.; Tian, Q. In Centernet: Keypoint triplets for object detection, Proceedings of the IEEE International Conference on Computer Vision, 2019; 2019; pp 6569-6578.

15. Rakhimkul, S.; Kim, A.; Pazylbekov, A.; Shintemirov, A. In Autonomous Object Detection and Grasping Using Deep Learning for Design of an Intelligent Assistive Robot Manipulation System, 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), 2019; IEEE: 2019; pp 3962-3968.

APPENDIX

A.SOURCE CODE

```
from track import*
import tempfile
import cv2
import torch
import streamlit as st
import os


if    name_ --       main    '·
st.title('Vehicle detection and counting')
st.markdown('<h3 style="color: red"> with Yolov5 and Deep SORT
<lh3', unsafe_allow_html=True)

# upload video
video_file_buffer = st.sidebar.file_uploader("Upload a video",
type=['mp4', 'mov', 'avi'])


if video_file_buffer:
st.sidebar. text('Input video')
st.sidebar. video(video_file_buffer)
# save video from streamlit into "videos" folder for future detect
with open(os.path.join('videos', video_file_buffer.name), 'wb') as f:
f.write(video_file_buffer.getbuffer())

st.sidebar.markdown('---')
st.sidebar.title('Settings')
# custom class
custom_class = st.sidebar.checkbox('Custom classes')
assigned_class_id = [O, 1, 2, 3]
names = ['car', 'motorcycle', 'truck', 'bus']

if custom_class:
assigned_class_id = []
assigned_class = st.sidebar.multiselect('Select custom classes',
list(names))
```

```
for each in assigned_class:
assigned_class_id.append(names.index(each))

# st.write(assigned_class_id)

# setting hyperparameter
confidence = st.sidebar.slider('Confidence', min_value=0.0,
max_value=1.0, value=0.5)

line = st.sidebar.number_input('Line position', min_value=0.0,
max_value=1.0, value=0.6, step=0.1)
st.sidebar. markdown('---')



status = st.empty()
stframe = st.empty()
if video_file_buffer is None:
status.markdown('<font size= "4"> **Status:** Waiting for input </font>',
unsafe_allow_html=True)
else:
status.markdown('<font size= "4"> **Status:** Ready </font>',
unsafe_allow_html=True)

car, bus, truck, motor= st.columns(4)
with car:
st.markdown('**Car**')
car_text = st.markdown('_')

with bus:
st.markdown('**Bus**')
bus_text = st.markdown('_')

with truck:
st.markdown('**Truck**')
truck_text = st.markdown('_')

with motor:
st.markdown('**Motorcycle**')
motor_text = st.markdown('_')
```

```
fps,_, _, _ = st.columns(4)
with fps:
st.markdown('**FPS**')
fps_text = st.markdown(' __')



track_button = st.sidebar.button('START')
# reset_button = st.sidebar.button('RESET ID')
if track_button:
# reset ID and count from O
reset()
opt = parse_opt()
opt.conf_thres = confidence
opt.source= f'videos/{video_file_buffer.name}'

status.markdown('<font size= "4"> **Status:** Running... </font>',
unsafe_allow_html=True)
with torch.no_grad():
detect(opt, stframe, car_text, bus_text, truck_text, motor_text, line,
fps_text, assigned_class_id)
status.markdown('<font size= "4"> **Status:** Finished ! </font>',
unsafe_allow_html=True)
# end_noti = st.markdown('<center style="color: blue"> FINISH
</center>', unsafe_allow_html=True)

# if reset_button:
# reset()
# st.markdown('<h3 style="color: blue"> Reseted ID <lh3>',
unsafe_allow_html=True)
```

B.SCREENSHOTS

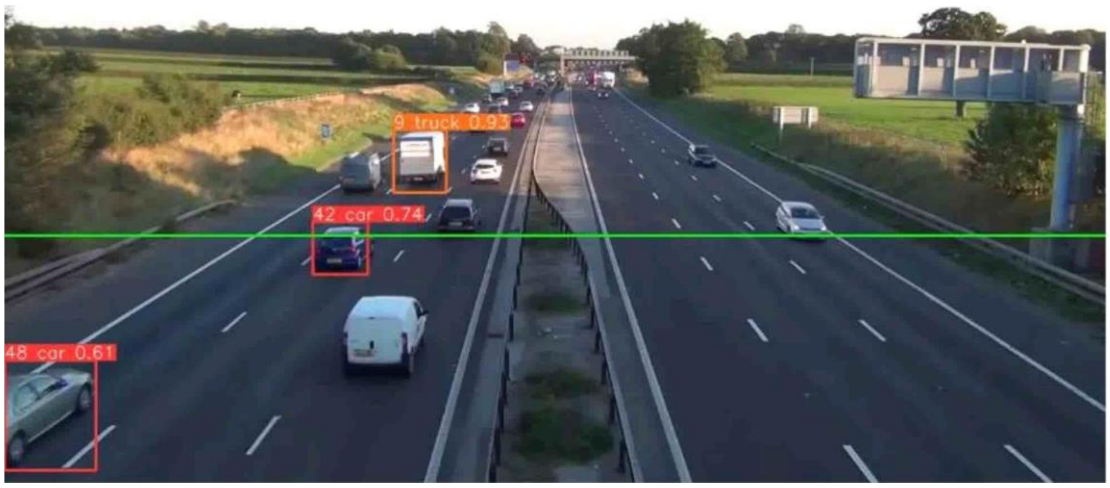
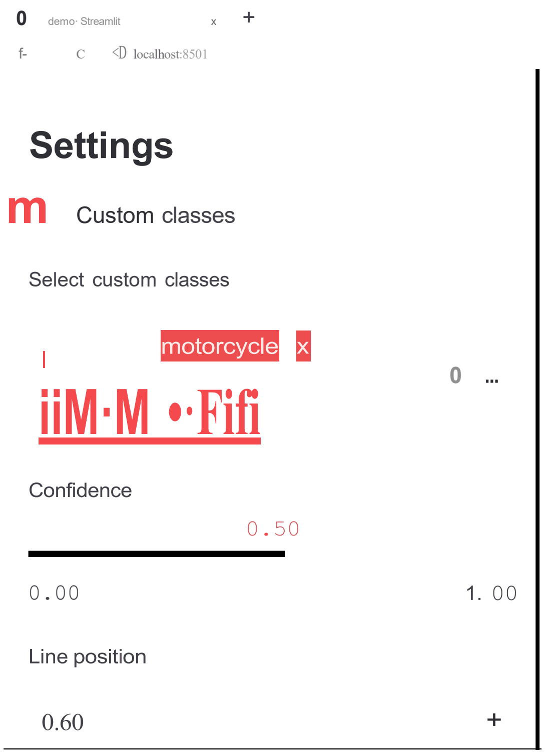


Figure 1. Tracking of vehicles



Settings

Custom classes

Select custom classes

motorcycle x

iiM·M •·Fifi

Confidence

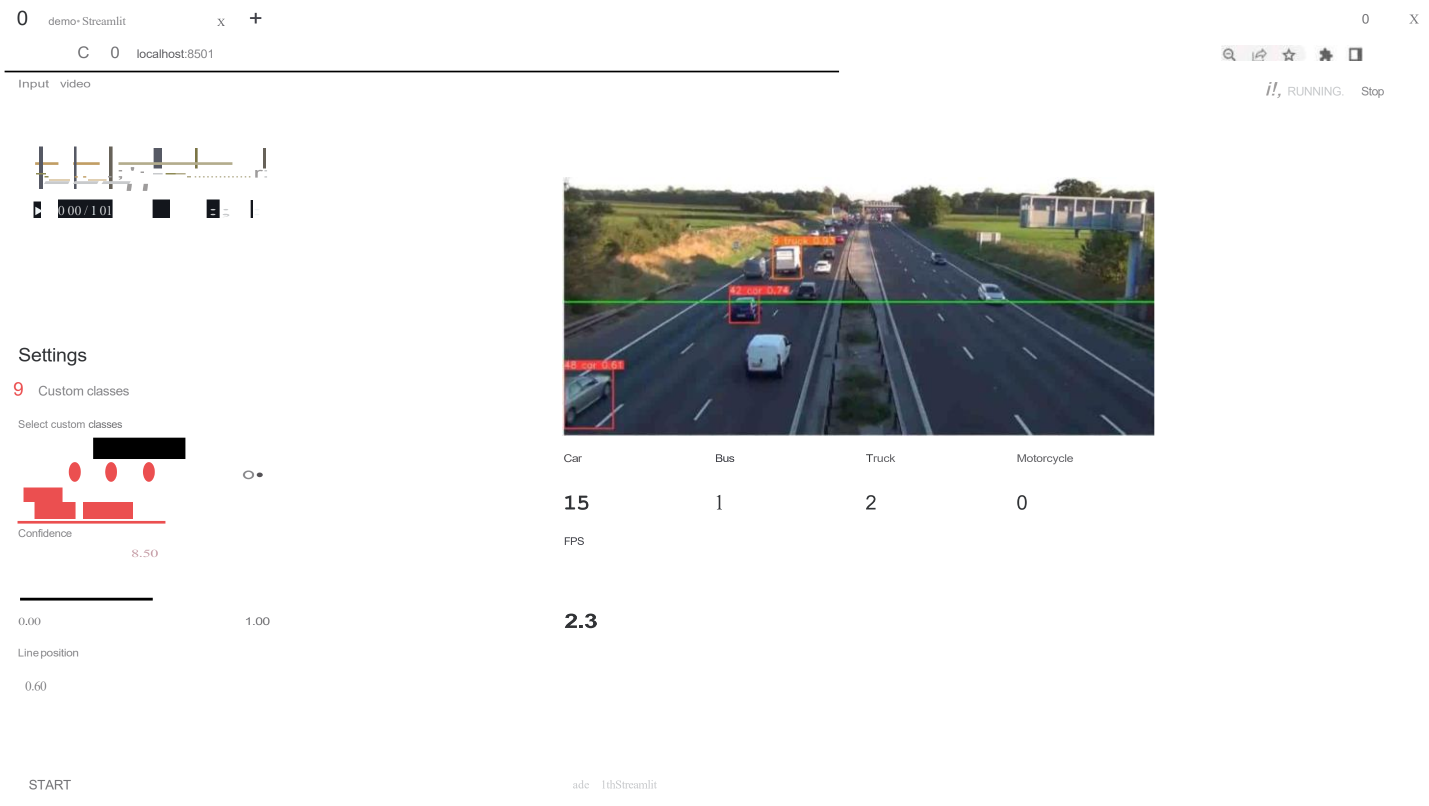
0.50

0.00 1. 00

Line position

0.60 +

Figure 2. Classification of vehicles types



demo· Streamlit

localhost:8501

Input video

0 00 / 1 01

RUNNING. Stop

Settings

Custom classes

Select custom classes

Confidence

8.50

0.00 1.00

Line position

0.60

START

Car	Bus	Truck	Motorcycle
15	1	2	0

FPS

2.3

ade 1thStreamlit

Figure 3. Final Output

Artificial Intelligence Enabled Traffic Monitoring System

Rojar Benny K
UG Scholar
Department of IT
Sathyabama Institute of Science and Technology
Chennai, India
bennyk72002@gmail.com

Senduru Srinivasalu B
Associate Professor
Department of IT
Sathyabama Institute of Science and Technology
Chennai, India

Jushwanth Kumaar T
UG Scholar
Department of **IT**
Sathyabama Institute of Science and Technology
Chennai, India
jushwanthk@gmail.com

Abstract- **As a result of the numerous cameras that Traffic Management Centers put over a network, personally checking on visitors might be extremely unsettling. Small-scale automation should make it simpler to make proactive judgements and decrease the negative effects that accidents and routine traffic congestion have on roads. This paper provides a novel method for regularly assessing real-time images using a standalone GUI and deep convolutional neural networks. The authors outline the results of research done as part of a system for creating clothes that has a framework for a device that can track people using artificial intelligence. The proposed machine, which utilises a number of state-of-the-art deep learning techniques, may automate specific visitor tracking requirements.A enormous library of annotated video surveillance data is used to teach deep learning-based designs how to detect queues and count cars. With the help of a pixel-stage segmentation technique, visitor lines may be recognised and their severity can be predicted. Real-time item detection algorithms are combined with special monitoring structures to frequently find stalled motors and do vehicle counts. Exciting experimental findings are presented that show how successful the suggested machine is at all phases of development. Overall, the findings demonstrate that the suggested framework performs admirably in a range of circumstances without being considerably impacted by environmental risks like foggy digital camera images, dim lighting, rain, or snow.**

Keywords - Renn, cnn, yolo, Django.

Introduction

Ensuring efficient traffic monitoring has been acknowledged as a crucial aspect of transportation engineering for a considerable time. Currently, most traffic monitoring centres depend on human operators to oversee traffic movements and road accidents. However, this manual approach can be intricate and time-consuming. Human weariness and inaccuracy usually cause disparities in the outcomes. To avoid burdening human operators and improve output efficiency, it is beneficial for all to develop automated traffic monitoring technologies. Consequently, the development of self-governing traffic monitoring systems has become a significant area of research within intelligent transportation systems. It is noteworthy that Traffic Management Centers rely predominantly on camera-based systems to monitor traffic presently. However, because the bulk of existing vision-based systems are controlled by humans, effectively tracking congestion, detecting stationary cars, and tracking the number of vehicles is difficult. As a result, traffic management centres (TMCs) have been striving to introduce various levels of traffic management automation.Artificial Intelligence (AI) based automated traffic monitoring systems are capable of effectively managing and assessing present situations while also minimizing the occurrence of road accidents. These systems have the ability to identify and track each vehicle's unique movement pattern to detect unsafe driving practices like sudden lane changes. Additionally, the AI-enabled system can accurately detect parked cars on the road, which is a crucial feature. When cars are left immobile, they obstruct the flow of vehicles ahead of them and cause them to stack up. Congestion prohibits cars from moving freely as a result.

II. LITERATURE REVIEW

The use of AI-enabled vehicle counting systems has attracted considerable attention in recent years due to their accuracy, efficiency, and potential to provide valuable insights into traffic patterns and trends. Here are some key findings from the literature

review:
Computer Vision Techniques: AI-enabled vehicle counting systems rely on computer vision techniques, such as object detection and image segmentation, to identify and track vehicles in real-time. These approaches have advanced greatly in recent years as a result of improvements in deep learning algorithms, which have enabled models to be developed on massive datasets of annotated images.
Types of Systems: There are several types of AI-enabled vehicle counting systems, including those that use cameras, infrared sensors, or other types of sensors to detect vehicles. Camera-based systems are the most common, as they offer high resolution and a wide field of view, making it possible to detect and count vehicles accurately over a large area.
Accuracy: The accuracy of AI-enabled vehicle counting systems varies depending on several factors, including lighting conditions, camera angle, and the type of vehicle being counted. In general, these systems have been found to be more accurate than traditional manual counting methods, with error rates as low as 1-2%.

Real-World Applications: AI-enabled vehicle counting systems have been deployed in a variety of real-world applications, including traffic management, parking management, and toll collection. In traffic management, these systems can be used to optimize traffic flow and reduce congestion, while in parking management, they can be used to monitor occupancy levels and identify available parking spaces.
Challenges: There are several challenges associated with AI-enabled vehicle counting systems, including the need for high-quality video footage, the potential for privacy violations, and the risk of system failures. To address these challenges, researchers are exploring new techniques for enhancing the accuracy and reliability of these systems, while also developing new privacy-preserving algorithms and protocols.
Overall, the literature suggests that AI-enabled vehicle counting systems have significant potential for improving traffic management and other applications, and that ongoing research is likely to lead to further improvements in accuracy, efficiency, and reliability.

III. METHODOLOGY

The primary element of an independent traffic monitoring system's technology is a GPU-powered backend (on-premise) that enables swift training of highly complex models, and rapid deployment across various cameras in nearly real-time. These systems use deep convolutional neural network models that can identify and classify multiple objects, as well as segment traffic scenes.
To train these models, hand-annotated traffic images were used as the primary dataset. The system uses several techniques to track samples and create trajectories for each identified item on the traffic scene in real time, resulting in a high level of situational awareness. Combining these techniques, the system can obtain multiple traffic flow variables, such as traffic volume and occupancy, & monitor various traffic scenarios, like queues, accidents, & unusual occurrences in the traffic scene.

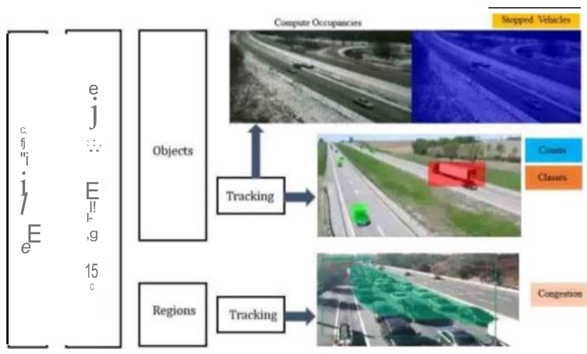


Figure 1. Visual Representation of the Proposed AI-Enabled System

The AI-powered traffic monitoring system can detect and pinpoint congestion, track stationary cars in real-time, maintain track of the number of different vehicle types, and more. The cornerstone of the AI-powered traffic tracking system is its constituent objects. The primary dataset used to train these models comprised manually annotated traffic images. The system employs several object-tracking algorithms to continuously generate trajectories for every object detected in the traffic scene, resulting in heightened situational awareness. These processes are subsequently integrated to derive various traffic flow variables, including traffic volume and occupancy, and to monitor diverse traffic situations such as queues, accidents, and peculiarities in the traffic scene. The AI-powered traffic monitoring system can spot and detect congestion, track stationary vehicles in real time, tabulate the number of various vehicle classes, and more.

Faster R-CNN

The Faster RCNN [13] is a target detection algorithm that functions in two stages. It utilizes a region proposal network (RPN) in the framework, which combines convolutional features from the entire image with a dataset to generate region proposals without additional cost.At each place, the RPN anticipates both the object limits and the associated score values.Faster R-CNN predicts objects using high-quality region suggestions generated by RPN's end-to-end training.

The us .of RPN in Faster R-CNN improves object recognition to a greater extent than the selective ea_r h m thod used in Fast R-CNN. Each image Is d1v!ded into a variety of compact portions by the algorithm. Afterwards the, each section is evaluated via a succession of convolutional filters to extract high-quality feature descriptors, which are then input into a classifier. The classifier then g nerates the likelihood of items in each part of a picture. The model has been trained for five classes: the, each section is evaluated via a succession of convolutional filters to elevated !iltration fea_t re descriptors, which are then input into a class1f1er. The classifier then generates the likelihood of items in each part of a picture. The model has been trained for five classes: car, truck, bus, bicycle, and pedestrian. Training took roughly 8 hours on an NVIDIA GTX 1080Ti GPU. The model handled video streams at a rate of five fps.

Mask R-CNN

Mask R-CNN, which is also known as Mask-region based Convolutional Neural Network [11], is a modified version of Faster R-CNN that improves upon it by introducing more accurate masks and segmenting the region of interest at the pixel level. Additionally, Mask R-CNN can perform all the tasks that Faster R-CNN can accomplish. In this study, the Resnet 101 backbone and the Feature Pyramid Network (FPN), which serves as the model's basis, are both utilized. In this scenario, the model's feature extractor was ResNet101. Adding a second pyramid that collected higher-medium features from the first pyramid and then passes them over to lower tiers enhanced the conventional feature extraction pyramid when using FPN. The minimal detection confidence rate was established at 90% in this investigation, which employed 50 validation steps. Using an image-centric training paradigm, each picture was trimmed to the shape of a square.

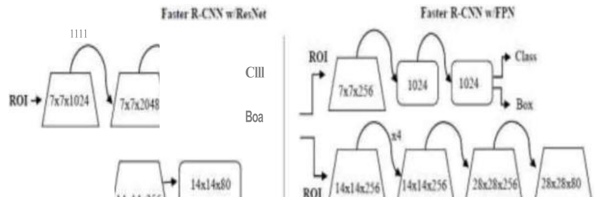


Figure 2. Mask R-CNN Framework

YOLO

In this, the modem object detection method You Only Look Once (YOLO) [12] is employed. Unlike traditional object detection algorithms, YOLO merely looks at an image once to determine whether there are any objects. In this experiment, YOLOv4 was employed to determine vehicles, count vehicles, and examine the results for the establishment of traffic queues. Most advanced object detection methods recycle CNN classifiers to conduct detections. These techniques, for example, utilise a classifier to discover the item and test it at various sizes and positions throughout the test picture. YOLO, on the other hand, reframes object detection, which means it only has to look at a picture once to properly guess what is there rather than scanning through it a thousand times. The YOLO model employs a lone CNN that can predict numerous bounding boxes and their associated class probabilities concurrently. YOLO models take an average of 20 to 30 hours to build. Mask RCNN and YOLO were both trained on the same hardware.

CenterNet

CenterNet [14] recognises visual patterns in each segment of an image that is clipped at a lower computing cost than previous approaches. CenterNet uses a triplet to identify objects instead of a pair of key points, which enhances the precision and recall. The model addresses the disadvantage of ComerNet [35], which detects objects using two comer-keypoints. ComerNet, on the other hand, falls short of providing a high global picture of an object, which CenterNet does by including an extra keypoint to gather more core info

about a picture. CenterNet discovers objects by employing a central key point that is likely to belong to an identical category if the detected bounding box has a high crossroads over union (IoU) with the ground-truth box. This method enhances accuracy and recall and is more successful than

anchor-based detection approaches that employ simply pairs of keypoints. Although utilising a triplet, CenterNet is a single-stage detector that benefits from Roi pooling. It creates two comer and one centre keypoint heatmaps utilizing cascade comer and centre pooling on a CNN backbone. Figure 3 depicts this procedure. The CenterNet centre pooling enables the centre keypoints to sense the core area of the bounding box and discover extra recognised visual patterns inside the objects. On the other hand, cascade comer pooling aids the first comer pooling module in getting internal info. The approach of predicting a bounding box is as like as Comer Net, which employs a pair of observed comers and good embeddings. Lastly, the detected centre keypoints are utilised to construct the eventual bounding boxes.

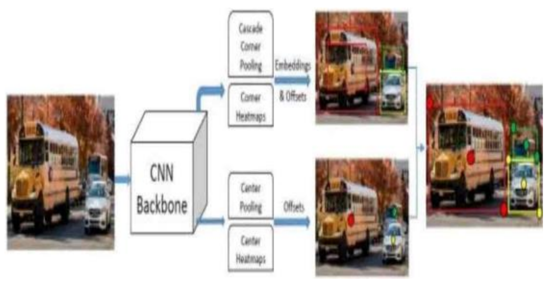


Figure 3. Architecture of CenterNet

Monitoring Traffic Queues

Figure 4 depicts the mechanism utilised by the intelligent queue monitoring system. The VGG imagery annotator [36] is used for the initial stage of the annotation. The annotated photos are then utilised to develop the Mask RCNN and YOLO models. Mask R-CNN and YOLO training times are around 3.5 hours and 22 hours, correspondingly.
Following training, the models were evaluated using live traffic recordings. The key rationale for utilising mask R-CNN is its capacity to create pixel-level segmentation masks, which improves the accuracy of meat detection. Because YOLO detects using bounding boxes, it can identify both crowded and uncongested locations. As a result, Mask-RCNN beats YOLO in properly forecasting categorization zones of concern.

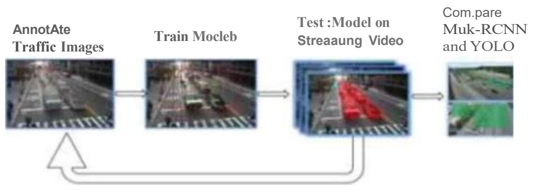


Figure 4. Flowchart of step-wise operations

Detection of Stationary Vehicles

The proposed approach for identifying stopped or trapped cars is depicted in Figure 5. To begin the procedure, the YOLO model is taught to recognize vehicles. The findings are then followed by using Intersection Over Union (IOU) procedure, and each vehicle's journey is displayed against the traffic location. The tracking findings are then utilised to depict specific driving directions (east, west, north, or south), the kind of route being evaluated (intersections or motorways), and assess the speed of target vehicle. Following that, the followup data is employed to account for discrete driving directions, road kinds, and projected vehicle speeds. If the speed of a vehicle remains at a certain threshold for a considerable duration, the model can identify that the target is not moving on certain types of roads.

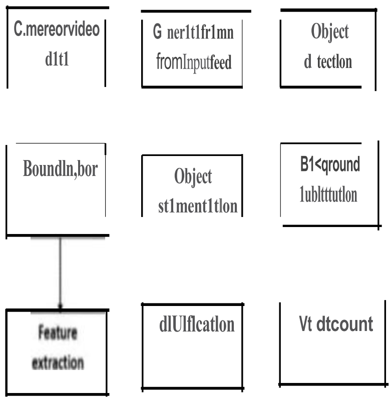


Figure 5. Flowchart for traffic anomaly detection

RESULT & DISCUSSION

The AI enabled vehicle counting system is a computer vision-based solution that uses advanced algorithms and deep learning models to detect and count vehicles in real-time. This system can be used tor various applications such as traffic management, parking management, and surveillance. The results of the AI enabled vehicle detection system are typically highly accurate and reliable. The system uses advanced machine techniques, such as object detection and tracking, to detect and count vehicles. These techniques are based on deep learning models that have been trained on large datasets of images and videos of vehicles. As a result, the system can accurately detect and track vehicles in various conditions, including different lighting conditions, weather conditions, and different types of vehicles.

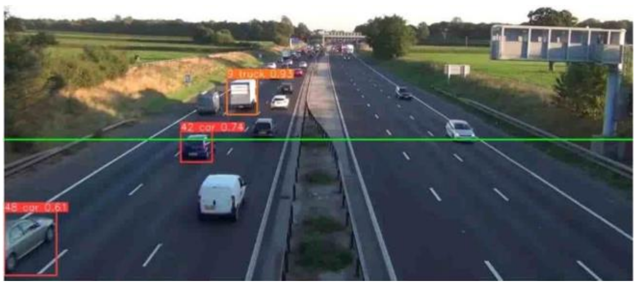


Figure 6. Vehicle Tracking

The AI vehicle counting system can also provide real-time analytics and insights into vehicle traffic patterns. By analyzing the data collected from the system, traffic management authorities can identify peak traffic periods, congestion hotspots, and other traffic-related issues. This info can then be used to ease traffic flow and reduce congestion, resulting in faster and more efficient transportation. However, the effectiveness of the AI vehicle counting system depends on various factors such as camera placement, camera angle, and the quality of the camera used. The system may not be able to accurately count vehicles if the camera is not positioned correctly or if the camera quality is poor. Additionally, the system may have difficulty distinguishing between vehicles of similar sizes and shapes, such as cars, trucks, buses, motorcycles and bicycles.

Figure 7. Final Output

In conclusion, the AI vehicle counting system is a highly effective and reliable solution tor counting vehicles in real-time. However, the accuracy and reliability of the system depend on various factors, and it is important to carefully consider these factors when implementing the system tor a specific application.

CONCLUSION

Because of the fast development of DL and high-spec computers, the capabilities of video-based traffic surveillance systems have substantially grown. This work creates an intelligent traffic surveillance system with a graphical user interface that is built on strong deep-learning models. Deep learning methods such as Feature Tracker and

IOU are used alongside two separate object monitoring systems, Mask R-CNN, Faster RCNN, YOLO, and CenterNet. Mask R-CNN was used to determine congestion queues from real-time traffic CCTVs, whilst YOLO and Faster R-CNN were used to forecast items, which were subsequently used with object trackers to identify stationary cars. When it came to predicting traffic lines, Mask R-CNN was 92.8% accurate, whereas YOLO was 95.5% accurate. The key reasons of the queue recognition mistake were low picture quality, the distance between the queues and the camera, and glaring impacts. This matter have a substantial influence on the accuracy of the suggested models. Similarly, the F1, RMSE, and S3 scores for recognizing stationary cars were 0.8333, 154.7741, and 0.4034, respectively. The algorithm properly detected stranded automobiles that remained near to the camera, but it struggled to identify immobile vehicles that were further away. Furthermore, factors such as pixelation in video and the presence of traffic junctions correlated to poorer S3 ratings. Regardless, measures such as anomaly reduction and pixelation in video adjustments improved the performance of the suggested model. It is crucial to highlight that these changes resulted in an effective stationary vehicle prediction system. Last but not least, the vehicle counting framework performed well in both CenterNet and YOLO's combinations with Feature Tracker. However, more study into the vehicle identification framework and further development of the existing models might be carried out in order to achieve a nearly flawless counting system. Because the bulk of transportation agencies significantly rely on turning movement counts to manage intersection traffic lights, this may be excellent. By including a few low-cost backup software suites into the system presented in this research, it may be a more cost-effective and trustworthy option to implementing some degree of traffic automation.

FUTURE SCOPE

As a result, future studies may focus on determining the minimum needed length between the chamber and the junction region. Furthermore, further study is needed for roads to build vehicle position correction algorithms. Training with the heavy vehicle is also essential to enhance recognization rates. Future research should take these restrictions into account in order to further enhance image processing-based traffic monitoring systems.

REFERENCES

1. Land, E. H., An alternative technique for the computation of the designator in the retinex theory of color vision. Proceedings of the national academy of Sciences 1986, 83, (10), 3078-3080.

2. Tao Gong, Bin Liu , Qi Chu, Nenghai Yu "Using multi-label classification to improve object detection", neurocomputing, ELSEVIER(2019).

3. YOLO real-time object detection https://pjreddie.com/darknet/yolo/.

4. R. Girshick, J. Donahue, T. Darrell , J. Malik and UC Berkeley, "Rich feature hierarchies for accurate object detection and semantic segmentation", IEEE Int. Conf. on Computer Vision and Pattern Recognition, USA, June 2014.

5. G.Jocher, https://github.com/ultralytics/yolov5, 2020.

6. G. C. De Silva, "Automation of Traffic Flow Measurement Using Video Images," Master of Engineering, University of Moratuwa, Sri Lanka, 2001.

7. R. J. Franklin, Mahana -Traffic Signal Violation Detection using Artificial Intelligence and Deep Learning, ‖ International Conference on Communication and Electronics Systems (ICCES), 2020, pp. 839-844..

8. Hendry, Rung-Ching Chen "Automatic License Plate Recognition via sliding-window darknet- YOLO deep learning", Image and Vision Computing 87 (2019) ELSEVIER 47-56.

9. A. Ambardekar, et al., "Efficient Vehicle Tracking and Classification for an Automated Traffic Surveillance System," in International

Conference on of Signal and Image Processing, 2008, pp. 1-6.

10. A. Gomaa, M. M. Abdelwahab, -Robust Vehicle Detection and Counting Algorithm Employing a Convolution Neural Network and Optical Flow, ‖ Sensors, vol. 19, no. 20, 2019.

11. He, K.; Gkioxari, G.; Dollar, P.; Girshick, R. In Mask r-cnn, Proceedings of the IEEE international conference on computer vision, 2017; 2017; pp 2961-2969.

12. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y. M., YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv preprint arXiv:2004.10934 2020.

13. Ren, S.; He, K.; Girshick, R.; Sun, J. In Faster r-cnn: Towards real-time object detection with region proposal networks, Advances in neural information processing systems, 2015; 2015; pp
91-99.

14. Duan, K.; Bai, S.; Xie, L.; Qi, H.; Huang, Q.; Tian, Q. In Centernet: Keypoint triplets for object detection, Proceedings of the IEEE International Conference on Computer Vision, 2019; 2019; pp 6569-6578.

15. Rakhimkul, S.; Kim, A.; Pazylbekov, A.; Shintemirov, A. In Autonomous Object Detection and Grasping Using Deep Learning for Design of an Intelligent Assistive Robot Manipulation System, 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), 2019; IEEE: 2019; pp 3962-3968.